Toward predicting turbulence structure in street canyons using deep learning

By T. M. Jaroslawski, A. Patil AND B. J. McKeon

1. Motivation and objectives

Air quality in urban environments is a pressing contemporary issue, with significant socio-economic implications. Understanding the urban climate poses challenges due to the geometric complexity of built areas and the presence of numerous interacting thermodynamic processes. Turbulence plays a fundamental role in determining the instantaneous dynamics of airflow. Specifically, the atmospheric flow, combined with the complex geometry of the urban canopy, exhibits pronounced multi-scale characteristics, both in space and time. The canyon geometry and roughness are found to be important parameters, with complex non-linear amplitude modulations occurring between the large scales in the boundary layer and the separated low-frequency flapping shear layer at roof level. Therefore, comprehending the spatial structure of such flows is crucial, especially when investigating transient phenomena such as accidental pollutant releases or predicting flow states with a limited number of sensors.

Recently, machine learning has been applied to study urban flow dynamics. This includes computing pollution concentrations from mobile field data (Alas *et al.* 2022), examining inter-scale turbulent interactions over obstacle arrays (Liu *et al.* 2023), determining drag coefficients on buildings using large eddy simulations (LES) (Lu *et al.* 2023), and even developing reduced-order models of flow dynamics (Xiao *et al.* 2019). However, there are few works leverage machine learning (ML) algorithms systematically to predict spatial-temporal complex turbulent physics, especially when using simplified idealized experimental data for training to predict results in more complex conditions.

Several studies have focused on spatial and temporal reconstruction, as well as spatial supersampling (Schmidt *et al.* 2021). Hybrid deep neural network architectures have been designed to capture the spatial-temporal features of unsteady flows (Han *et al.* 2019), and machine learning—based reduced-order models have been proposed for three-dimensional complex flows (Nakamura *et al.* 2021). A deep learning framework combining long short-term memory networks and convolutional neural networks has been used to predict the temporal evolution of turbulent flames (Ren *et al.* 2021).

New deep learning architectures, including transformers, are emerging for temporal problems in structured and unstructured data. Inspired by convolutional neural networks, transformers build input features using self-attention to assess the relevance of other data points in the dataset, without relying on recurrence. They excel in natural language processing tasks and are replacing traditional recurrent neural networks like long short—term memory networks. Transformers have also been applied in spatio-temporal contexts, such as video analysis. However, they have not been used for spatio-temporal prediction in experimental flow fields involving turbulent flows in urban street canopies.

The primary focus of this brief is to tackle the difficult task of understanding urban flow dynamics, which are characterized by their complex nature, non-linear interactions, and high-dimensional data. To address this challenge, we employ an experimental PIV dataset

and introduce an approach that leverages a convolutional encoder-decoder transformer model along with autoregressive training to make accurate spatio-temporal predictions. Our findings demonstrate the performance of this method in forecasting flow patterns and turbulence structure, underscoring the promise of data-driven techniques for solving intricate problems related to urban flows.

2. Methods

2.1. *The experiment*

The experimental data was obtained from Jaroslawski *et al.* (2019, 2020) which consited of wind tunnel campaigns conducted at École Centrale de Nantes, France, using a low-speed boundary-layer wind tunnel measuring 2 m (width) × 2 m (height) × 24 m (length) with a 5:1 inlet contraction. The experimental setup is presented in Figure 1. Street canyons with aspect ratios of 1 and 3, using various upstream roughness configurations like staggered cubes or spaced bars, were explored. Stereoscopic particle image velocimetry (PIV) measurements were taken horizontally at a height of $0.9h \pm 0.05h$. The PIV setup, situated beneath the wind-tunnel floor, utilized a Litron double cavity laser and DANTEC Dynamic Studio software to generate vector fields with a 1.6 mm spatial resolution and a 7 Hz sampling frequency. An iterative cross-correlation analysis computed velocity vector fields using a 64 × 64 pixel window size and a 32 × 32 pixel interrogation window with a 50% overlap and a pulse interval of 500 μs. Measurement uncertainties were estimated at 0.9%, 1.4%, and 3.9% for mean velocity, standard deviation, and turbulent shear stress, respectively, based on 2551 independent samples from 10 000 velocity field recordings. The freestream velocity of $U_e = 5.9$ ms^{-1}, measured using a pitot-static tube, remained constant across experiments, resulting in a Reynolds number of 1.9×10^4 based on this speed and canyon height, h.

