Computational Mechanics
https://doi.org/10.1007/s00466-023-02386-9

ORIGINAL PAPER

A general framework of high-performance machine learning algorithms: application in structural mechanics

George Markou[1] · Nikolaos P. Bakas[2,3] · Savvas A. Chatzichristofis[4] · Manolis Papadrakakis[5]

Received: 19 June 2023 / Accepted: 21 August 2023

Abstract
Data-driven models utilizing powerful artificial intelligence (AI) algorithms have been implemented over the past two decades in different fields of simulation-based engineering science. Most numerical procedures involve processing data sets developed from physical or numerical experiments to create closed-form formulae to predict the corresponding systems' mechanical response. Efficient AI methodologies that will allow the development and use of accurate predictive models for solving computational intensive engineering problems remain an open issue. In this research work, high-performance machine learning (ML) algorithms are proposed for modeling structural mechanics-related problems, which are implemented in parallel and distributed computing environments to address extremely computationally demanding problems. Four machine learning algorithms are proposed in this work and their performance is investigated in three different structural engineering problems. According to the parametric investigation of the prediction accuracy, the extreme gradient boosting with extended hyper-parameter optimization (XGBoost-HYT-CV) was found to be more efficient regarding the generalization errors deriving a 4.54% residual error for all test cases considered. Furthermore, a comprehensive statistical analysis of the residual errors and a sensitivity analysis of the predictors concerning the target variable are reported. Overall, the proposed models were found to outperform the existing ML methods, where in one case the residual error was decreased by 3-fold. Furthermore, the proposed algorithms demonstrated the generic characteristic of the proposed ML framework for structural mechanics problems.

Keywords Machine learning · Deep learning artificial neural networks · Parallel training · Finite element method · Structural mechanics

1 Introduction

Artificial intelligence (AI) techniques have emerged over the last decades as an effective and efficient tool to predict analysis outputs for computationally demanding engineering problems. Application areas requiring multiple algorithmic operations (nonlinear dynamics analysis, design optimization, structural reliability, stochastic simulations) have benefited from AI computational approaches eliminating the need for performing full-scale numerical analyses by providing adequate estimations for the outputs of interest [1–5]. Furthermore, significant work was performed on machine learning (ML) in computational science and engineering [6–10], while the use of ML algorithms in handling constitutive material modeling is also an emerging field [11–15].

Furthermore, work related to the prediction of the fundamental period of infilled frame structures can be found in [16], where an artificial bee colony-based neural network was proposed. Another research work that studied the development of predictive models for computing the fundamental period of frames is [17], where ML algorithms and nonlinear models were used to investigate the structural problem. Finally, different ML methods are explored in [18], where the

✉ George Markou
george.markou@up.ac.za

1 Civil Engineering Department, University of Pretoria, Hatfield Campus, Pretoria 0028, South Africa

2 National Infrastructures for Research and Technology – GRNET, 7 Kifisias Avenue, Athens 11523, Greece

3 School of Liberal Arts and Sciences, Technology & AI Lab, The American College of Greece, Deree, Athens, Greece

4 Intelligent Systems Lab and Department of Computer Science, Neapolis University Pafos, 2 Danais Avenue, Pafos 8042, Cyprus

5 Department of Civil Engineering, National Technical University of Athens, 9, Iroon Polytechniou str, Athens 15780, Zografou, Greece

Published online: 09 January 2024

prediction of cement-based mortar's compressive strength was explored.

Nevertheless, despite the fast inference time of a trained model, the AI training process is a demanding task in terms of computational resources, which makes necessary the use of parallel processing. Artificial neural networks (ANNs) consist of interconnected nodes that process information and transmit them through the nodes in the network. The number of nodes and the depth of the network determines its ability to learn complex patterns and relationships. However, training ANNs can be tedious and time-consuming. The model must undergo multiple analyses to adjust its weights and biases to predict outputs for a given set of input parameters accurately. ANNs may comprise millions of parameters to train via iterative procedures, such as the stochastic gradient descent algorithm. The structure of the data these algorithms yield require special handling in a parallel computing environment, mainly when parallel processing is performed with graphics processing units (GPUs), where the data structures cannot be accommodated in the GPU's RAM.

