Deep learning to predict extreme wind loads on high-rise buildings

Giacomo Lamberti (giacomol), William McCloskey (mcclow12)
Stanford University

1 Introduction

In recent decades, the number of high-rise buildings drastically increased following the trend of urban expansion. These tall buildings can be subjected to extreme wind pressure loads, since wind velocity increases over altitude; therefore, the correct estimate of the wind loads acting on these structures is critical.

Several methods can be employed to predict wind loads on buildings and their components, such as wind tunnel tests and numerical simulations. The most common approach is to rely on wind tunnel experiments, which nowadays represent the most reliable method for computing wind loads on tall buildings. During a wind tunnel experiment, air is blown onto the building and pressure time-series are recorded around the building's surface. The main drawbacks of wind tunnel tests are: the high operational cost and the fact that the models need to be scaled in order to fit inside the facility.

On the other hand, computational fluid dynamics (CFD) has the capability to provide detailed information in complex geometries, at real scale, together with a substantial reduction in cost. CFD simulations discretize and solve the Navier-Stokes equations, which describe mass and momentum conservation of air around the building [1]. Two CFD models, characterized by different level of fidelity and computational cost, are available: large-eddy simulations (LES) and Reynolds-averaged Navier-Stokes simulations (RANS). In particular, RANS simulations take the time-average of the Navier-Stokes equations and output the following time-averaged quantities: mean pressure (P) and velocity (U), turbulence kinetic energy (k), turbulence dissipation rate (ϵ) and turbulence viscosity (ν_t) [1]. The averaging procedure drastically reduces the computational cost, making RANS simulations suitable for wind engineering problems; however, the pressure acting on the building is not stationary and in fact can fluctuate significantly, resulting in extreme pressure events, which determine the design pressure loads.

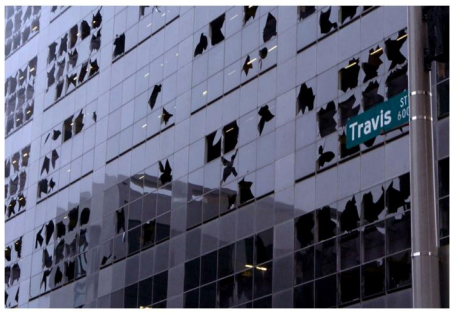

Figure 1: Photo courtesy of the U.S. Navy/Interior Communications Electrician 1st Class Jason Stephens

Therefore, we need some models to estimate pressure peaks from time-averaged quantities, in order to extensively employ CFD simulations in the design process. A common approach is to estimate the root mean square (rms) of the pressure signal, to get an idea of the possible deviations from the mean. In this context, the goal of the present work is to employ deep learning to formulate a model that relates the rms pressure to time-averaged quantities. Specifically, the input features of our algorithm consist of 8 quantities constructed from RANS variables. We then use neural networks to output the rms pressure, and compare the result to the wind tunnel measurements.

2 Related work

Various empirical models that relate the rms pressure to the time-averaged pressure and velocity, have been formulated in the past [2]. These models allow one to efficiently estimate the pressure fluctuations using the outcome of RANS simulations. These empirical models were formulated using wind tunnel data, acquired during experiments on low-rise buildings [3, 4, 5, 6]. Not only was the geometry not representative of a high-rise building design, but also the measurements were limited to few locations around the building. As a result, none of these relationships produces accurate results, especially when considering high-rise building's lateral facades [2]. The main limitation of these models is that they assume a linear relationship between rms and mean pressure, which is not verified by the available wind tunnel measurements.

Alternatively, we can employ a data-driven approach to find the functional form that better represents the data. Machine learning in CFD is a relatively recent topic. Some work has been done to relate RANS time-averaged quantities to high-fidelity data. In [7] the authors compute 20 physical quantities from RANS outputs, and use these as features in a machine learning procedure, that aims to estimate regions of high uncertainty in RANS simulations. Inspired by their work, we compute some of the same physical quantities from our RANS simulations, and use neural networks to relate these to wind tunnel data.

3 Dataset and Features

3.1 Wind tunnel measurements

The model of the high-rise building that we consider, is a $1m$ long, $0.3m$ wide and $2m$ high rectangular box, representative of a $100m$ tall building in full-scale, which was tested in the wind tunnel of "Politecnico di Milano" [8]. The model was equipped with 2 aluminum tiles containing 224 pressure taps each, with a minimum spacing of $3mm$ (Figure 2). Tile A is located on the top-corner of the model while tile B is centered at $1m$ height and adjacent to the building edge (Figure 2).