2.2. *Machine learning framework*

Transformers can be combined with convolutional encoder-decoder models for optimal performance when dealing with spatio-temporal data. This combination is effective for various computer vision tasks, including video frame prediction. The self-attention mechanism on convolutional layers enhances spatial representation by focusing on crucial features and suppressing less important ones. Spatio-temporal learning is formulated as a task with a given time-series containing N sequential snapshots $\left[x_t, x_{t+\Delta t}, ..., x_{t+(N-1)\Delta t}\right]$, in order to predict the same quantity of interest on M steps ahead in time. The input X of the deep learning model is $\left[x_t, x_{t+\Delta t}, ..., x_{t+(N-1)\Delta t}\right]$, and the output Y is $\left[x_{t+N\Delta t}, ..., x_{t+N+(M-1)\Delta t}\right]$. Each snapshot x_t can be a scalar field or a vector field containing multiple features.

Encoder-decoder architectures are employed to estimate the reduced or latent state. The encoder extracts relevant information from input tensors and maps it to a high-dimensional representation. The decoder converts this representation to target output tensors using up-sampling and convolutions. The encoder and decoder weight matrices jointly map input to output, enabling small-scale feature learning. The decoder transforms the latent space $(n_z \times n_z)$ to the original spatial dimensions at $x_{t+\Delta t}$. When a transformer block follows a convolutional layer, the model learns to emphasize significant features across channels and spatial dimensions. Initially, input sequences are concatenated channel-wise to the input layer, followed by convolutional operations in the encoder. In convolutional layers, intermediate feature maps $\mathbb{F} \in \mathbb{R}^{C \times H \times W}$ (where

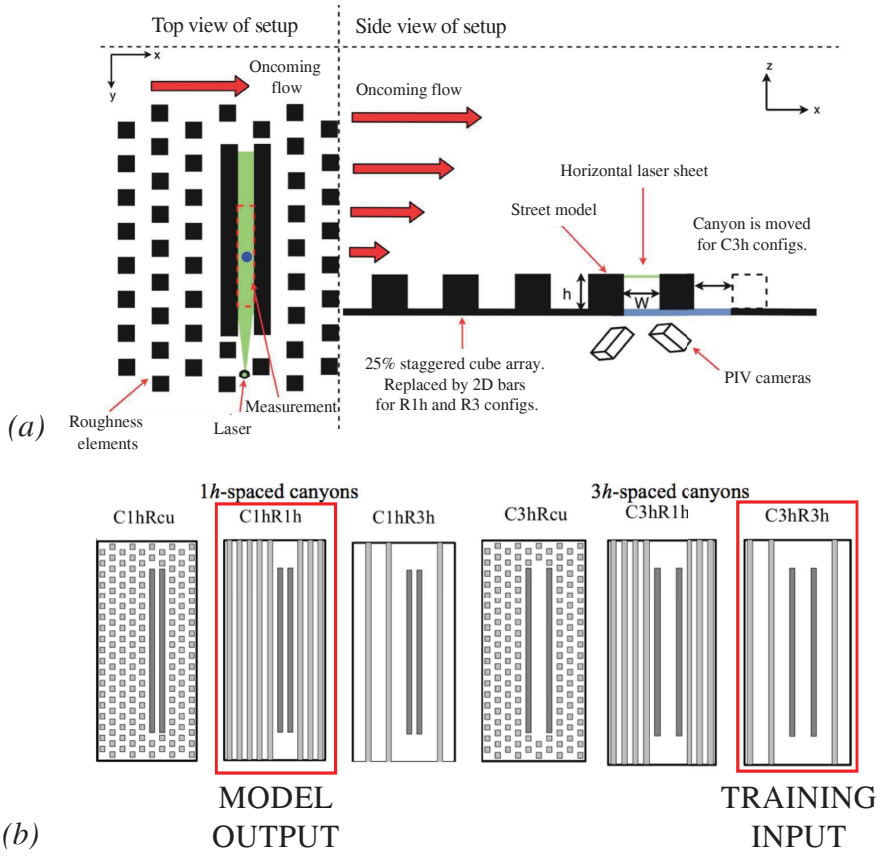

FIGURE 1. (a) Schematic of a street canyon roughness setup in a wind tunnel. Blue point located in the middle of the canyon ($x_{ref} = 0, y_{ref} = 0$) denotes the reference point used for two-point correlations. (b) Experimental configurations studied with specified training inputs and outputs for the ML model.