Furthermore, the distributed optimization algorithms deployed during a deep network training process can be parallelized by the following two alternative routes. The first one is data parallelism [19–22], which foresees the splitting of the batch of samples (utilized in each iteration) into a number of smaller mini-batches, which are processed in parallel, depending on the number of available resources (GPUs). Alternatively, the model parallelism [23] may be used by partitioning the tasks of the deep learning model on distributed GPUs. Despite the fast-pacing growth of deep learning and the vast progress in computational resources, relatively limited engineering applications are reported in the literature related to parallel processing during training [24–28]. This illustrates the need to research further the potential of ML techniques for addressing complicated engineering problems.

The primary task of the approach that will be used for the needs of this research work will foresee the generation of data sets with the use of the 3D detailed modeling of reinforced concrete (RC) structures as, described in [29, 30] and for steel structures, as presented in [31]. All finite element analyses are performed using the Reconan FEA [32], a research software code for performing nonlinear and modal analyses, to generate the data sets for the needs of this work. The software code has been extensively validated through the use of experimental results found in the literature, where its ability to predict the nonlinear response of structural members and structures is demonstrated.

This paper presents a generic, high-performance computational framework for developing ML models trained on data sets stemming from structural mechanics problems. Section 2 presents the proposed ML algorithms and a novel procedure for feature selection. In this section, the ML models' hyper-parameter tuning and statistical reliability are discussed. Furthermore, the vital statistical inference techniques regarding the analysis of errors, the sensitivity of the target variable concerning features, and the adequacy of the training data set are presented. In Sect. 3 the training of the proposed deep neural network in a distributed computing cluster comprising sixty-four Nvidia Tesla V100 40GB GPUs is performed. The deep-learning results serve as a benchmark for evaluating existing ML results in the literature. Subsequently, ML algorithms are applied and compared on a testbed of three structural engineering data sets associated with RC slender beams discussed in Sect. 4.1. RC buildings in Sect. 4.2, and steel structures in Sect. 4.3. Finally, the main numerical findings are outlined in Sect. 5.

2 Machine learning

The use of ML algorithms has increased exponentially in recent years in a variety of scientific fields, from computer vision [33, 34], to real estate evaluations [35], including simulation-based engineering science [36–38]. Their predictive abilities are confirmed in many applications, making them a very promising tool for solving computationally demanding problems. Different methods can be found in the scientific literature and, more recently in industry, where they are utilized to provide solutions that cannot be obtained through conventional numerical approaches. Methods like random forest [39], gradient boosting [40], and ANNs [38] are some of the widely used ML numerical tools. However, these models are usually complex enough that they cannot be efficiently implemented and validated in real-world applications. In order to address this deficiency, improved ML techniques have been developed recently [41, 42].

It is important to note at this point that the linear regression (LR) method [43] is used here as the base comparison ML tool, while three of the four proposed high-performance ML models that are implemented in this research work are based on the following ML algorithms:

1. The polynomial regression (PR) [44, 45], is very useful for automatically extracting closed-form prediction formulae. PR models describe the relationship between independent and dependent variables with a polynomial expression. It provides a more flexible model than LR, which can be used to fit non-linear data. The polynomial degree can be increased to capture complex relationships between the variables. However, this can lead to overfitting, where the model fits the noise in the data rather than the underlying connection.
2. The extreme gradient boosting (XGBoost) [40, 46] with hyper-parameter tuning (grid search) and cross-validation. XGBoost is a gradient-boosting library for

ML problems such as classifying and regressing. It implements the gradient boosting framework designed to be fast and scalable, making it suitable for use with large data sets and high-dimensional data. XGBoost is used extensively in industrial applications due to its strong performance on various data types and its ability to handle missing values and unbalanced classes. In addition, XGBoost avoids overfitting, which is a common problem in ML, with built-in feature selection and regularization.

3. The random forest (RF) [39, 47] is a popular ML algorithm for classification and regression problems. It creates a large number of decision trees and combines their outputs to make a prediction. RF is an ensemble method that combines multiple decision trees to make a prediction. Each tree is trained on a random sample of the data, and the final prediction is made by aggregating the results of all the trees. This approach reduces overfitting and increases the robustness of the model.

