Scientific Research Publishing

Open Journal of Civil Engineering, 2025, 15(3), 341-369
https://www.scirp.org/journal/ojce
ISSN Online: 2164-3172
ISSN Print: 2164-3164

Prediction of Building Displacements during Wind Gusts Using Support Vector Regression and Deep Neural Network

Gustavo Cristante Izar, Reyolando Manoel Lopes Rebello Da Fonseca Brasil

Escola Politécnica, Universidade De São Paulo, São Paulo, Brazil
Email: gustavo.cristanteizar@hilti.com

How to cite this paper: Cristante Izar, G. and Lopes Rebello Da Fonseca Brasil, R.M. (2025) Prediction of Building Displacements during Wind Gusts Using Support Vector Regression and Deep Neural Network. *Open Journal of Civil Engineering*, **15**, 341-369.
https://doi.org/10.4236/ojce.2025.153019

Received: June 17, 2025
Accepted: August 18, 2025
Published: August 21, 2025

Abstract

The significance of artificial intelligence in contemporary human activities cannot be underestimated. Although the application of this technology in structural engineering is not recent, advancements in computational capacity and recent developments in artificial neural networks have introduced a new range of possibilities and a distinct perspective on addressing complex engineering problems. The dynamic analysis of tall structures under wind action stands for an overly complex engineering challenge. In this context, this research aims to develop two computational solutions: the first generates synthetic wind gusts and models their static and dynamic loads, while the second employs support vector regressions and deep neural networks to predict the temporal response (maximum horizontal displacements) of a reinforced concrete building subjected to these synthetic wind loads. The results are analyzed and compared with responses obtained from the finite element software Ansys Mechanical. This study explores emerging domains in structural and computational engineering, as, although machine learning algorithms are not exhaustively addressed, their application to predict structural behavior under complex dynamic loadings is investigated.

Keywords

Machine Learning, Structural Engineering, Synthetic Wind

1. Introduction

Artificial Intelligence is an interdisciplinary field that integrates studies from computer science, statistics, mathematics, psychology, and neuroscience to replicate human cognitive functions through artificial means. One of the earliest strategies

to simulate human cognition was to define "knowledge" as a collection of interrelated facts. It was believed that, once a database of "knowledge" on a specific topic was constructed, its inherent interconnections could be revealed, and rules could be created to answer any questions. To this end, algorithms were developed with the ability to learn to identify these interconnections. The area of Artificial Intelligence where algorithms possess this capability is called Machine Learning. In this area, algorithms are created and refined through training data without explicit programming. Machine Learning programs can thus identify patterns in complex data under uncertainty through learning. One form of learning from training data is through message-passing architectures, which model artificial intelligence based on information transfer, as in human neural networks. Artificial Neural Networks (ANNs) are Machine Learning algorithms that utilize this model.

Among the Artificial Neural Networks developed, Convolutional Neural Networks (CNNs) have proven highly efficient in image recognition and computer vision due to their intrinsic relationship with processing two-dimensional tensors, such as pixel matrices of images. This enables the use of images to address structural engineering problems. Such images can efficiently train a CNN to predict results. A study using images as training data was conducted by Xue, Xiang, and Ou [1]. To predict the displacement of transmission towers under wind action, they transformed graphical images of input data (wind velocity over time) and output data (horizontal displacement of the tower top over time) into pixel matrices. As CNNs are optimized for image processing, they identify complex patterns in this manner. Simulations were performed using the Ansys software to generate outputs (displacements of the tower top) as a function of horizontal wind velocities over time. Once trained, the Convolutional Neural Network can predict the 'displacement vs. time' graph when new input data are provided. The results were compared to those obtained by Ansys and proved highly accurate.

2. Literature Review

2.1. Synthetic Wind for Computational Models

2.1.1. Mean and Fluctuating Wind Velocity

"Wind is nothing more than the movement of air masses resulting from variations in their heating by the sun. Certain regions are more intensely heated, causing the air to rise and be replaced by cooler air masses moving in at specific velocities. As these moving air masses experience friction with the Earth's surface, their velocities vary with altitude, increasing until they reach heights where they become relatively constant. (...) Wind, therefore, also possesses kinetic energy due to the mass of air set in motion" [2].

To simulate wind action in computational systems, it is necessary to consider its specific characteristics. According to Boggs, Dragovich [3], wind is a dynamic and stochastic phenomenon, both in time and space. Wind speed can be described as a mean value over which random fluctuations are superimposed. The mean wind speed increases with height above ground level, and there is no clear corre-

lation between fluctuations at different altitudes. Furthermore, these fluctuations are also random over time: waveforms do not repeat, and wind speed at any given moment can be described statistically but not predicted precisely.

According to Wahrhafti, Da Silva, and Brasil [4], wind speed is analyzed by considering two components: the mean wind speed and the fluctuating wind speed. The mean wind speed induces static loads, while the fluctuating component manages dynamic loads.

To generate artificial wind gusts that simulate natural characteristics, both the static and fluctuating components of wind loads must be included, along with an understanding of their probabilistic nature. According to Wieringa [5], the gust factor—defined as the ratio between the peak gust speed and the mean wind speed over a given interval—depends on terrain roughness, atmospheric stability, and other factors. In the most common case, under neutral conditions and average roughness, a gust factor of 1.20 is used, indicating gusts approximately 20% higher than the mean (*i.e.*, the ratio between the instantaneous peak wind speed and the mean wind speed calculated over a longer time interval). This research adopts a gust factor value of 1.20.

Another important concept is atmospheric stability, as defined by Stull [6]. Since air temperature varies with height above the ground, it also influences wind behavior with altitude and its interaction with structures. Atmospheric stability is classified into three categories: neutral (50% probability), unstable (30% probability), and stable (20% probability). These conditions decide whether the air tends to rise, descend, or remain at the same level. Unstable conditions amplify dynamic wind loads, while stable conditions reduce them. In this research, a multiplication factor of 1.50 is applied to gust wind speed under unstable conditions, and a factor of 0.70 under stable conditions.

Finally, to artificially generate wind speeds that function as loads on the structure proposed in this study, we use the stochastic nature of wind gusts. Brasseur [7] suggests that wind gusts depend on the components mentioned above (gust factor and atmospheric stability) and occur randomly, yet have a significant impact on structures. This approach is used in the research to generate not only variations in basic wind speed but also the occurrence of gusts incorporating these probabilistic factors, thereby computationally simulating real wind actions on structures.

2.1.2. Wind Power Spectrum and Dynamic Loads

The ABNT NBR 8800 [8] requires a dynamic analysis model to verify the structural behavior of buildings whose height exceeds five times the smallest horizontal dimension and is greater than fifty meters, while ABNT NBR 6123 [9] emphasizes the importance of dynamic effects in buildings with a fundamental period greater than one second. In this regard, we highlight the importance of understanding the dynamic effects of wind on structures for computational simulations. Brasil and Silva [2] note that there are currently models that adopt frequency-domain-based methods and perform this stochastic and dynamic wind analysis satisfactorily.

They cite the "synthetic wind" method proposed by Franco [10], which is used in this research and described in this chapter. This method uses the decomposition of the wind velocity power spectrum into harmonic components. The power spectrum describes how the power of a stochastic excitation, in this case, wind fluctuations, is distributed as a function of frequency. This method also defines a spatial correlation for each harmonic component, *i.e.*, how each component behaves along the structure's height, considering the gust center.

As seen in the previous chapter, wind velocity is a random variable that can be divided into mean velocity and fluctuating velocity:

$$V(t) = \overline{V} + v'(t)$$

- $\overline{V}$: mean component of the wind velocity.
- $v'(t)$: fluctuating component (turbulence).

The peak wind pressure, in N/m^2, according to ABNT NBR 6123 [9], is modelled as a function of height:

$$P(z) = 0.613 \cdot v_3^2$$

where v_3 is the peak velocity for a 3-second period at height z:

$$v_3 = V_0 \cdot S_1 \cdot b_3 \cdot F_{r,\mathrm{II}} \cdot (z/10)^{p_3} \cdot S_3$$

V_0 is the basic wind velocity; b_3, p_3, and $F_{r,\mathrm{II}}$ are meteorological parameters defined by ABNT NBR 6123, (1988) (Table 21, p. 42). For this research, with terrain in category IV, we have $b_3 = 0.86$, $p_3 = 0.12$ and $F_{r,\mathrm{II}} = 1$. The total pressure P forms the mean part $\overline{P}$ *and* the fluctuating component P'.

$$P = \overline{P} + P'$$

Also based on ABNT NBR 6123 [9], the ratio of mean wind pressure (P_{600} for 10 minutes) to peak wind pressure (P_3 for 3 seconds) is calculated, showing that the mean pressure $\overline{P}$ represents 48% of the peak pressure, and consequently, the fluctuating pressure P' represents 52% of the peak pressure.

$$\frac{P_{600}}{P_3} = \left(\frac{U_{600}}{U_3}\right)^2 = 0.69^2 = 0.48$$

The equation presented for the wind velocity power spectrum is adopted by the National Research Council Canada [11] and describes the energy distribution of wind velocity fluctuations across different frequencies:

$$\frac{nS(n)}{u_*^2} = 4 \cdot \frac{x^2}{(1+x^2)^{4/3}} \quad \text{and} \quad x = \frac{1220n}{U_0}$$

where:

- $S(n)$: spectral density of wind velocity—energy of wind fluctuations per unit frequency (m^2/s^2/Hz).
- u_* : friction velocity—depends on terrain roughness (m/s).
- U_0: mean wind velocity at 10 m height above open terrain (m/s).
- n: frequency (Hz).

To estimate the wind friction velocity (u_*), we used the formula based on the logarithmic wind velocity profile [12]:

$$u(z) = \frac{u_*}{\kappa} \ln\left(\frac{z}{z_0}\right)$$

The fluctuating pressure along the structure (standing for 52% of the peak pressure) constitutes a random Gaussian process that can be represented through a Fourier integral. Instead of an infinite number of functions, the fluctuating pressure can be approximated by a finite number m of harmonic functions chosen so that their periods uniformly cover the time interval of interest, with one of them having a period coincident with the structure's fundamental period. The periods of the other functions will be multiples or submultiples of T_R by a factor of two. We consider the approximate representation of the fluctuating pressure:

$$p'(t) \triangleq \sum_{k=1}^{m} C_k \cos\left(\frac{2\pi}{T_r r_k} t - \theta_k\right)$$

where the phase angle theta is chosen arbitrarily and used to introduce randomness in the wind simulation, and

$$C_k = \sqrt{2\int_{(k)} S(n)\mathrm{d}n}$$

is the amplitude of the *k*-th harmonic part.

$$r_k = 2^{k-r}$$

is the ratio of harmonic periods to the fundamental period.