H and W are input resolutions for each snapshot and C is the amount of model layers) from a specific layer pass through the self-attention convolutional transformer layer, which considers spatial representation and positional embeddings of input sequence channels. This layer has a 3×3 kernel and incorporates convolutional features. Combining convolutional neural networks with self-attention enhances learning of spatio-temporal structures, benefiting turbulent flows by capturing spatial filters and temporal dependencies. In turbulent flow problems, dimensionality reduction techniques are valuable due to complex spatio-temporal dynamics. In addition to the convolutional transformer layer, the model is trained in an autoregressive fashion. Formally, autoregressive models are those which forecast future sequences from the previously forecasted sequences in a cyclical way, and thus here auto indicates the regression of the variable sequence against itself. For a trained model $\mathbb{M}$ as shown in Figure2, multi-step training is performed for quantity X_t in an auto-regressive manner, $X_{t+\Delta t}$ is predicted from previously predicted X_t, where t is some non-dimensional time. In other words, an initial condition X_t is inputted to the model to learn $\widehat{X}_{t+\Delta t}$, after this predicted $\widehat{X}_{t+\Delta t}$ is then fed back to the

model again to learn $\widehat{X}_{t+2\Delta t}$ and so on, in an autoregressive manner:

$$\begin{cases} \widehat{X}_{t+\Delta t} = \mathbb{M}(X_t), \\ \widehat{X}_{t+2\Delta t} = \mathbb{M}(\widehat{X}_{t+\Delta t}), \\ \qquad\qquad \cdots \\ \widehat{X}_{t+(n-1)\Delta t} = \mathbb{M}(\widehat{X}_{t+(n-2)\Delta t}), \end{cases} \tag{2.1}$$

where t is the time step and $X \in \mathbb{R}^{C\times H\times W}$ is the input tensor snapshot at instant t. In the following, the autoregressive training sequence length is set equal to two in order to limit the computational cost. To train the model, the Adam optimizer (Kingma & Ba 2014) is used to iteratively minimize the total equi-weighted mean squared error (MSE) loss defined by

$$\mathcal{L} = \frac{1}{n_s}\Bigg[\sum_{i=1}^{n_s}\left((X_{t+\Delta t})^i - \left(\widehat{X}_{t+\Delta t}\right)^i\right)^2 + \sum_{i=1}^{n_s}\left((X_{t+2\Delta t})^i - \left(\widehat{X}_{t+2\Delta t}\right)^i\right)^2 + \cdots + \sum_{i=1}^{n_s}\left(\left(X_{t+(n-1)\Delta t}\right)^i - \left(\widehat{X}_{t+(n-1)\Delta t}\right)^i\right)^2\Bigg]. \tag{2.2}$$

The activation function used in the neural network was ReLU, which is known to help stabilize the weight update during training (Nair & Hinton 2010). During training, the entire training dataset was presented to the network repeatedly after shuffling, and each complete pass is called an epoch. An early stopping criterion was used to stop the training process, along with a learning rate reduction if learning improvement did not occur after every 100 epochs. The TensorFlow library (Abadi *et al.* 2016) was used to implement the deep learning architecture, and Nvidia RTX A4500 GPU was used to train it. In turbulent flow problems, the high-dimensional state-space is characterized by intricate spatio-temporal dynamics, and therefore, dimensionality reduction techniques can be useful. The prediction and reconstruction problems are interpreted as estimating the reduced or latent state, making it natural to use encoder-decoder architectures. As, shown in Figure 2, the encoder takes input tensors, learns the most relevant parts, and maps them to a spatially low-dimensional representation. This spatially low-dimensional representation is then converted to target output tensors by the decoder, which involves successive up-samplings and convolutions. By connecting the encoder and decoder, their weight matrices learn to jointly map the input to the output tensors, allowing small-scale features to be learned. The decoder aims to transform the latent space representation with a dimension of $n_z \times n_z$ to the original spatial dimensions of the target output at time $t_{t+\Delta t}$. When a transformer block follows a convolutional layer, the model learns to highlight significant features across the channel sequence and spatial dimensions. The input sequences are initially concatenated channel-wise to the input layer, and subsequent convolutional operations take place in the encoder.