2.1 Polynomial regressions with hyper parameter tuning (POLYREG-HYT)

Polynomial regression is applied to develop predictive models in higher-order classes. The model relies on nonlinear combinations of all independent variables, a procedure performed up to any arbitrarily high degree. The proposed POLYREG-HYT model automatically selects the nonlinear features that correspond to the minimum prediction error. The number of these combinations decrease quickly in terms of the number of features and polynomial degree. For n number of features and k polynomial degrees the combination number c is expressed in terms of n and k by the following equation:

$$c = \left(\binom{n}{k}\right) = \binom{n+k-1}{k} = \frac{(n+k-1)!}{k!(n-1)!}, \tag{1}$$

For $n = 10$ and $k = 5$, the possible combinations are 2002, while for $n = 20$ and $k = 10$, the number of combinations becomes 2×10^7. Henceforth, since data sets are limited, an appropriate choice among those nonlinear features is based according to the degree of impact on enhancing the algorithmic performance. Thus, for the case of a problem with s number of samples, the maximum number of features $m < s$ is taken $m = \frac{s}{10}$ or $m = \frac{s}{20}$, to have enough degrees of freedom to regress and obtain statistically reliable results. Therefore, the number of nonlinear features combinations is given as:

$$c_f = \binom{c}{m} = \frac{(c)!}{k!(m-k)!}, \tag{2}$$

resulting in more than 2×10^{17} combinations for the case where $n = 10$, $k = 5$, and $m = 10$, while for the case of $n = 20$, $k = 10$, and $m = 10$, the total combinations become excessively high, making its implementation impossible. This is a computationally intensive task, given that a structural analysis problem has to be solved for each combination. An improved version is proposed to optimize this procedure, as presented in the following Section.

2.1.1 Feature selection algorithm

The POLYREG-HYT feature selection algorithm is described in Algorithm 1. The proposed algorithm has as its main task to create a closed-form prediction formula, which is written in Julia Language [48] that has been found to be computationally efficient when handling large numerical problems. Thereafter, the code was also developed by using Python [49]. Through the link found in [49], all data sets used herein and the Python algorithm incorporating the proposed ML methods are provided as open access files and can be directly downloaded. Instructions on how to install the NBML software are also provided. It is important to note that the proposed algorithm is generic and, at the same time, transparent to the user, given that it returns an explicit, closed-form formula based on the computed predictions. Algorithm 1 is an improvement to the embedded optimization problem of the polynomial feature selection of existing PR algorithms [50, 51], as will be shown in the numerical tests section.

Algorithm 1: Polynomial Feature Selection Algorithm of POLYREG-HYT

Data: $\mathbf{X}, \mathbf{y}, m_f$ (maximum number of features)
Result: Initialize $[o] = 1$ with the constant term $\in [p]$
Solve Linear System $\mathbf{X}' \times \mathbf{a} = \mathbf{y}$, where $\mathbf{X}' \subset \mathbf{X}$, with $[o]$ columns.
Compute regression errors e_1.
Set as optimal error $\hat{e} \leftarrow e_1$.
Set as optimal indices $[\hat{o}] \leftarrow [o]$.
for $i \in [1, 2, \ldots, l]$ **do**
  repeat
    Select an index $d \in [p]$ randomly.
    if $d \in [o]$ **then**
      $r \leftarrow \mathcal{U}(0, 1)$
      if $r < \frac{1}{2}$ **then**
        Select randomly $o_d \in [p] : o_d \notin [o]$
        $[o] \leftarrow ([o] \setminus d) \cup o_d$;
      else
        $[o] \leftarrow [o] \setminus d$;
      end
    else
      if $o < m_f$ **then**
        $[o] \leftarrow [o] \cup d$;
      else
        Select randomly $o_d \in [o]$
        $[o] \leftarrow ([o] \setminus o_d) \cup d$;
      end
    end
  until $rank(\mathbf{X}') \equiv o$;
  Solve Linear System $\mathbf{X}' \times \mathbf{a} = \mathbf{y}$.
  Compute regression error e_i.
  if $e_i < \hat{e}$ **then**
    $\hat{e} \leftarrow e_i$
    $[\hat{o}] \leftarrow [o]$
  else
    $[o] \leftarrow [\hat{o}]$
  end
end