To distribute this total fluctuating pressure among the *m* harmonic components and ensure that the sum of fluctuating pressures correctly represents 52% of the total wind pressure on the structure, the amplitudes of the harmonic components are normalized:

$$p'_k = \frac{C_k}{\sum_{k=1}^{m} C_k} \cdot p' = c_k \cdot p'$$

p'_k is the contribution of the *k*-th part of the fluctuating pressure, and c_k is the fraction of fluctuating energy attributed to part *k*. The harmonic part whose period equals the structure's resonant period (denoted by index *r*) can cause overestimation of the structural response if not controlled. To avoid this, the weight c_R is redistributed as follows:

$$c_{*r} = \frac{c_r}{2} \quad c_{*(r-1)} = c_{(r-1)} + \frac{c_r}{4} \quad c_{*(r+1)} = c_{(r+1)} + \frac{c_r}{4}$$

The pressure fluctuations at different points of the structure are not perfectly correlated. Therefore, the concept of an "equivalent gust" with finite height is used to represent the average wind effect over a certain height segment of the structure. The vertical coherence, which measures how wind fluctuations remain correlated between two points vertically separated by Delta *z* for a frequency n_k, is:

$$Coh(\Delta z, n_k) = \exp\left(-\frac{7 \cdot \Delta z \cdot n_k}{U_0}\right)$$

And the height Δz_{0k} of the equivalent gust is:

$$\Delta z_{0k} = 2\int_0^{\infty} \exp\left(-\frac{14 \cdot \Delta z \cdot n_k}{U_0}\right) \mathrm{d}(\Delta z) = \frac{U_0}{7 \cdot n_k}$$

The pressure fluctuations are thus applied with amplitudes decreasing linearly from the gust center, assuming a triangular distribution with a total height of two Δz_{0k}.

3. Objective

The objective of this research is to train machine learning algorithms, using data obtained from computational software employing FEM, to predict the maximum displacements in a reinforced concrete building during synthetic wind gusts generated by a numerical method. A regression in supervised machine learning is used, where the algorithm learns the relationship between inputs and outputs and generates a predictive model.

Initially, for the research, a reinforced concrete building model is created in the Ansys Mechanical software. This building is subjected to synthetic wind forces over time on one of its external faces. For this, the forces on each floor of the building are calculated for each wind velocity and input into the Ansys Mechanical software, which provides the responses of interest to the research: maximum horizontal displacements. With the input data (mean wind velocities) and output data (maximum displacements), the ground truth is constructed.

The ground truth data are used to train a machine learning algorithm. Through this data, the algorithm extracts patterns and "understands" the behavior of the reinforced concrete structure of this building, *i.e.*, it learns the relationship between inputs and outputs, generating a trained Machine Learning model. With the trained Machine Learning model, it is possible to predict the displacements of this building under any wind gusts with any velocities.

The response of the trained model to new gusts is immediate, *i.e.*, it does not require the processing time and cost that would be necessary if using Ansys Mechanical or any other FEM software. The trained machine learning model has thus constructed a multi-degree polynomial regression that responds accurately to any inputs and can therefore find the maximum displacements of this building under any new wind gust, at an extremely low computational cost. To verify the quality of the responses generated by the trained ML model, three different test sets with synthetic wind are used. The same test sets are input into Ansys Mechanical and the trained ML model. The results are compared and analyzed.

4. Methods

4.1. Building Structure

A reinforced concrete building model with thirty-two floors was created for research. Slabs with a thickness of 12 cm were included on all floors. The floor-to-floor height of all floors is three meters.

4.2. Modal Analysis of the Structure

Modal analysis of the structure was performed using Ansys Mechanical to obtain the natural vibration frequencies of the structure. The result obtained for the fundamental frequency was 0.24335 Hz. Thus, we obtain the fundamental period T_R of the structure: 4.109 seconds. This value is used to determine the value corresponding to the resonant harmonic function in the spectral decomposition of fluctuating pressures in the synthetic wind method, through the equality:

$$T_k = T_r \cdot r_k$$

4.3. Wind Action

4.3.1. Mean Velocities of Synthetic Wind

The synthetic wind was generated computationally using the Python programming language. A total of 300-time instants were generated, each occurring every 8 seconds, totaling 2400 seconds. At each of the three hundred instants, a mean wind velocity was generated, and thus each of the three hundred mean wind velocities remains unchanged for 8 seconds. The velocity of the first instant is defined: 32 m/s.

The gust factor is defined with a value of 1.20, and the probabilities of random occurrence of atmospheric stabilities—neutral, unstable, and stable (50%, 30%, and 20%, respectively)—are set. When unstable, the gust factor is multiplied by 1.50, and when stable, it is multiplied by 0.70.

The mean wind velocities are generated randomly, with each instant (every eight seconds) being up to 10% higher or lower than the earlier velocity. Occasional gusts occur arbitrarily with a 5% probability, affecting the variation of the mean wind velocity, and their value depends on atmospheric stability and the gust factor. When the wind velocity crosses the lower threshold of 15 m/s, there is an 80% probability of increasing. When it crosses the upper threshold of 45 m/s, there is an 80% probability of decreasing.

Four synthetic wind sets were generated: one training set with 792 seconds (99 mean velocities varying every 8 seconds) and three test sets with four hundred seconds each (50 mean velocities varying every 8 seconds), illustrated in **Figures 1-4**.

The gust parameters, including gust factor variations and atmospheric stability probabilities, were designed to produce unseen gust patterns in the test sets, differing from those in the training data. To prevent data leakage, the test sets were strictly isolated during the hyperparameter tuning process. Specifically, the grid search for Support Vector Regression (SVR) hyperparameters was conducted using cross-validation solely on the training data, and the Deep Neural Network (DNN) validation set (20% of the training data) was also derived exclusively from the training set, ensuring that no samples from the test sets influenced the model optimization.

These figures clearly show the randomness of the values with the occurrence of sporadic gusts, showing that the algorithm correctly simulates the stochastic nature of wind.

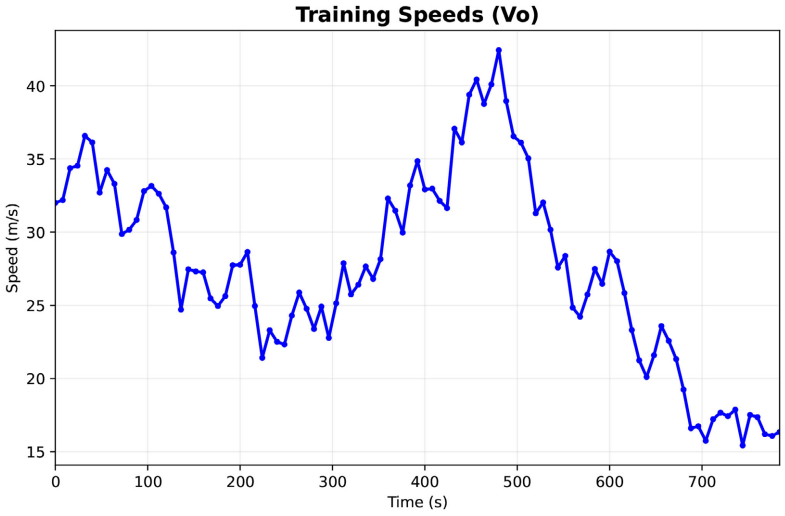

Figure 1. Training speed vs. time.

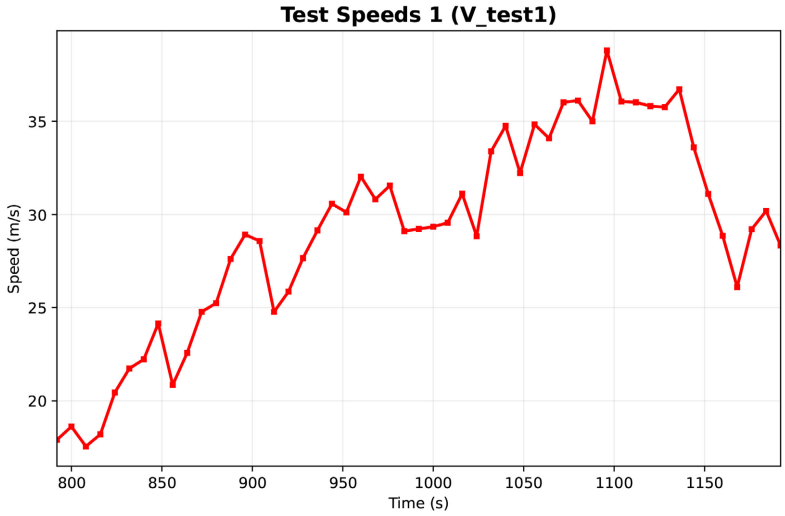

Figure 2. Test speed 1 vs. time.

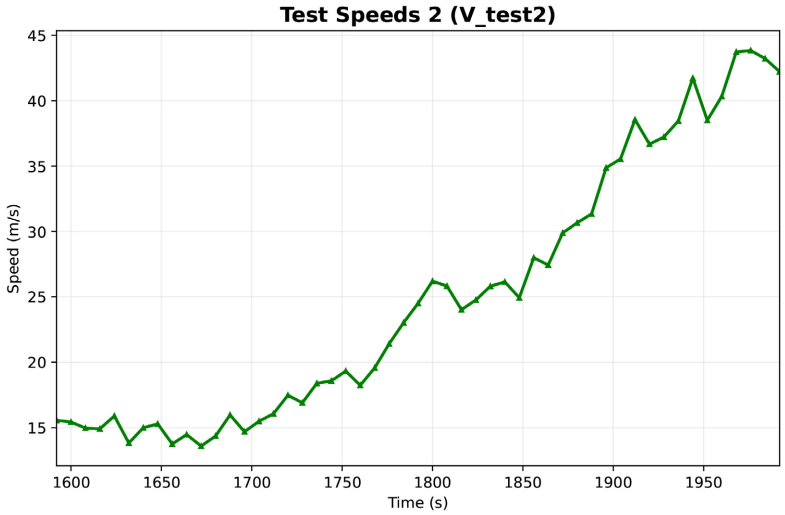

Figure 3. Test speed 2 vs. time.

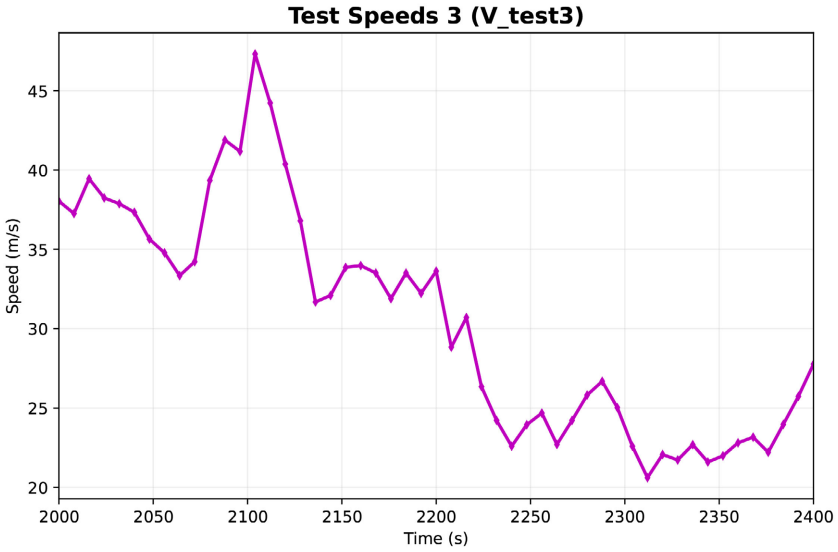

Figure 4. Test speed 3 vs. time.