3. Results

This section presents comparisons between the experimental data and the ML model. Firstly, the mean flow is examined, followed by the results of two-point correlations on both the experimental and ML data. Subsequently, potential model improvements are discussed, followed by conclusions.

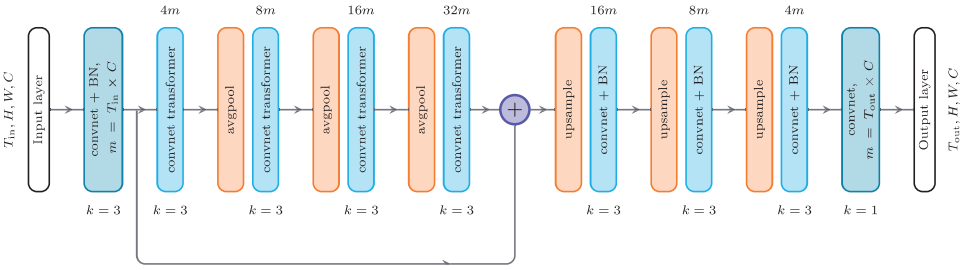

FIGURE 2. Convolutional encoder-decoder transformer deep learning architecture : Model architecture of the convolutional encoder-decoder transformer to process low and high level features. The canonical four-stage design is utilized in addition to the convolutional transformer blocks or layers. H, W are the input resolutions for each snapshot in T_{in} sequence and T_{out} sequence, k is the kernel size, and m, the number of filters.

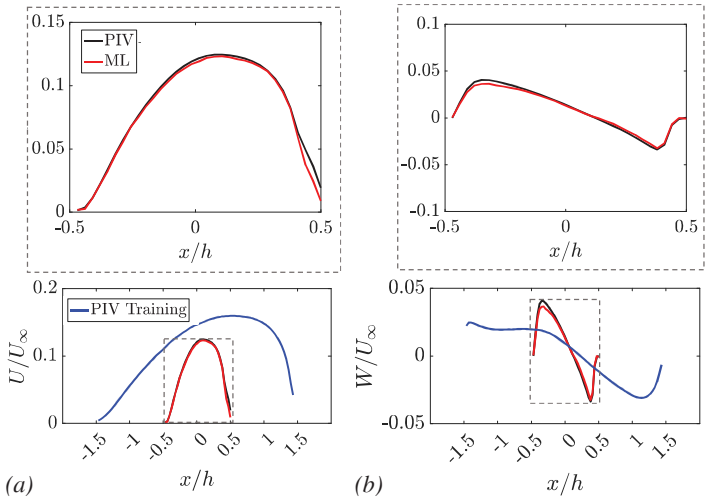

FIGURE 3. Time- and spatially- averaged turbulent statistics: (a) streamwise velocity component and (b) vertical velocity component observed in both the training data and the predicted data. ML represents the model's prediction, while PI' references the experimental data.

3.1. *Mean flow*

The ML model is trained using the C3hR3h configuration, and its predictive capabilities are evaluated by applying it to the C1hR1h configuration, which features a smaller aspect ratio and a different upstream roughness setup. Figure3 presents the training data and compares the model's predictions with the test experimental data, illustrating the evolution of spanwise-averaged streamwise roof-level U (streamwise) and W (vertical) velocity profiles. A significant difference between the training and test datasets is observed due to distinct flow regimes. The test case exhibits a skimming flow regime, contrasting the wake interaction flow regime seen in the training data (Oke 1988), which ensures substantial differences between the training and test datasets. Particularly notable is the W velocity component, indicating a larger and asymmetric recirculation that is skewed towards the aft portion of the canyon. Upon examining the model's predictive capabilities, a remarkable agreement is observed between the model's results and the experimental findings concerning mean flow.