In Algorithm 1, where p is the total number of polynomial features, m_f represents the maximum number of formula features. The proposed algorithm aims to identify the indices $[o] = \{o_1, o_2, \ldots, o\} \subset [p] = \{1, 2, \ldots, p\}$ for minimizing the regression error e_i during the loop over iteration i. It must be noted here that $\mathbf{X}$ is the full input matrix consisting the values found within the data set, and $\mathbf{X}'$ is the corresponding input matrix with $[o]$ columns only. This algorithm is a modified version of the one proposed in [50, 51] for a combinatorial optimization case, using cross-validation and producing stable yet highly accurate results. Moreover, as the numerical test section demonstrated, the proposed ML algorithms achieved minimal errors during the training and testing stages for developing closed-form formulae for different structural mechanics problems.

At this point, it is important to note that all proposed ML and AI algorithms are presented in an open-source software, and through a user's manual found in **nbml** [49].

2.2 Hyperparameter tuning of XGBoost random forests (XGBoost-HYT-CV and RF-HYT)

For improving the XGBoost algorithm [40, 46, 52], hyperparameter tuning and cross-validation is proposed. This ML framework will be referred to as XGBoost-HYT-CV. This tuning was found to exhibit high accuracy for all data sets that were investigated and presented in Sects. 4.1, 4.2, and 4.3. The numerical results obtained with deep learning in Sect. 3 are similar to the ones derived by XGBoost for the same data set (Sect. 4.1 for slender beams), while XGBoost-HYT-CV was found to be less computationally demanding.

Specifically, a grid search with cross-validation is proposed in this paper, depending on the data set's underlying noise. These optimised recommended training parameters resulted after an extensive parametric investigation. According to the problem, the number of folds and the training-

validation split percentage is selected to attain similar behavior of the training and validation sets. The tuning is performed for the following significant training parameters, affecting the accuracy and generalization capabilities of the proposed algorithm:

1. Maximum Number of XGBoost Rounds $\in [10, 20, \ldots, 1000]$,
2. Maximum tree depth $\in [1, 7, 15]$,
3. Learning Rate Eta $\in [0.05, 0.2, 0.5]$,
4. Colsample_bytree $\in [0.5, 1]$,
5. Subsample $\in [0.5, 1]$.

If the XGBoost-HYT-CV model is trained for 1000 rounds, then the prediction can be performed for the corresponding rounds $\in [100, 200, \ldots, 900]$. Hence, to obtain results from all combinations of parameters, there is the need to train 3 (maximum tree depth), times 3 (eta), times 2 (colsample_bytree), and times 2 (subsample), resulting in 36 training to obtain 3600 models. However, because each fold comprises a different sub-set of the training set, these 36 models must be trained for each fold $\in [f] = [1, 2, \ldots, f]$ separately, resulting in $36 \times f$ training processes. Therefore, using $f = 5$ folds, a total of 180 XGBoost-HYT-CV models must be trained. This training is usually fast and can be performed on a standard PC, while XGBoost-HYT-CV is inherently parallelizable, making the process computationally affordable.

For the case of the proposed random forests with hyperparameter tuning (RF-HYT), and for the same case study considered previously, it is proposed to use [39, 47, 53–55]:

1. Number of Trees $\in [10, 50, 100]$
2. Subfeatures' Percentage $\in [0.25, 0.5, 0.75]$
3. Partial Sampling $\in [0.25, 0.5, 0.75]$
4. Maximum Depth $\in [1, 5, 10]$
5. Minimum Leaf Samples $\in [1, 5, 10]$

The number of trees used in the forest is a critical parameter that determines the accuracy and stability of the model. The percentage of subfeatures determines the number of variables randomly sampled as candidates for each split. Partial sampling is used to select a subset of the training data to train each tree, which helps to reduce overfitting. The maximum depth of the trees sets the limit on how deep each tree can grow and is used to control the complexity of the model. Finally, the minimum number of samples, required to be at a leaf node, determines the size of the terminal nodes in the trees.