4.3.2. Fluctuations of Synthetic Wind

As mentioned earlier, the mean wind velocities remain constant during the eight seconds between each value change. Throughout all synthetic wind sets, time-variable wind fluctuations occur, dependent on the mean wind velocity and height relative to the ground. The time step Δt was defined for generating the values of synthetic wind fluctuations:

$$\Delta t < \frac{T_r}{20} = 0.20 \text{ seconds}$$

A small Δt compared to the fundamental period was chosen to ensure adequate representation of higher modes [13], which are important in dynamic loadings, and to obtain a more complete structural response under wind action. At the end of the algorithm generating the synthetic wind, we have the wind pressures (mean and fluctuating) on all floors of the building every 0.2 seconds. The pressures are transformed into forces and used to find the horizontal displacements over time in Ansys Mechanical and to train and test the machine learning models. The power spectrum of the fluctuations was divided into 11 harmonic functions ($m = 11$) and the resonant index $k = 4$ and the resonant index $k = 4$ corresponds to the harmonic function whose period is resonant with the structure's fundamental mode.

$$T_4 = T_r = 4.109 \text{ s} \quad r_4 = 2^{(4-4)} = 1$$

The terrain roughness (for calculating the wind friction velocity) is 0.3 meters. The topographic factor (S1) and statistical factor (S3) have a unit value. The gust center is forty-eight meters above ground. The drag coefficient (in low turbulence) is 1.39, obtained from ABNT NBR 6123 [9]. The phase angles are random values between 0 and 2π.

Python code was created to generate the pressures and forces on the building floors as a function of time and the mean velocities of the synthetic wind, as shown in Appendix A.1. The forces are calculated for each floor using the formula:

$$F_a = C_a \cdot q \cdot A_e$$

The pressures, in kN/m², generated by the 11 harmonic functions (wind fluctuation) during the first eight seconds ($V = 32$ m/s) at height $z = 48$ meters are shown in Figure 5.

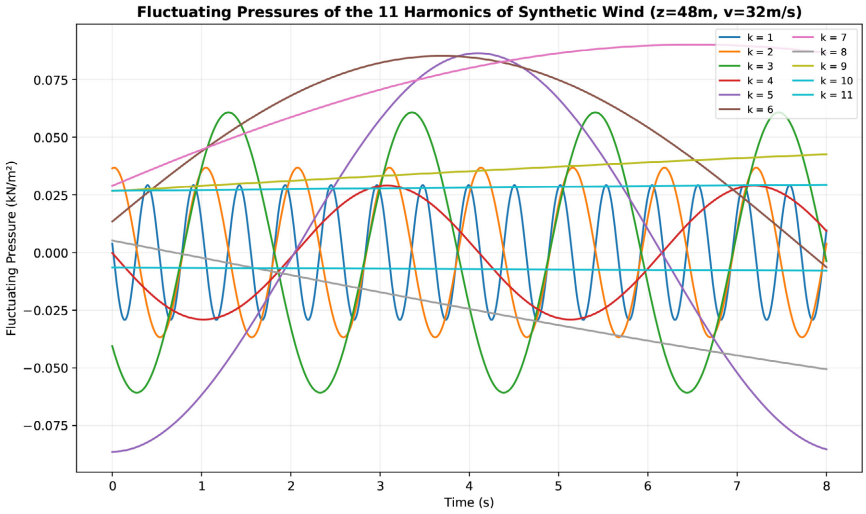

Figure 5. Floating pressure vs. time.

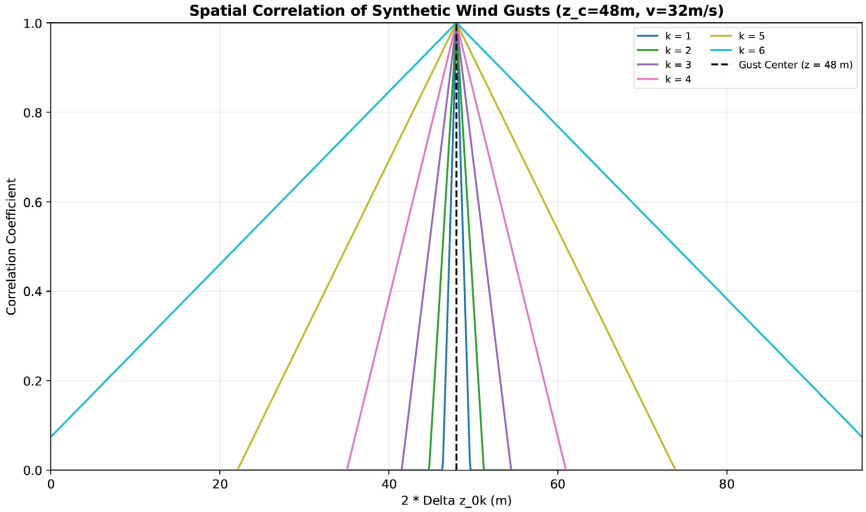

Figure 6. Spatial correlation of synthetic wind gusts.

Figure 6 shows the equivalent gust value for the first six harmonic functions for the first eight seconds ($V = 32$ m/s), and Figure 7 shows the forces as a function of time at various heights. Due to the spatial correlation factor (equivalent gust), higher frequencies are attenuated as the distance from the gust center increases. The forces, in kN, from the eleven harmonic functions added to the mean wind force, at each 0.2-second instant, are the data that feed the FEM and ML models. Figure 8 shows these forces for all building floors in the first eight seconds. As the time step is 0.2 seconds, for each mean wind value, we have forty force responses on each of the thirty-two building floors. Figure 9 shows the forces in the first 58

seconds, with seven different mean velocities.

4.4. Horizontal Displacements

4.4.1. Data Input

The structural analysis model was generated in Ansys Mechanical, where the forces were input at each instant $\Delta t = 0.2$ seconds, for all 32 floors, for the training set and the three test sets. The wind forces act on one of the wider faces of the building. The base nodes were fixed to the ground. Gravitational force was added to the model, and the calculation of second-order effects on the response was enabled.

4.4.2. Computational Model Responses

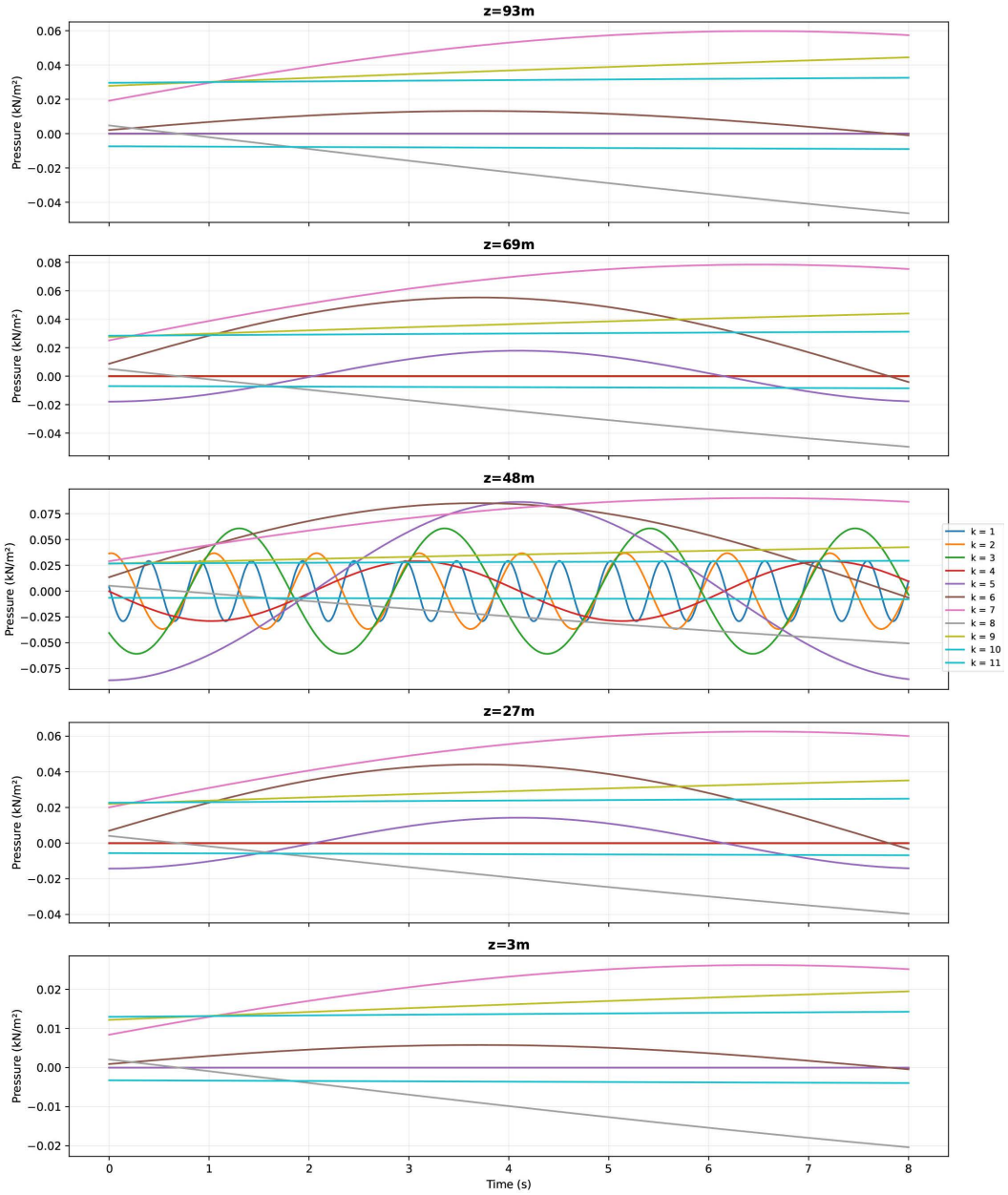

Figure 7. Pressure vs. time at different elevations.

The Ansys Mechanical software provided the responses (maximum horizontal displacements) as a function of time for the training set (Figure 10) and the three test sets. The training set took 6 hours and 29 minutes to complete the responses, and each test set took 3 hours and 16 minutes. Figure 11 shows the maximum displacements for the training set, in meters. Figures 12-14 show the maximum displacements, in meters, for the three test sets.

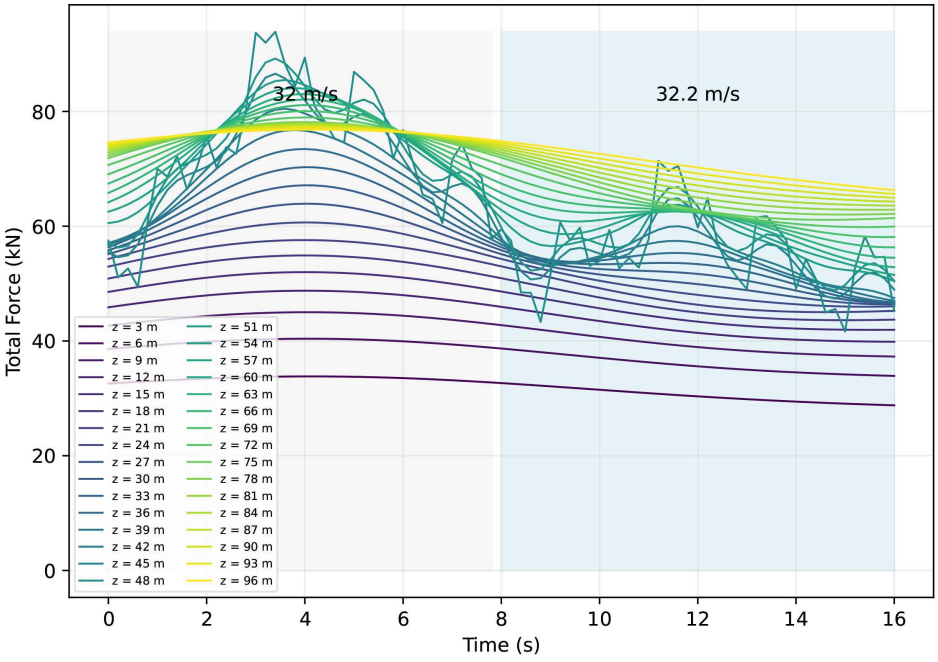

Figure 8. Total force at different elevations vs. time, under two wind speeds.