The configuration was designed to allow a very detailed study of the pressure distribution in the regions of the building, where the largest pressure peaks are expected, namely near the corners and edges. The model was placed at the center of a turntable to allow testing at different wind directions.

In the present work we focus on eight exposure angles: $0°$, $10°$, $20°$, $170°, 180°$, $190°$, $200°$ and $260°$, following the convention defined in Figure 2. The outcome of the test consists of $300s$ time-series of pressure p, sampled at a rate of $500Hz$. From these time-series, we can compute the rms pressure coefficient (i.e. the output quantity of interest of our deep learning model), defined as:

$$C_{p,rms} = \frac{\sqrt{var(p)}}{p_{ref}},$$

where p_{ref} is a reference pressure, measured somewhere upstream of the model.

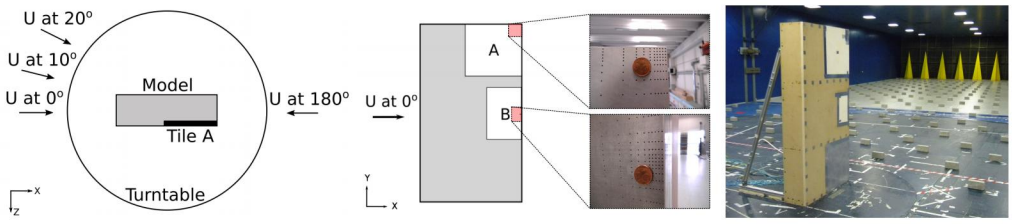

Figure 2: wind tunnel experiment on the high-rise building model: wind direction convention (left), sketch of the building and measurement locations (center) and photo of the actual test (right).

3.2 RANS

RANS simulations solve the Reynolds-averaged Navier-Stokes equations, which are obtained by first decomposing the instantaneous physical quantities in a time-averaged and a fluctuating component, and then by taking the average of the Navier-Stokes equations [1]. The outcome of these simulations

are: mean pressure (P) and velocity (U); turbulence kinetic energy (k), dissipation rate (ϵ) and viscosity (ν_t). The main advantage of RANS simulations is the relatively low computational cost, which comes at the price of lower accuracy and the need of an additional model to retrieve the information on the time-evolution of the flow.

From these simulations we get the features $\mathbf{x}$ of our deep learning models. Specifically, we construct 8 physical quantities:

$$\mathbf{x} = \mathbf{x}(P, U, k, \epsilon, \nu_t, \nabla P, \nabla U),$$

which are function of RANS outcomes and their spatial gradients [7].

3.3 Dataset Split

Accounting for all the wind directions and pressure taps locations, our dataset consists of $224 \times 2 \times 8 = 3,584$ examples. To train, develop and test our models, we divide our data by wind directions, as highlighted in Table 1. The rationale behind this choice is that we would like the model to be able to generalize to unseen wind directions, namely $20°$ in this case. Moreover, for the $190°$ and $260°$ configurations, we employ one of the tiles (A or B) for training and the other for validation; the idea here is to find the model that better generalizes to unseen areas of the building.

	Train	**Dev**	**Test**
Tile A	$0°, 10°, 170°, 180°, 200°, 260°$	$190°$	$20°$
Tile B	$0°, 10°, 170°, 180°, 200°, 190°$	$260°$	$20°$

Table 1: data-split

To test the performance of the models in extrapolating to unseen regions of the building (i.e. gray area in Figure 2 in the center), we employ data from a high-fidelity simulation (LES) performed at $0° - 180°$ configuration (employed only at test time).

4 Methods

We employ neural networks for our task, and specifically two different architectures: an artificial neural network (ANN) and a convolutional neural network (CNN).

ANNs can learn highly non-linear mapping from the input features to the output quantity of interest. In this setup, we learn a function $f(\mathbf{x}) : \mathbb{R}^8 \rightarrow \mathbb{R}$; therefore, each example is treated independently and we lose information on the correlation between close data points. A sketch of the model is shown in Figure 3.