2.3 Cleaning of the data set

Another critical issue that needs to be addressed is the process of data set cleaning. This is an important part of any ML training algorithm, which is performed in this study as follows:

1. Locate the missing values in the training and testing sets, and replace them (e.g., with the mean of the particular feature in the train set).
2. Identical rows in the training set, in terms of features and target, are identified for retaining only the unique ones.

This feature was included in the proposed algorithm which can be found in [49], for the preparation of the data sets before training and testing.

2.3.1 Multicolinearity

Even when the data set satisfies missing values and identical rows, it is important to check for co-linear features, as they are redundant and may disorientate the training procedure and the predictions. This is achieved by: Extracting multi-colinear columns; Computing the rank k, of the training matrix; Applying QR-Factorization, and selecting the first pivoted k columns; and Updating train and test sets with the new ones with non-colinear columns.

The additional computational demand that is introduced to the system attributed to the QR-factorization prior to performing training and testing, is trivial when compared to the thousands of training cycles that are performed during the model development. Furthermore, the QR-factorization decreases the required number of training cycles leading to a reduced overall computational demand for the extraction of the predictive models.

2.4 Statistical reliability of model selection

A trained model might exhibit high accuracy, even for a group of folds. However, there is no guarantee that this accuracy is not happening by chance. To overcome this potential risk, the reliability of the optimization process for both XGBoost-HYT-CV and POLYREG-HYT is examined and improved when necessary. For this purpose, the training-validation-test curves are obtained, corresponding to the performance of each one of the models, sorted by the validation performance. Subsequently, the empirical cumulative distribution function (CDF) for each set (Train-Validation-Test) is derived. If the CDF is similar to a uniform or normal distribution, then this is a good indicator that the training process is convergent and the model's accuracy is reliable.

2.5 Error analysis

One of the most critical issues, which has not been adequately addressed in previous studies, is the error analysis

of the results. In this work, the difference between the prediction and target variable for each model is computed for the Training and Testing Sets, and an error analysis is performed, comprising the:

1. The residual errors vs. target diagrams, and
2. The probability density functions and cumulative density functions of the errors,

This numerical procedure will identify specific patterns occurring in the prediction results that will impair both the generalization capability and reliability of the model.

2.6 Sensitivity analysis

A sensitivity analysis is needed to evaluate and assess each feature's impact within the derived predictive model. Therefore, as soon as the trained models are constructed, all features at some specific values are kept constant (i.e., 25%, Median, and 75% quantiles), allowing the permutation of only one feature around its given values. These values are set in training mode, and the corresponding sensitivity curves for each feature and each of the trained models are obtained. Thus, the corresponding impact of each one of the features on the target variable can be identified.

2.7 Impact of data-set volume

Data-centric AI is an emerging trend in the community working with ML algorithms [56]. In this context, iterative training is performed for all methods and for a partial random subset of the train set, starting from 25% up to 100% of the observations, with 20 intermediate new training iterations. By implementing this numerical process, the adequacy of the data can be assessed by evaluating the shape of the curve depicting the performance of each individual training. Thus, if the curve is stabilized, this indicates that the data within the training set was sufficient. Otherwise, more training samples are needed to achieve a data set that will ensure the derivation of an accurate predictive model.

The final fourth proposed high-performance ML algorithm that will be numerically evaluated in this research work is based on a deep learning framework through the use of ANNs, and is described in the next section. It is important to note here that the developed Python code has the ability to be used on Windows and Linux operating systems, where the analysis can be performed in a parallel and serial manner. More details can be found in the manual found in [49].

3 Parallel training of deep learning ANNs on GPUs

PyTorch (high-performance deep learning library [57]) was used for the needs of this research work due to a straightforward, yet efficient implementation of an automatic differentiation algorithm [58]. The Horovod library [59], developed at Uber, was implemented for multi-GPU training. By using Horovod, one may take a single-GPU training script and efficiently scale it to run across many GPUs in parallel. Through the use of message passing interface (MPI) commands [60], each process is initialized and assigned its MPI rank in a straightforward manner, which is achieved with fewer code changes compared to other approaches. Horovod scripts can run on a single GPU, multiple GPUs, or even multiple hosts without further code changes [49]. Algorithms on various experiments were tested for the needs of this research work on the Cyclone Supercomputer,[1] utilizing PyTorch for computer vision as well as regression tasks,[2] highlighting the efficiency of data parallelism, as well as the scaling-up capabilities compared with standard ML platforms, such as Kaggle and Google Colab.