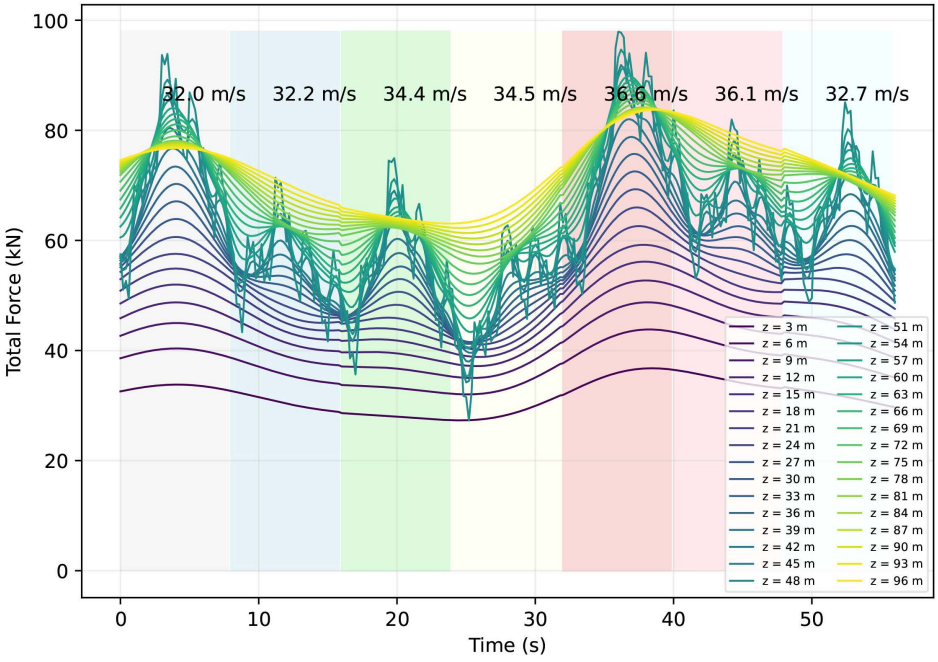

Figure 9. Total force at different elevations vs. time, under seven wind speeds.

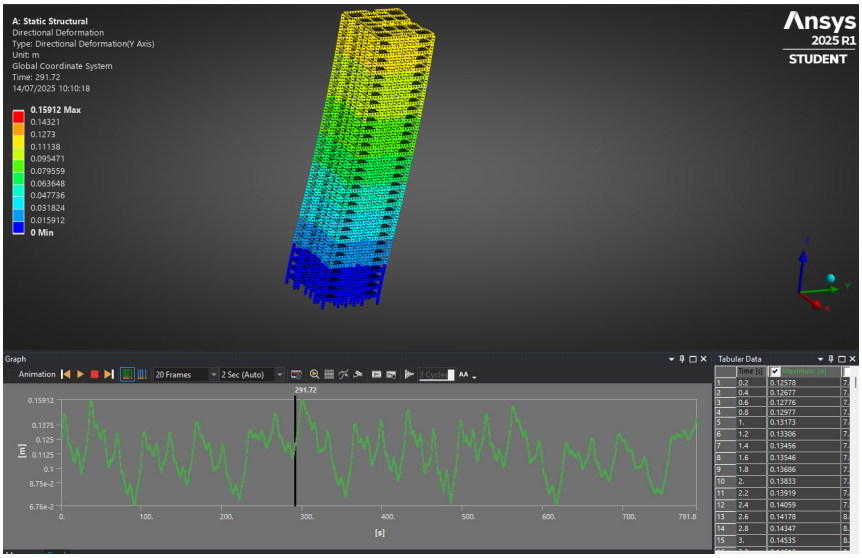

Figure 10. Maximum displacements of the training set vs. time, obtained in Ansys mechanical.

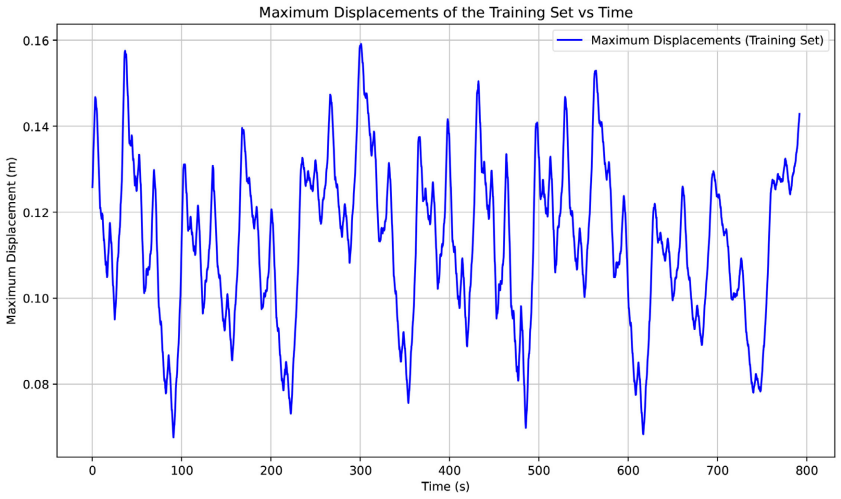

Figure 11. Maximum displacements of the training set vs. time.

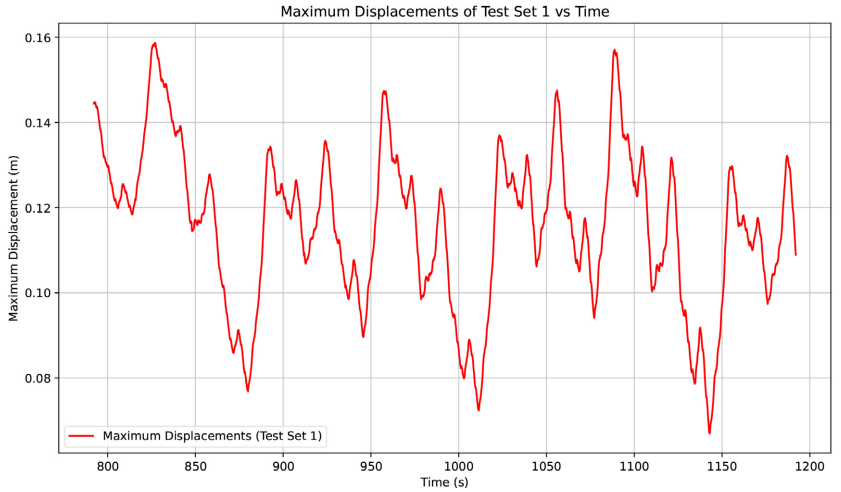

Figure 12. Maximum displacements of the test set 1 vs. time.

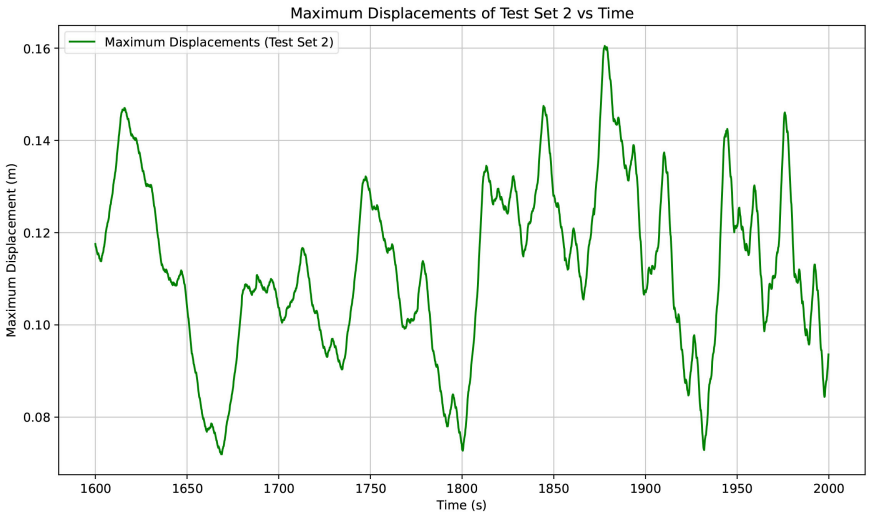

Figure 13. Maximum displacements of the test set 2 vs. time.

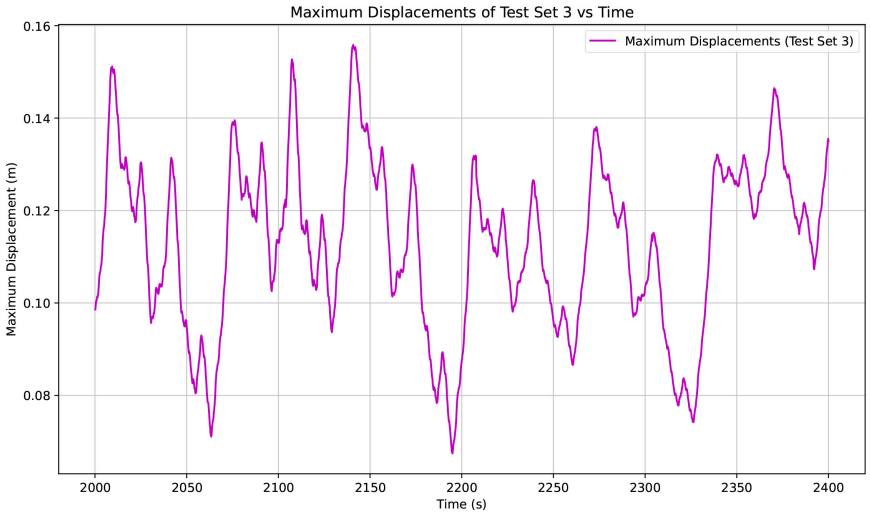

Figure 14. Maximum displacements of the test set 3 vs. time.

4.5. Data Input into the ML Algorithm

The training set generated a response (displacements in meters) for each 0.2-second time step. Thus, the response matrix has one row with 3960 values. The input matrix (forces due to wind) has thirty-two rows (one for each building floor) with 3960 values (forces in kN) per row. In other words, the input matrix has dimensions 32×3960, and the output matrix has dimensions 1×3960.

Due to the inherent damping of the structure, it is not only a single instant that influences the structure's response to loads, but a series of past instants [14]. A window of 32 wind force instants was chosen to be considered as input to the model and influence the response. Thus, the ML model inputs are matrices of dimensions 32×32, and the responses are displacements at the 32nd instant of this temporal window. There are 3929 input matrices (3960 gust instants minus the first thirty-one instants that only form the first matrix) and 3929 Ansys responses

to train the machine learning algorithm (Figure 15).