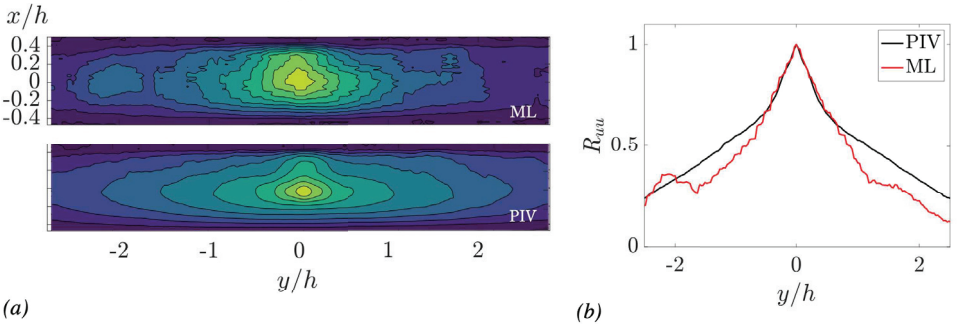

FIGURE 4. (a) Two-point correlation fields, R_{uu}, of the ML model and PIV dataset. (b) Spanwise slice at $x = 0$ of the R_{uu} field.

3.2. *Two-point correlations*

To further evaluate the model's capabilities in predicting turbulence structure, we compared its results to a two-point spatial correlation analysis of the streamwise velocity fluctuation, denoted as R_{uu}. Two-point spatial fluctuating velocity correlations offer important information regarding the structure of the flow field that single-point measurements are unable to provide. A two-point spatial correlation was conducted using the middle of the street canyon as the reference point. The two-point correlation coefficient was computed using

$$R_{uu} = \frac{\overline{u'(x_{ref}, y_{ref})u'(x, y)}}{\sqrt{\overline{u'(x_{ref}, y_{ref})}}\sqrt{\overline{u'(x, y)}}}. \tag{3.1}$$

Figure 4(a) shows contour fields of R_{uu} for both the ML model and the PIV data. The ML model effectively captures the general spatial structure, particularly near the reference point $x_{ref} = 0, y_{ref} = 0$. In Figure 4(b), a spanwise slice is presented at the streamwise position of $x = 0$. This further shows the model's capability to predict fluctuations' decorrelation at smaller spatial lags, indicating its potential in forecasting the spanwise turbulence structure. We note that the noise observed in the ML results can be attributed to the use of 400 snapshots for computations, whereas the experimental data utilized 10000 snapshots.

3.3. *Future model improvements*

The presented ML model exhibits potential for improvement. Figure 5 displays the spanwise-averaged streamwise roof-level standard deviation profiles for the streamwise and vertical velocity components, denoted as σ_u and σ_w, respectively. These profiles illustrate the comparison between the ML model trained with one configuration (solid red line) and two configurations (dashed red line). Referring to Figure 5, there is an underprediction and incorrect spatial evolution in both σ_u and σ_w. However, after incorporating an additional configuration (C3hR1h) into the model's training, there's an improvement in agreement with the experiment. Specifically, the spatial evolution of the standard deviation is closer to the experimental data, albeit accompanied by a slight overestimation.

This observation motivates further refinement of the ML model. Our proposed approach involves leveraging findings from a quadrant analysis and the computation of amplitude modulation coefficients derived from experimental data. We aim to utilize

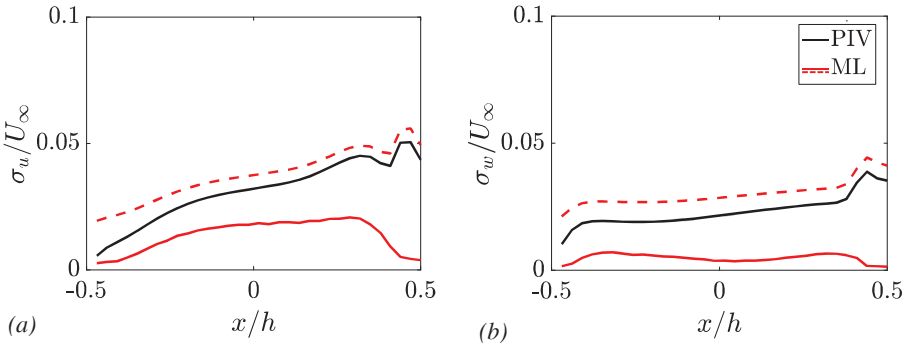

FIGURE 5. Time- and spatially- averaged turbulent statistics. Standard deviation of the (a) streamwise velocity component and (b) vertical velocity component. Dashed lines denote ML model trained with two PIV data sets.

these outcomes as informed constraints, integrating physics-based insights to enhance the predictive capacity of the present ML model.