3.1 Parallel stochastic gradient descent

In the stochastic gradient descent (SGD) method, a random data point i is selected at each iteration, and the loss function for this data point is differentiated. Then the weights for the entire network are updated. In the mini-batch SGD, i is the current "batch" of data, a subset of all data set indices. The parallelization is performed at this point by updating the weights for all batches in parallel. Hence, in the parallel stochastic gradient descent (P-SGD) algorithm, a random data point is selected for each GPU, for which the gradient is computed, and the weights are mixed. The weights can be mixed by averaging or using other methods, such as ensembles and AdaSum. Subsequently, the update of the weights for all GPUs takes place. To perform this operation in parallel, the utilization of MPI[3] tool is required for gathering the results among all GPUs, then reducing, and once more broadcasting them across all available GPUs. This P-SGD algorithmic procedure implemented here is described in Fig. 1.

In general, when using larger batch sizes, the procedure becomes faster, whereas by adopting smaller batch sizes the process becomes more accurate but computationally demanding. In any case, the batch must fit into the relevant hardware memory, so the utilization of a larger number of nodes is required when a GPU cannot handle the batch size.

[1] https://hpcf.cyi.ac.cy/

[2] https://github.com/CaSToRC-CyI/artificial-intelligence-hpc.

[3] https://mpitutorial.com/tutorials/mpi-reduce-and-allreduce/.

Fig. 1 The parallel stochastic gradient descent (P-SGD) algorithmic procedure

$\forall$ epoch n, do:

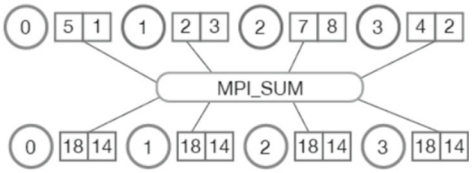
$\forall$ batch i, do in parallel:

$\forall$ GPU g, do:

$$\Delta L_i = \frac{\partial L_i}{\partial w}$$

$$\Delta \mathrm{L} = \frac{1}{b}\sum_{i=1:b} \Delta L_i \quad \Rightarrow \quad MPI\ allreduce = \mathrm{MPI\_Reduce} + \mathrm{MPI\_Bcast}$$

$$w_{n+1} = w_n - \gamma \Delta \mathrm{L}$$



Furthermore, the weights mix causes some accuracy losses in practice. However, all these aspects regarding the computational behavior of the P-SGD procedure are occasionally valid, as the loss function is a highly non-linear function of the ANN's weights and depends on the data set's particular features. It is, therefore, difficult to predict the performance of the algorithm in terms of accuracy and computational efficiency. Thus, the training of the network has to be performed with a variety of architectures. To effectively address this issue, hyperparameter tuning is adopted in the present work, leading to extensive network training, which was successively performed on the Cyclone supercomputer.

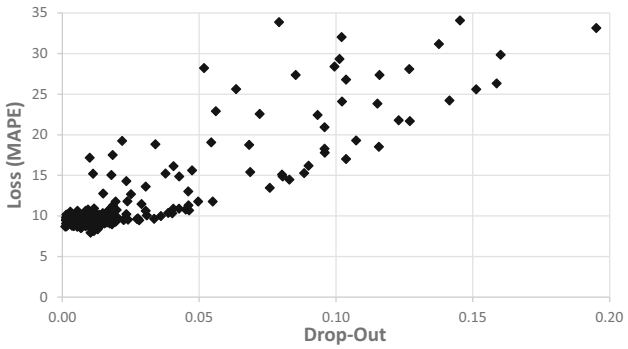


Fig. 2 Ray-tune results

3.2 Hyperparameter tuning of deep neural networks

Aiming to investigate the effect of the network's hyperparameters, the Ray-Tune[4] module of the PyTorch framework was employed. First, a number of ANNs are constructed by varying the batch size, the drop-out ratio, the number of epochs, the neurons, and the learning rate. Subsequently, the training of the networks is conducted by recording the training time for each batch of the training set. At the same time, the final loss function and the ratio among the validation and train loss *ratio V-T* are computed.