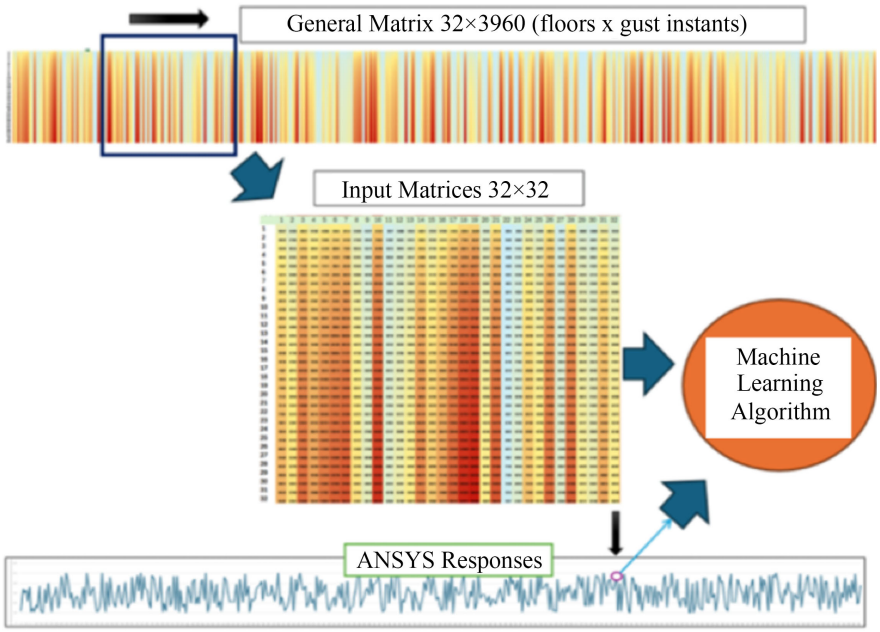

Figure 15. Overview of 32 × 32 input matrices and corresponding Ansys responses used for machine learning training.

The choice of a fixed 32 × 32 force window for the machine learning model input was based on the dynamic characteristics of the structure, particularly its dominant vibration periods and damping decay. The structure's fundamental period, calculated as 4.109 seconds from the modal analysis, corresponds to the resonant mode most influenced by wind-induced vibrations. The 32-time-step window, spanning 6.4 seconds (32 × 0.2 seconds), adequately captures the structural response over approximately 1.5 fundamental periods, ensuring that the model accounts for the dominant oscillatory behavior. Additionally, the window size aligns with the damping decay of the reinforced concrete building, which exhibits significant response attenuation within this timeframe due to inherent structural damping. This duration assures the inclusion of relevant past force inputs affecting the displacement response.

The 3959 obtained responses (equivalent to 3600 gust instants minus the first instant) are stored in a data matrix. The first instant (instant zero) has no maximum displacement response because no forces were applied to the structure before instant zero. For each input matrix of dimensions 32 × 32, the response corresponding to the last instant of this data window is always used.

4.6. Support Vector Regression Algorithm

Two machine learning algorithms are tested: Support Vector Regression (SVR) and Deep Neural Network (DNN). For SVR, the data is pre-processed for training.

Searches were conducted for the best SVR hyperparameters. The best values found are Radial Basis Function (RBF) kernel with parameter C equal to 10 and

parameter epsilon equal to 0.001. The code for the Support Vector Regression is shown in Appendix A.2.

4.7. Deep Neural Network Algorithm

For the DNN, a validation dataset separate from the training data is needed. The data are divided, in this research, into 80% training and 20% validation. Attempts were made with different deep neural networks and various hyperparameters. The code for the Deep Neural Network is shown in Appendix A.3.

The DNN that yields the best results in this research is a Densely Connected Network (DenseNet) with four stages, with the following number of interconnected dense blocks in each stage:

First stage: four dense blocks.

Second stage: four dense blocks.

Third stage: six dense blocks.

Fourth stage: five dense blocks.

A total of 19 dense blocks, each with 6 layers ($19 \times 6 = 114$ layers). There are also three transition layers (between blocks) with four layers each, totaling a total of 126 layers. A total of 2000 complete training cycles were executed.

5. Results

After training the machine learning algorithm with the set of 3960 gust instants and the corresponding 3960 responses from Ansys Mechanical, predictions are made on the three test sets (each with 2001 instants) using the machine learning algorithms. The predictions (maximum horizontal displacements) from the SVR and DNN algorithms are then compared to the responses obtained by FEM. The metrics used to evaluate the differences between the predicted and actual values are:

1) R^2.

2) mean absolute error (MAE).

3) mean squared error (MSE).

5.1. SVR Results—Test Set 1

For Test Set 1, the following evaluation results for maximum displacements were obtained:

MAE = 4.878178023086366e−05.

MSE = 5.774536491300574e−09.

R^2 = 0.9999830095453743.

Figures 16-18 illustrate the results for Test Set 1 (real values vs. predicted values over time, absolute error over time and distribution of errors).

5.2. SVR Results—Test Set 2

For Test Set 2, the following evaluation results for maximum displacements were

obtained:

MAE = 8.916634688714385e−05.

MSE = 2.4000133207421877e−08.

R^2 = 0.9999330007252047.

Figures 19-21 illustrate the results for Test Set 2 (real values vs. predicted values over time, absolute error over time and distribution of errors).

5.3. SVR Results—Test Set 3

For Test Set 3, the following evaluation results for maximum displacements were obtained:

MAE = 4.373478684629423e−05.

MSE = 5.9572713456588325e−09.

R^2 = 0.999982329333223.

Figures 22-24 illustrate the results for Test Set 3 (real values vs. predicted values over time, absolute error over time and distribution of errors).

5.4. DNN Results—Test Set 1

For Test Set 1, the following evaluation results for maximum displacements were obtained:

MAE = 0.001698095234358.

MSE = 0.000003047171220.

R^2 = 0.991034289170174.

Figures 25-27 illustrate the results for Test Set 1 (real values vs. predicted values over time, absolute error over time and distribution of errors).

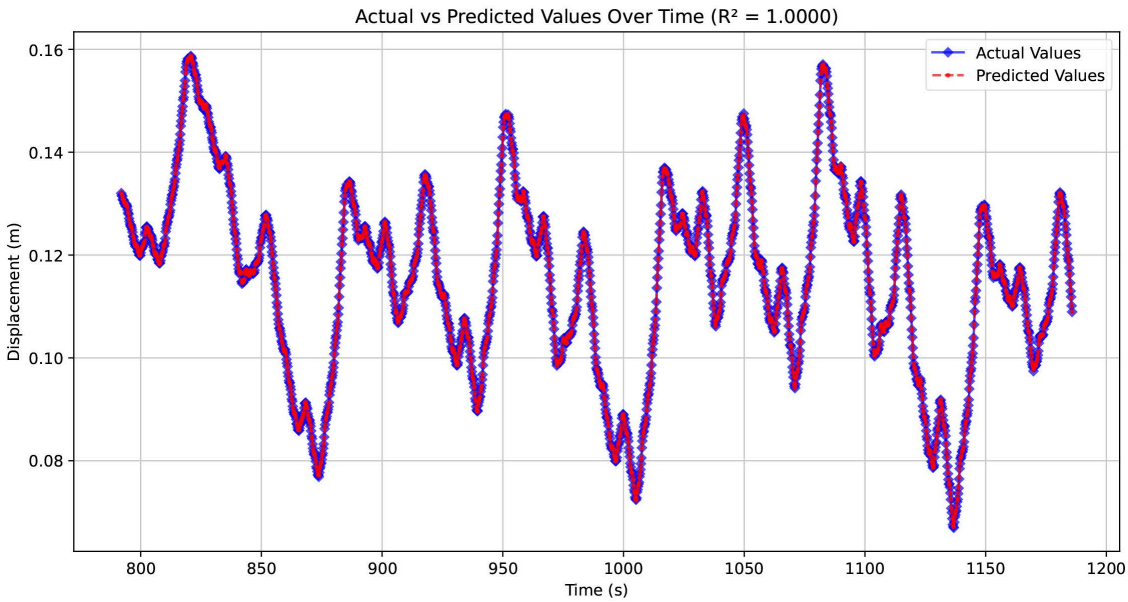

Figure 16. Actual vs. predicted values over time for test set 1 (SVR results).

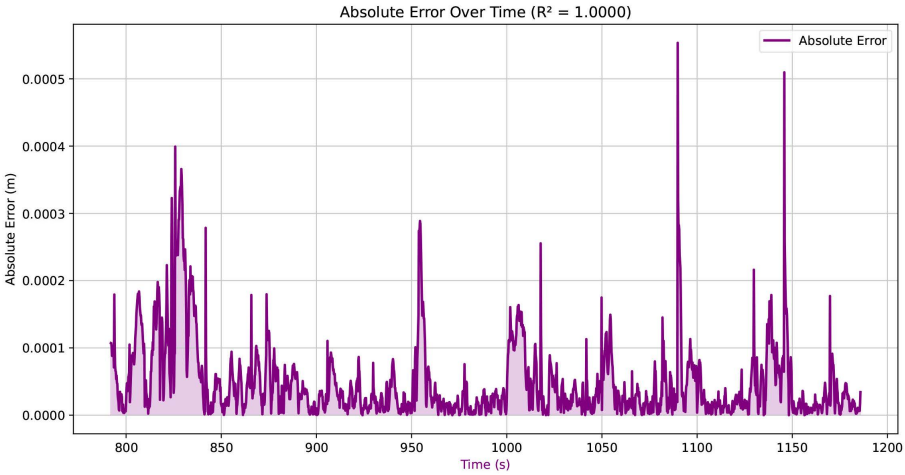

Figure 17. Absolute error over time for test set 1 (SVR results).

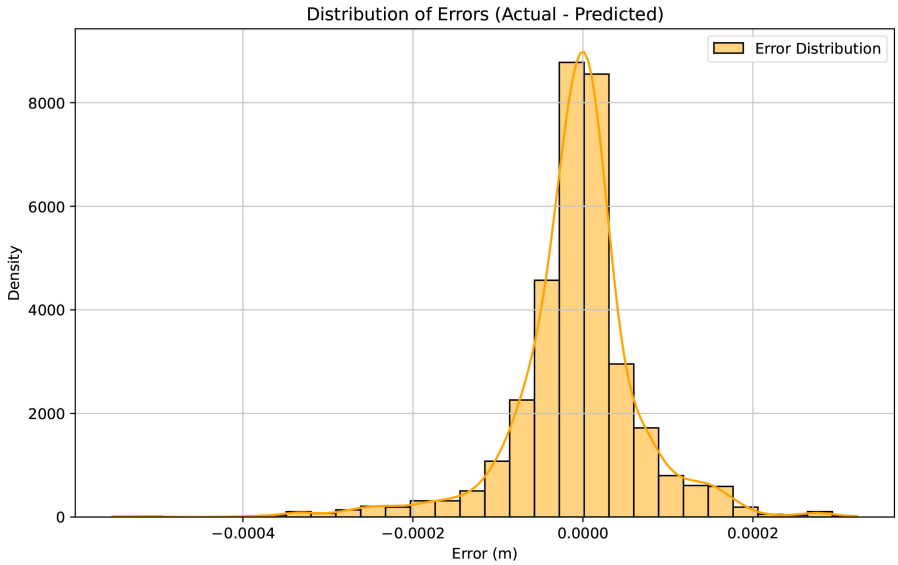

Figure 18. Distribution of errors for test set 1 (SVR results).