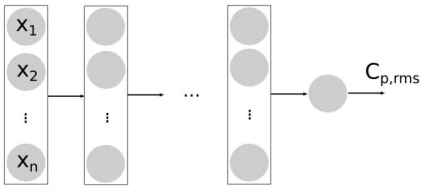

Figure 3: sketch of the artificial neural network

CNNs are often employed when dealing with images, because of their ability to take advantage of the correlation between nearby pixels. In the present problem, we can view each pressure tile (regions A and B in Figure 2) as a 2D image with 224 pixel values. The input of the learning algorithm will now consist of 2D images with 8 channels (i.e. the features values), while the output will consist of 2D images with a single channel (i.e. the rms C_p value). The main limitation of this approach is the amount of data: by considering each pressure tile as a 2D image, we end up with only 16 independent examples. Moreover, the pressure taps are not uniformly distributed throughout the tiles and the two tiles (A and B) have different number of rows and columns. We decide to represent each tile as a $(\max\{\text{nrow}_A, \text{nrow}_B\} \times \max\{\text{ncol}_A, \text{ncol}_B\}) = (15 \times 18)$ image, and use zero-padding to fill the missing values.

In this setting, we would like to learn a function $f(\mathbf{x}) : \mathbb{R}^{(15\times18\times8)} \rightarrow \mathbb{R}^{(15\times18)}$.

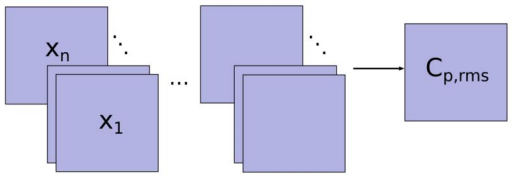

Figure 4: sketch of the convolutional neural network

5 Results

In this section, we first describe the hyper-parameters search, performed for the two models, and then we compare the performance of the methods over the test sets.

5.1 Hyper-parameter tuning

The tuning was performed over 500 evaluations for the ANN, and $1,000$ evaluations for the CNN, since it is cheaper to train. Within all the trained models, we selected the two that had lower mean-squared error on the validation set; table 2 summarizes the hyper-parameters of these models. Both models use Adam optimization algorithm, which at this point is not part of the tuning process.

Hyper-parameter	**ANN**	**CNN**
# epochs	300	1,000
# hidden layers	5	1
# hidden units	7	-
activation	reLu	reLu
learning rate	0.01	0.01
dropout	None	None
batch-normalization	yes	None
# number of filters	-	14
filter size	-	1

Table 2: hyper-parameters optimal values

5.2 Performance on the test sets

To evaluate the performance of the models, we employ two different test sets. The first one is a leftover wind direction (i.e. $20°$) from the wind tunnel database, to test the ability of the models in extrapolating to different wind directions. The idea here is that we always have data at a limited number of wind directions, but the design pressure still needs to be representative of all possible wind directions. The second test set comes from a high-fidelity simulation, which allows us to compute the rms C_p around the entire building's facade and evaluate the ability of the models in extrapolating to different regions of the building. The idea here is that we always have limited measurement locations, but we want to be able to compute the pressure fluctuations everywhere around the building.

To evaluate the overall performance of the two models, we compute the mean percentage error, defined as follows:

$$\frac{\Sigma_i |C_{p,rms;i} - \hat{C}_{p,rms;i}|}{\Sigma_i |C_{p,rms;i}|}$$

where $\hat{C}_{p,rms;i}$ is the prediction of our models, while $C_{p,rms;i}$ is the reference value. Table 3 summarizes the results of the two methods over train, dev and test sets. It seems that the ANN performs better in both tasks of generalizing to different regions and wind directions. The worse performance of the CNN can be possibly explained by the very limited amount of training examples and by the fact that the pixels are not uniformly distributed around the image.

Set	ANN	CNN
Train	4.8%	21.5%
Dev	1.7%	8.8%
Test: tiles A-B at 20°	10.0%	24.6%
Test: whole building at 0°, 180°	3.9%	11.5%

Table 3: mean percentage error

Figure 5 shows the $C_{p,rms}$ field around tiles A and B as predicted from the two neural networks and compared to the experimental data (on the right). Both models do a decent job in predicting the pressure fluctuations at an unseen wind direction, especially the ANN (Figure 5 on the left).