Tuning is also time-consuming, therefore, it was found useful to identify an optimal drop-out region of ratios [61], as depicted in Fig. 2, which is observed at the upper and right section of the diagram. It can be concluded that, for lower drop-out values, the corresponding values of the loss function are also lower, while for the reasons explained below, the drop-out could not be zero.

[4] https://pytorch.org/tutorials/beginner/hyperparameter_tuning_tutorial.html.

3.3 Combination of MPI and Horovod-based DANN-MPIH-HYT

During an ANN's training phase, the loss function is automatically recorded for updating the weights. However, the mean squared error (MSE) does not offer meaningful information in engineering applications, and the mean absolute percentage error (MAPE) is adopted as a more practical metric. Accordingly, as the optimum ANN architecture is not known in advance, multiple experiments must be performed, and recording the MAPE during training in real-time is necessary. However, as the data is split and reside on many GPUs, a gather operator should be implemented on the individual predictions to obtain an aggregated metric for the data set. In addition, a variety of metrics can be applied in evaluating the model's performance, such as the a20 and a10 index proposed in [16].

For this reason, a combination of Horovod with MPI is proposed to perform this numerical procedure computationally. In Fig. 3, three code snippets are depicted describing the implementation of such an operation. The overall process starts with the initialization of vector *pred* in the code at the upper right part of Fig. 3, which is used later as a container of all results. Next, the code imports the MPI module at the upper left part from mpi4py and gets the rank of each

```
from mpi4py import MPI
comm = MPI.COMM_WORD
rank = comm.Get_rank ()
torch.manual_seed (0)
```

(a) Initialize MPI an get processor's ranks

```
for e in tqdm(range(1, EPOCHS+1)):
    if rank = 0:
        t1 = time ()
        pred = np.empty (0)
        predV = np.empty (0)
    else:
        pred = None
        predV = None
```

(b) For each epoch, initialize and empty vector (pred for train and predV for validation) to add later (in c) the predictions.

```
optimizer.step ()
comm.Barrier ()
pred_i = copy(y_train_pred [:, 0])
pred_i = comm.gather(pred_i, root=0])
comm.Barrier ()
if rank = 0:
    for ii in range(len(pred_i )):
        pred = np.concatenate ((pred, pred_i[ii].detach().cpu().numpy()), axis=0)
comm.Barrier ()
```

(c) Place a barrier after each optimizer's step to collect the predictions for each GPU. Then concatenate all predictions. Then we can compute the MAPE and the loss function during training.

Fig. 3 (DANN-MPIH-HYT): a combination of MPI and Horovod

GPU. Then, as shown in the code at the bottom-left part of Fig. 3, after each step of the optimizer, a barrier is set to wait for all GPUs before finishing its task. Then, the response *y_train_pred* of each GPU is stored in vector *pred_i* and, with the *comm.gather* command, all results are placed in *pred_i*. Finally, an additional barrier is introduced to wait for all GPUs to finish and gather the results in rank o vector *pred*. At this stage, vector *pred* holds the concatenated predictions and can be utilized for calculating the MAPE.

The proposed parallel deep learning ANNs algorithm with hyperparameter tuning, MPI and Horovod will be referred to as DANN-MPIH-HYT. Given that the data sets of structural mechanics problems dealing with the shear capacity of structural members or the fundamental period of structure, their data set size is relatively small. For this reason, it was decided to use a data set of 21397 images where each one had a size of 5.76 Mb deriving a data set with a total size of 123 Gb. The numerical problem foresaw the optimization of 19 million weights and the analysis was performed on Tesla GPUs.