4. Conclusions

Urban flow dynamics exhibit high Reynolds numbers nonlinear interactions; and involve high-dimensional data, posing challenges for simulation and modeling. To tackle these complexities, we leverage an ML algorithm utilizing a convolutional encoder-decoder transformer model with autoregressive training for accurate spatio-temporal predictions. An experimental PIV dataset of a street canyon flow is used as the training input.

Our findings demonstrate the ML model's ability to predict mean flow fields and capture the spanwise structure of roof-level turbulence in the street canyon. The results outlined in this brief highlight the framework of the ML model and underscore the clear potential for further refinement, enabling predictions of higher-order statistics.

Acknowledgments

Partial support of ONR under grant N00014-22-1-2150 is gratefully acknowledged. The data was obtained by T.M. Jaroslawski during his stay at École Centrale de Nantes, under the supervision of L. Perret and E. Savory.

REFERENCES

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. *et al.* 2016 Tensorflow: a system for large-scale machine learning. In *12th USENIX Sym Oper Syst Des Implement (OSDI 16)*, pp. 265–283.

Alas, H. D., Stöcker, A., Umlauf, N., Senaweera, O., Pfeifer, S., Greven, S. & Wiedensohler, A. 2022 Pedestrian exposure to black carbon and pm2. 5 emissions in urban hot spots: new findings using mobile measurement techniques and flexible bayesian regression models. *Expo Environ Epidemiol* **32**, 604–614.

Han, R., Wang, Y., Zhang, Y. & Chen, G. 2019 A novel spatial-temporal prediction method for unsteady wake flows based on hybrid deep neural network. *Phys Fluids* **31**.

Jaroslawski, T., Perret, L., Blackman, K. & Savory, E. 2019 The spanwise variation of roof-level turbulence in a street-canyon flow. *Bound-Layer Meteorol* **170**, 373–394.

Jaroslawski, T., Savory, E. & Perret, L. 2020 Roof-level large-and small-scale coherent structures in a street canyon flow. *Environ Fluid Mech* **20**, 739–763.

Kingma, D. P. & Ba, J. 2014 Adam: Method for stochastic optimization. *arXiv Preprint* .

Liu, W., Zou, Y. & Li, X. 2023 Study of interscale interactions for turbulence over the obstacle arrays from a machine learning perspective. *Phys Fluids* **35**.

Lu, Y., Zhou, X.-H., Xiao, H. & Li, Q. 2023 Using machine learning to predict urban canopy flows for land surface modeling. *Geophys Res Lett* **50**, e2022GL102313.

Nair, V. & Hinton, G. E. 2010 Rectified linear units improve restricted boltzmann machines. In *Proc 27th ICML-10*, pp. 807–814.

Nakamura, T., Fukami, K., Hasegawa, K., Nabae, Y. & Fukagata, K. 2021 Convolutional neural network and long short-term memory based reduced order surrogate for minimal turbulent channel flow. *Phys Fluids* **33**.

Oke, T. R. 1988 Street design and urban canopy layer climate. *Ener Build* **11**, 103–113.

Ren, J., Wang, H., Chen, G., Luo, K. & Fan, J. 2021 Predictive models for flame evolution using machine learning: *apriori* assessment in turbulent flames without and with mean shear. *Phys Fluids* **33**.

Schmidt, D., Maulik, R. & Lyras, K. 2021 Machine learning accelerated turbulence modeling of transient flashing jets. *Phys Fluids* **33**.

Xiao, D., Heaney, C., Mottet, L., Fang, F., Lin, W., Navon, I., Guo, Y., Matar, O., Robins, A. & Pain, C. 2019 A reduced order model for turbulent flows in the urban environment using machine learning. *Build Environ* **148**, 323–337.