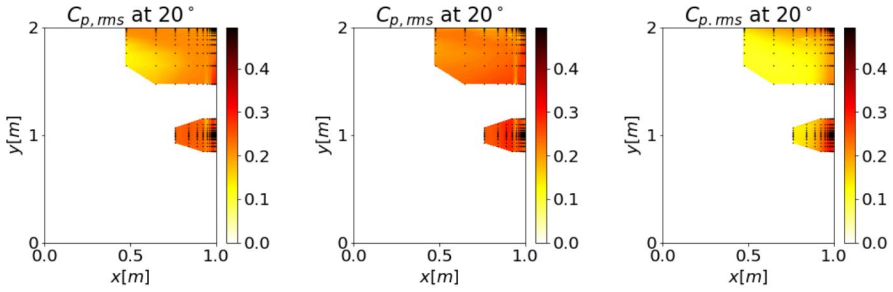

Figure 5: rms C_p around tiles A-B at 20°, as predicted by the ANN (left), CNN (center) and measured in the wind tunnel (right)

Finally, Figure 6 shows the $C_{p,rms}$ field throughout the whole building's facade as predicted from the two neural networks and compared to the high-fidelity simulation (on the right). Both models are able to capture the trend over the entire facade, even if training only on tiles A and B.

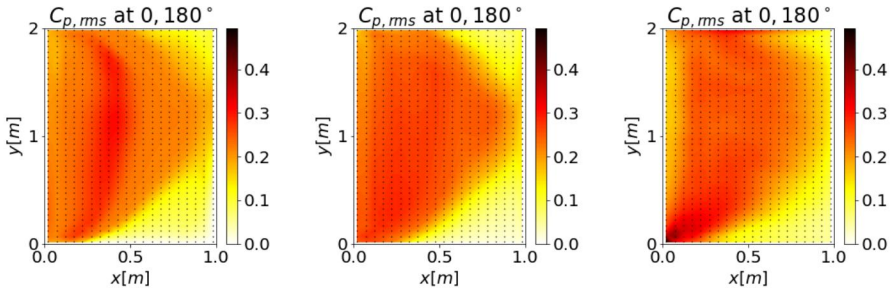

Figure 6: rms C_p around the whole building's facade at $0° - 180°$, as predicted by the ANN (left), CNN (center) and high-fidelity simulation (right)

6 Conclusion and future work

In the present work, we developed a deep learning approach to efficiently and reliably estimate pressure fluctuations, given inexpensive RANS simulations. Both methods do a pretty good job in generalizing to unseen regions of the building and different wind directions, especially the ANN. This raises hope about the possibility of employing the models in different configurations and geometries. Future work will focus on investigating the reasons why the CNN performs less well than the ANN, possibly the limited amount of data or the way we constructed the 2D images. Finally, we will test the models in different geometries.

Link to the code: `https://github.com/giacomolamberti90/CS230_project`

References

[1] Stephen B Pope. Turbulent flows, 2001.

[2] IM Kalkman, AJ Bronkhorst, CA van Bentum, and J Franke. A comparison of rans computations and wind tunnel tests for rms pressures on a high-rise building model. In *Proceedings of the Seventh International Colloquium on Bluff Body Aerodynamics and Applications (BBAA7) Shanghai, China; September 2-6, 2012, 1-10*, 2012.

[3] D A Paterson, J Holmes, et al. Computation of wind flow around the texas tech building. 1989.

[4] R Panneer Selvam. Computation of pressures on texas tech building. *Journal of Wind Engineering and Industrial Aerodynamics*, 43(1-3):1619–1627, 1992.

[5] DA Paterson. Predicting rms pressures from computed velocities and mean pressures. In *Computational Wind Engineering 1*, pages 431–437. Elsevier, 1993.

[6] PJ Richards and BS Wanigaratne. A comparison of computer and wind-tunnel models of turbulence around the silsoe structures building. *Journal of Wind Engineering and Industrial Aerodynamics*, 46:439–447, 1993.

[7] Julia Ling and J Templeton. Evaluation of machine learning algorithms for prediction of regions of high reynolds averaged navier stokes uncertainty. *Physics of Fluids*, 27(8):085103, 2015.

[8] L. Amerio. *Experimental high resolution analysis of the pressure peaks on a building scale model façades*. PhD thesis, Politecnico di Milano, 2018.