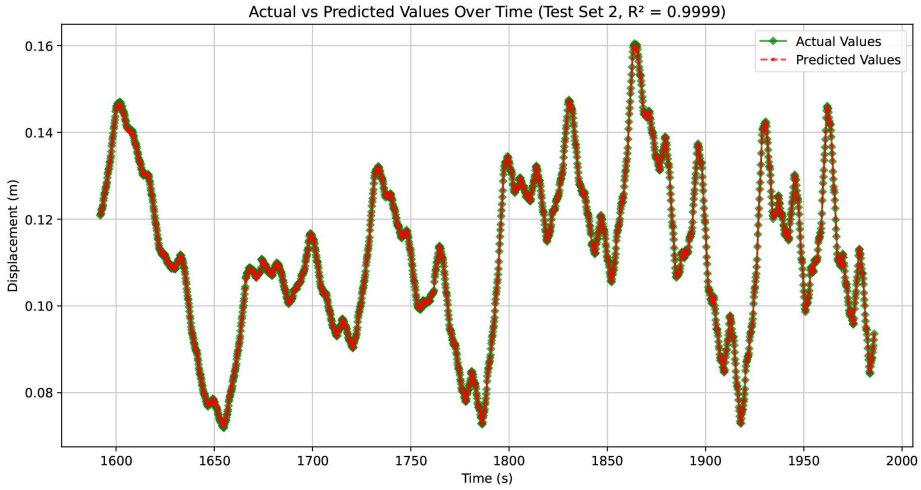

Figure 19. Actual vs. predicted values over time for test set 2 (SVR results).

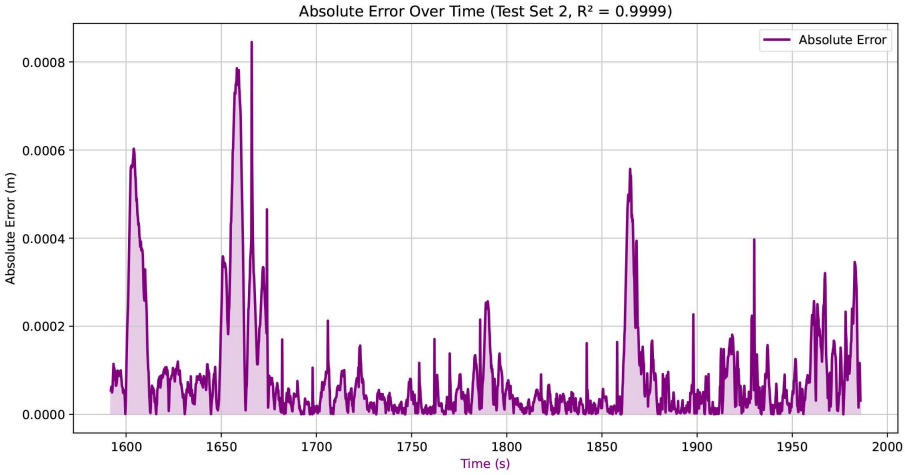

Figure 20. Absolute error over time for test set 2 (SVR results).

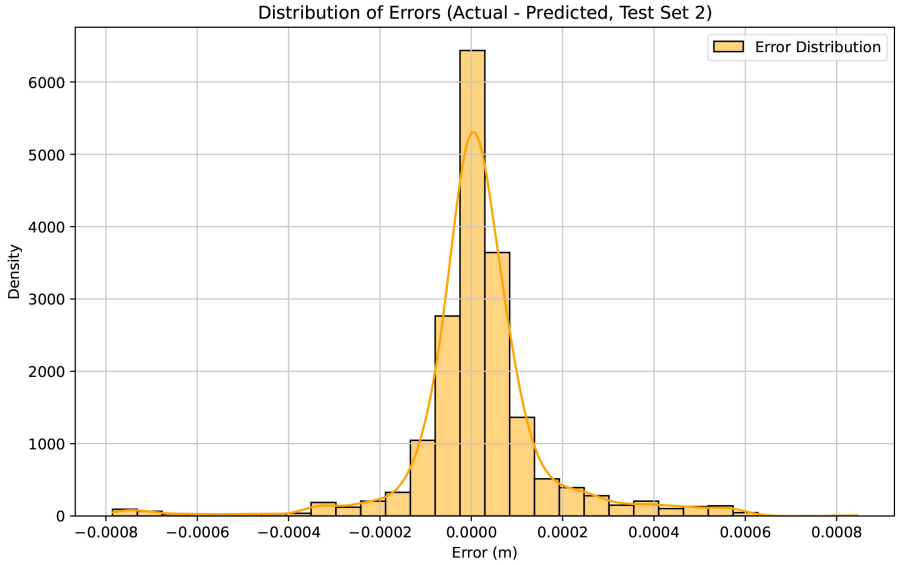

Figure 21. Distribution of errors for test set 2 (SVR results).

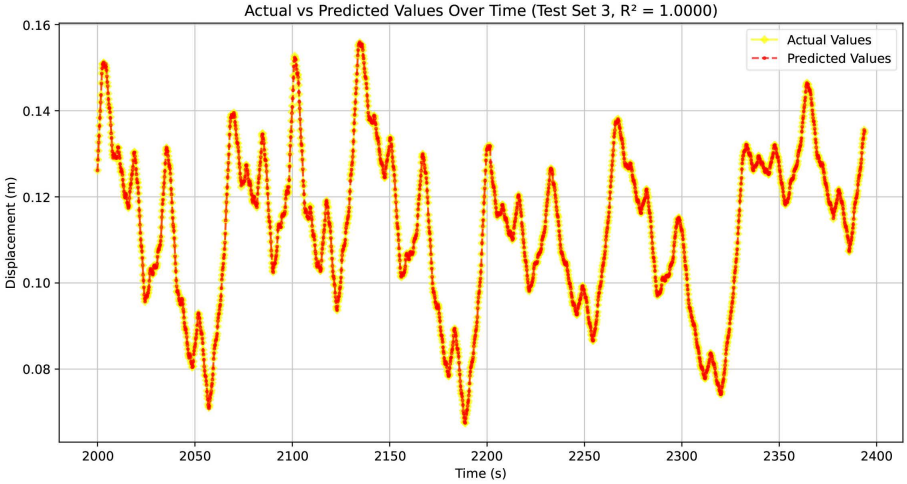

Figure 22. Actual vs. predicted values over time for test set 3 (SVR results).

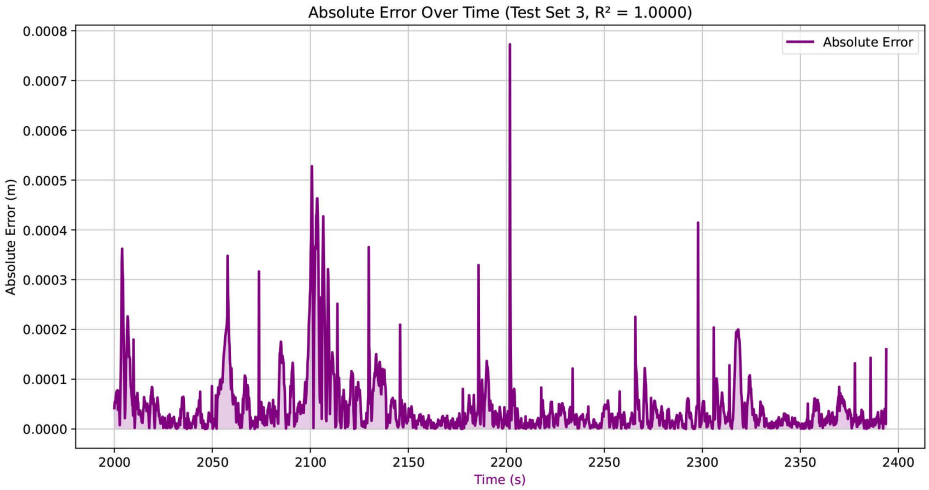

Figure 23. Absolute error over time for test set 3 (SVR results).

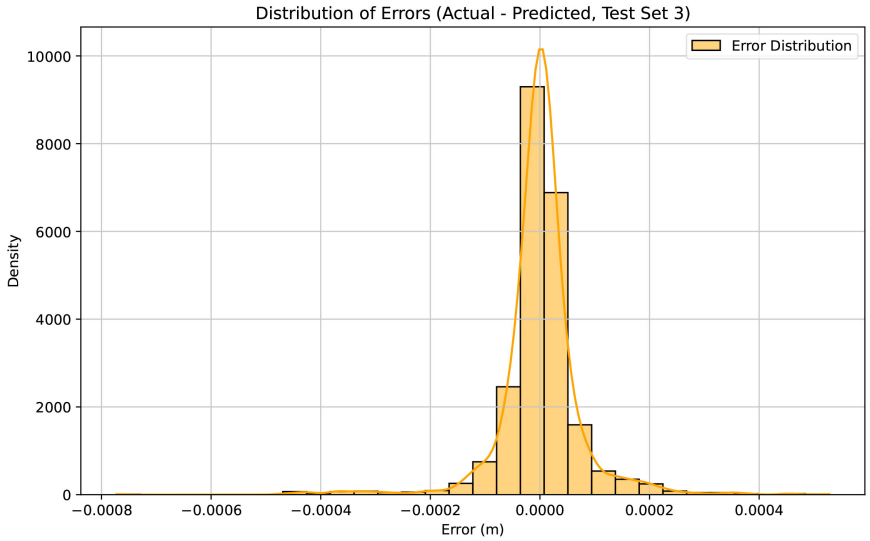

Figure 24. Distribution of errors for test set 3 (SVR results).

5.5. DNN Results—Test Set 2

For Test Set 2, the following evaluation results for maximum displacements were obtained:

MAE = 0.001698134639062.

MSE = 0.000003100036451.

R^2 = 0.991345873281185.

Figures 28-30 illustrate the results for Test Set 2 (real values vs. predicted values over time, absolute error over time and distribution of errors).

5.6. DNN Results—Test Set 3

For Test Set 3, the following evaluation results for maximum displacements were obtained:

MAE = 0.001660345922367.

MSE = 0.000002923071934.

R^2 = 0.991329481719409.

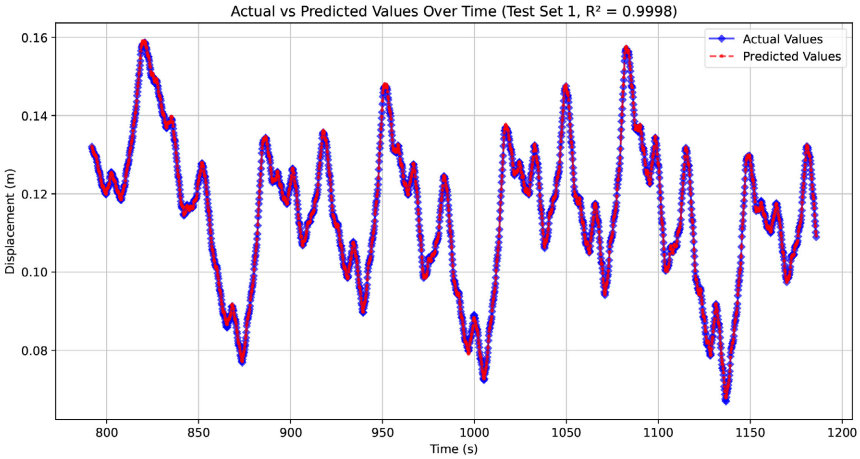

Figure 25. Actual vs. predicted values over time for test set 1 (DNN results).

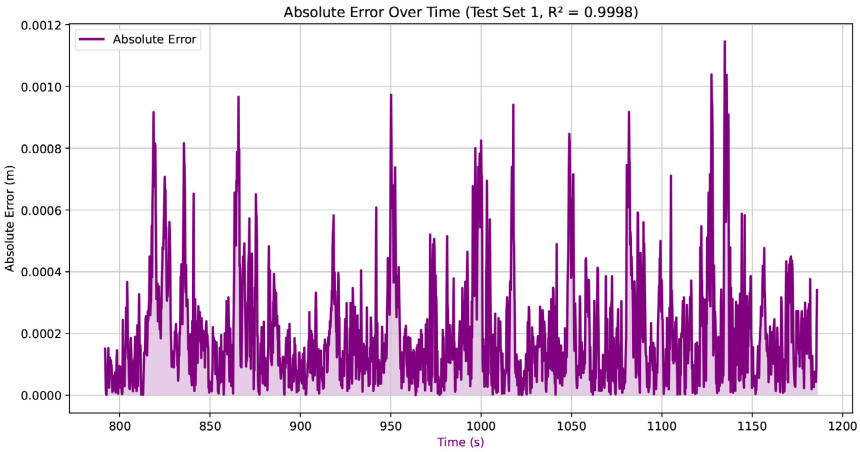

Figure 26. Absolute error over time for test set 1 (DNN results).

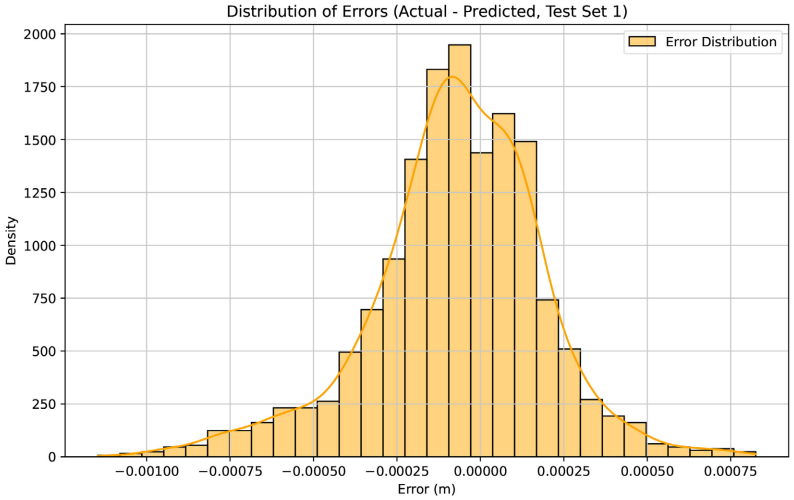

Figure 27. Distribution of errors for test set 1 (DNN results).

Figures 31-33 illustrate the results for Test Set 3 (real values vs. predicted values over time, absolute error over time and distribution of errors).

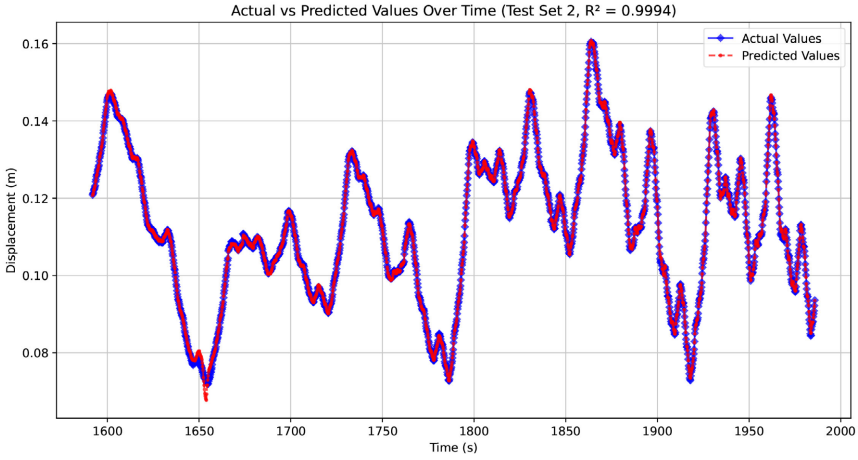

Figure 28. Actual vs. predicted values over time for test set 2 (DNN results).

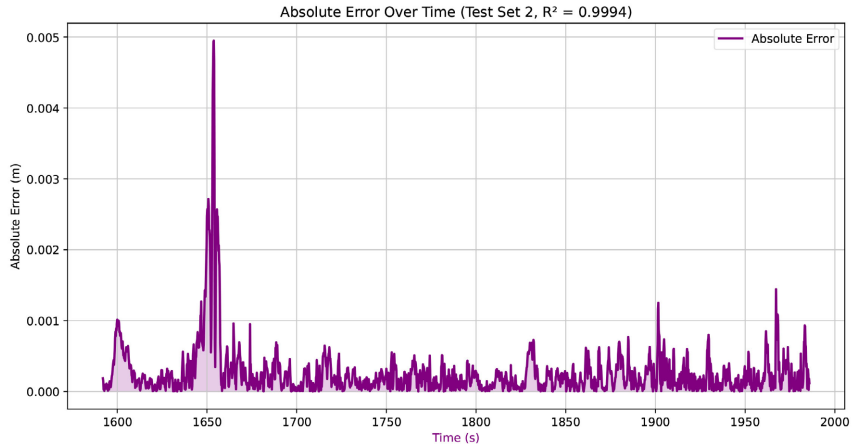

Figure 29. Absolute error over time for test set 2 (DNN results).

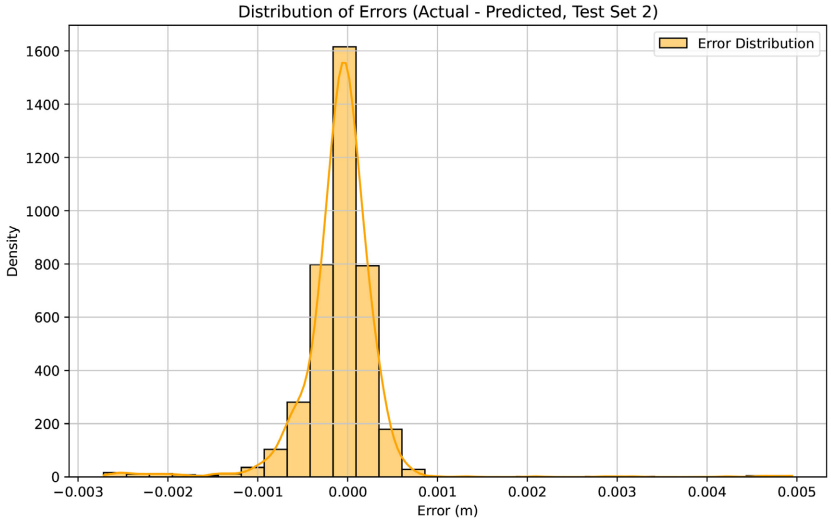

Figure 30. Distribution of errors for test set 2 (DNN results).

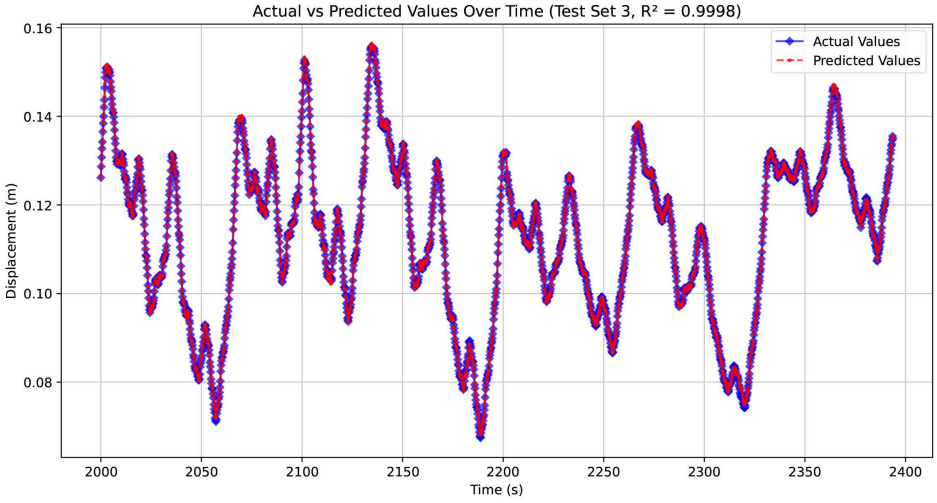

Figure 31. Actual vs. predicted values over time for test set 3 (DNN results).

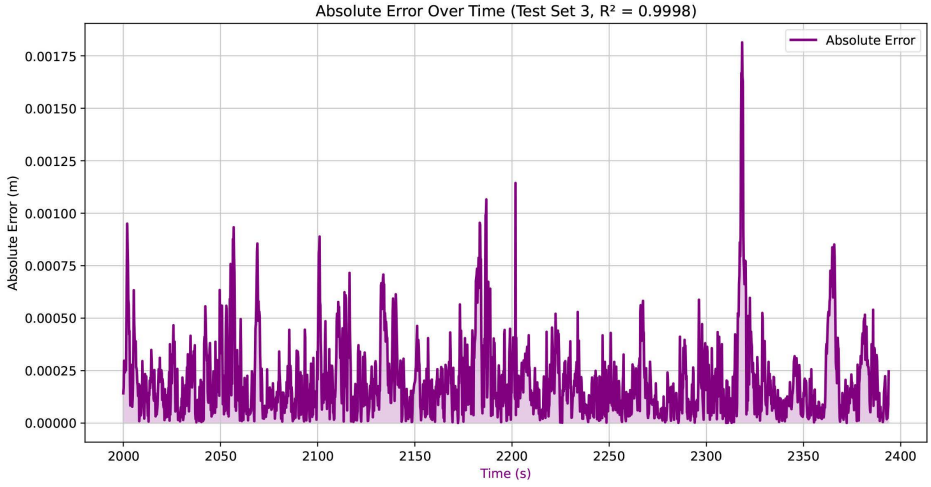

Figure 32. Absolute error over time for test set 3 (DNN results).

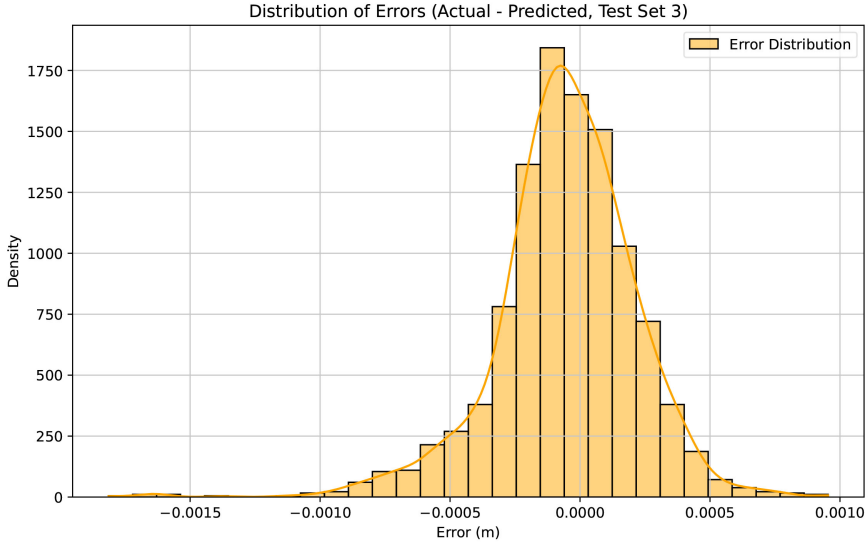

Figure 33. Distribution of errors for test set 3 (DNN results).

6. Conclusions

After comparing the predictions of the Support Vector Regression (SVR) and Deep Neural Network (DNN) algorithms with the actual results obtained by the Ansys Mechanical software across the three test sets, we can conclude:

1) Both models are highly exact, with coefficients of determination (R^2) exceeding 0.998 for SVR and close to 0.991 for DNN. SVR showed slightly superior performance, suggesting that simpler models may be more effective for the specific problem of predicting displacements under synthetic wind loads. However, DNN may be helpful in scenarios with greater data complexity.

2) The computational efficiency of the machine learning algorithms was significantly higher than that of Ansys Mechanical, with test set processing times of six milliseconds (DNN) compared to 3 hours and 16 minutes (Ansys), highlighting a much lower computational cost.

3) The results underscore the potential of artificial intelligence methods for analyzing complex structural engineering problems, such as responses to dynamic wind actions, and pave the way for other engineering issues, such as the analysis of seismic actions.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Xue, J., Xiang, Z. and Ou, G. (2021) Predicting Single Freestanding Transmission Tower Time History Response during Complex Wind Input through a Convolutional Neural Network Based Surrogate Model. *Engineering Structures*, **233**, Article 111859. https://doi.org/10.1016/j.engstruct.2021.111859

[2] Brasil, R.M.L.R.F. and de Silva, M.A. (2015) Introdução à dinâmica das estruturas para a engenharia civil. 2nd Edition. Editora Blucher.

[3] Boggs, D. and Dragovich, J. (2006) The Nature of Wind Loads and Dynamic Response. *International Concrete Abstracts Portal*, **240**, 15-44. https://www.concrete.org/publications/internationalconcreteabstractsportal/m/details/id/18290

[4] Wahrhaftig, A.D.M., de Silva, M.A. and Brasil, R.M.L.R.F. (2019) Analytical Determination of the Vibration Frequencies and Buckling Loads of Slender Reinforced Concrete Towers. *Latin American Journal of Solids and Structures*, **16**, 1-31. https://doi.org/10.1590/1679-78255374

[5] Wieringa, J. (1986) Roughness-Dependent Geographical Interpolation of Surface Wind Speed Averages. *Quarterly Journal of the Royal Meteorological Society*, 112, 867-889. https://doi.org/10.1256/smsqj.47315

[6] Stull, R.B. (1988) An Introduction to Boundary Layer Meteorology. The University of British Columbia. https://doi.org/10.1007/978-94-009-3027-8

[7] Brasseur, O. (2001) Development and Application of a Physical Approach to Estimating Wind Gusts. *Monthly Weather Review*, **129**, 5-25. https://doi.org/10.1175/1520-0493(2001)129<0005:daaoap>2.0.co;2

[8] ABNT Nbr 8800 (2008) Projeto de estruturas de aço e de estruturas mistas de aço e

concreto de edifícios. Associação Brasileira de Normas Técnicas.

[9] ABNT NBR 6123 (1988) Forças devidas ao vento em edificações. ABNT—Associação Brasileira de Normas Técnicas.

[10] Franco, M. (1993) Direct Along-Wind Dynamic Analysis of Tall Structures. Technical Bulletin BT/PEF/9303, Department of Structural and Foundation Engineering, Polytechnic School of the University of São Paulo (EPUSP), São Paulo, Brazil. 30 p.

[11] National Research Council Canada (2020) National Building Code of Canada.

[12] Kaimal, J.C. and Finnigan, J.J. (1994) Atmospheric Boundary Layer Flows: Their Structure and Measurement. Oxford University Press. https://doi.org/10.1093/oso/9780195062397.001.0001

[13] Clough, R.W. and Penzien, J. (1995) Dynamics of Structures. 3rd Edition, Computers & Structures, Inc.

[14] Brito, C.M.S.R.D. (2018) Modelagem e simulação numérica de treliças espaciais submetidas a carregamentos estáticos e dinâmicos em regime elástico não linear. Universidade Federal da Bahia. https://repositorio.ufba.br/handle/ri/27163

Appendix A: Program Codes

A.1. Code to Generate the Pressures and Forces on the Building Floors

```
def F_vel(v, z, t):
    # Parameters adjusted with V_0 = v
    V_0 = max(v, 1e−6)  # Avoid V_0 = 0
    U_0 = 0.69 * V_0  # Mean U (m/s)
    kappa = 0.4  # Von Kármán constant
    z_r = 10.0  # Reference height in meters
    z_0 = 0.3  # Roughness length in meters
    u_fric = (kappa * U_0)/math.log(z_r/z_0)  # Friction velocity (m/s)
    T_r = 1/0.24335  # Fundamental period (s)
    m = 11  # Number of harmonics
    k_res = 4  # Resonant index
    Z_c = 48  # Gust center (m)
    C_a = 1.39  # Aerodynamic coefficient
    A_e = 3 * 24.61  # Effective area (m²)
    # Generate phase angles phi_k
    np.random.seed(10)  # Set seed for reproducibility
    phi_k = np.random.uniform(0, 2 * np.pi, m)  # 11 angles between 0 and 2 * pi
    # Step 1: Calculate k, r_k, T_k, and n_k
    k = np.arange(1, m + 1)  # Indices from 1 to 11
    r_k = 2.0 ** (k − k_res)  # Scale factor: r_k = 2^(k − 4)
    T_k = T_r * r_k  # Periods: T_k = T_r * r_k
    n_k_values = 1/T_k  # Frequencies n_k
    # Step 2: Calculate c_k and c*_k
    def S_n(n):
        if abs(n) < 1e−10:
            return 0.0
        x = (1220 * n)/max(U_0, 1e−6)
        return (4 * u_fric**2/max(n, 1e−10)) * (x**2/(1 + x**2) ** (4/3))
    C_k_values = []
    for i in range(m):
        if i == 0:
            n_lower = max(n_k_values[1], 1e−10)
            n_upper = 1/(T_k[0]/2)
        elif i == m − 1:
            n_lower = n_k_values[i]/2
            n_upper = n_k_values[i−1]
        else:
            n_lower = max(n_k_values[i+1], 1e−10)
            n_upper = n_k_values[i−1]
```

```
            try:
                integral, _ = quad(S_n, n_lower, n_upper, epsabs=1e−8, epsrel=1e−8)
                C_k = np.sqrt(2 * max(integral, 0))
                C_k_values.append(C_k)
            except:
                C_k_values.append(0.0)
        C_k_sum = sum(C_k_values)
        if C_k_sum < 1e−10:
            C_k_sum = 1e−10
        C_k_values = [C_k/C_k_sum for C_k in C_k_values]
        C_ast_values = C_k_values.copy()
        C_ast_values[2] = C_k_values[2] + C_k_values[3]/4
        C_ast_values[3] = C_k_values[3]/2
        C_ast_values[4] = C_k_values[4] + C_k_values[3]/4
        # Step 3: Define delta_zok
        delta_zok_values = []
        for i in range(len(k)):
            delta_zok = U_0/(7 * max(n_k_values[i], 1e−10))
            delta_zok_values.append(delta_zok)
        # Step 4: Define pressure functions dependent on z
        def p_pico(z):
            return (V_0 ** 2) * (0.453/1000) * (z/10) ** 0.24
        def p_m(z):
            return 0.48 * p_pico(z)
        def p_f(z):
            return 0.52 * p_pico(z)
        # Step 5: Define p_flut(z, t)
        def p_flut(z, t):
            p_flut_value = 0
            for i in range(m):
                p_harm = C_ast_values[i] * p_f(z) * np.cos((2 * np.pi * t/T_k[i]) - phi_k[i])
                p_flut_value += p_harm
            return p_flut_value
        # Step 6: Use c*_k and include spatial modulation
        def p_esp(z, t):
            soma = 0.0
            for i in range(m):
                delta_zok = max(delta_zok_values[i], 1e−10)
                fator_espacial = max(1 − (abs(z − Z_c)/delta_zok), 0)
                termo = C_ast_values[i] * p_f(z) * np.cos((2 * np.pi * t/T_k[i]) - phi_k[i]) *
fator_espacial
                soma += termo
            return soma
```

```
    # Step 7: Force functions
    def F_med(z):
        return C_a * p_m(z) * A_e
    def F_flut(z, t):
        return C_a * p_esp(z, t) * A_e
    def F_tot(z, t):
        result = F_med(z) + F_flut(z, t)
        if not np.isfinite(result):
            return 0.0
        return result
    return F_tot(z, t)
    def F_flut(z, t):
        return C_a * p_esp(z, t) * A_e
    def F_tot(z, t):
        result = F_med(z) + F_flut(z, t)
        if not np.isfinite(result):
            return 0.0
        return result
    return F_tot(z, t)
```

A.2. Support Vector Regression Code

```
# Define the SVR model
svr = SVR(kernel = 'rbf')
# Define hyperparameters for grid search
parametros_busca = {
    'C': [0.1, 1, 10, 100],
    'epsilon': [0.001, 0.01, 0.1],
    'gamma': ['scale', 'auto', 0.001, 0.01, 0.1]
}
# Configure grid search with cross-validation
busca_grade = GridSearchCV(
    svr,
    parametros_busca,
    cv = 5,    # 5-fold cross-validation
    scoring = 'neg_mean_squared_error',
    n_jobs = −1,    # Use all available cores
    verbose = 1
)
# Train the model with grid search
busca_grade.fit(X_treino_escalado, y_treino_escalado)
```

A.3. Deep Neural Network Code

```
def dense_block(x, blocks, growth_rate):
```

```
    for i in range(blocks):
        x = conv_block(x, growth_rate)
    return x
def conv_block(x, growth_rate):
    x1 = BatchNormalization()(x)
    x1 = Activation('relu')(x1)
    x1 = Activation('relu')(x1)
    x1 = Conv2D(growth_rate*4, (1, 1), padding='same')(x1)
    x1 = BatchNormalization()(x1)
    x1 = Activation('relu')(x1)
    x1 = Conv2D(growth_rate, (3, 3), padding='same')(x1)
    x = Concatenate(axis=−1)([x, x1])
    return x
def transition_block(x, reduction):
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = Conv2D(int(x.shape[−1] * reduction), (1, 1), padding='same')(x)
    x = MaxPooling2D((2, 2), strides=(2, 2))(x)
    return x
def build_regression_densenet(input_shape, blocks, growth_rate, reduction):
    inputs = Input(shape=input_shape)
    x = Conv2D(64, (7, 7), strides=(2, 2), padding='same')(inputs)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
    # Build dense blocks and transition blocks
    for i, num_blocks in enumerate(blocks):
        x = dense_block(x, num_blocks, growth_rate)
        if i != len(blocks) − 1:
            x = transition_block(x, reduction)
    x = GlobalAveragePooling2D()(x)
    outputs = Dense(1, activation='linear')(x)    # Regression output layer
    model = Model(inputs=inputs, outputs=outputs, name='densenet_regression')
    return model
# Define input shape
input_shape = (32, w, 1)
# Define DenseNet parameters
blocks = [4, 4, 6, 5]    # Number of dense blocks in each stage
growth_rate = 32
reduction = 0.5
# Build the regression DenseNet model
model = build_regression_densenet(input_shape, blocks, growth_rate, reduction)
```