In Fig. 4, the parallel scalability capabilities of the proposed algorithm DANN-MPIH-HYT is presented through the relevant speed-up, when used for this image recognition problem. According to Fig. 4, the number of GPUs utilized to train the model is ranging from 1-64 and the speed-up is obtained as the ratio between the time needed to train on one GPU over the time on 2, 4, 8, 16, 32, 64 GPUs. It can be seen that an almost linear pattern occurs, highlighting the

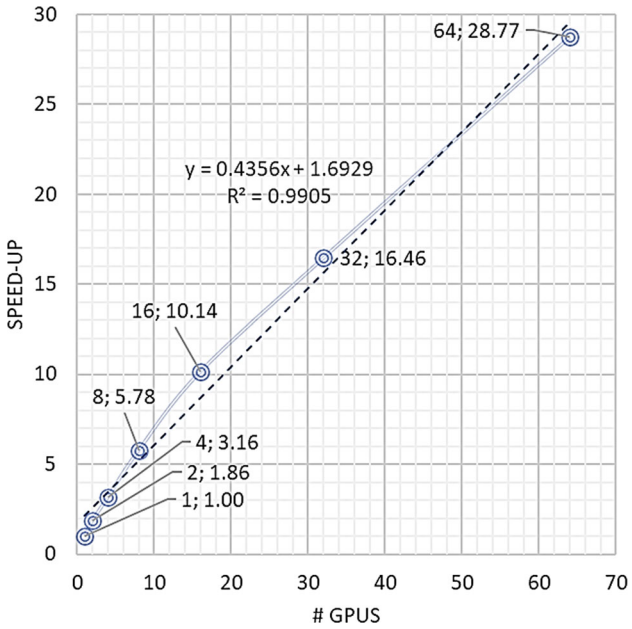


Fig. 4 Speed-up from one to sixty-four GPUs, using MPI & Horovod

algorithm's parallel efficiency in avoiding any overheads in the communication among the multiple GPUs utilized. Also, according to the parallel analysis it was found that the total ML training speed-up achieved when 64 Tesla GPUs are used was 28.77.

4 Numerical tests

In this section, three different structural mechanics problems are selected to illustrate the numerical performance of the proposed algorithms, POLYREG-HYT, XGBoost-HYT-CV, RF-HYT, and DANN-MPIH-HYT. The structural behavior of the selected test cases has different characteristics in an effort to examine the reliability and robustness of the proposed ML algorithms. More precisely, one of them refers to the structural response and limit capacity of reinforced concrete (RC) structural members, while the other two refer to the structural behavior of full-scale structures, including the soil-structure interaction (SSI) phenomenon. The main objective of the numerical experiments is to demonstrate the general nature of the proposed algorithms and their ability to reach an optimum predictive model regardless the nature and characteristics of the considered data sets. It is important to note here that the three data sets used for the needs of this research work are generated with Reconan FEA software code and consist of a relatively large number of data points when compared to data sets that foresee the use of results obtained through physical experimental tests.

The use of the specific numerically generated data sets does not imply that the proposed methods cannot be implemented on other data sets. For instance, an image recognition data set was used to generate Fig. 4, as discussed above, which had nothing to do with structural mechanics. Additionally, a research work that was performed in parallel with the research presented in this manuscript, foresaw the implementation of the proposed POLYREG-HYT on a data set consisting of experimental data of RC column shear capacities. The results were presented at the COMPDYN 2023 conference [62] and showed that the proposed ML algorithm managed to develop a predictive model that outperforms any design equation and predictive formula found in the international literature when computing the shear capacity of RC columns. It is also important to note that the proposed XGBoost-HYT-CV was used to train and test on the data set found in [63], where it was found that the proposed ML algorithm derived the best error metrics currently found in the international literature. Furthermore, the four proposed ML methods were parametrically investigated on the data set published in [31] which deals with curved steel I-beams and their respective deflection. It was found that the proposed methods managed to improve the predictive accuracy of the current models proposed in [31]. For brevity purposes this numerical implementation will not be discussed in this manuscript.

Therefore, the four proposed ML methods presented herein are general and can be used for any type of data set that consists of input features and an output parameter. Regardless of this feature, it was chosen to demonstrate the numerical performance of the four proposed ML algorithms on larger in size data sets and data sets that refer to full-scale structures, where the availability of data sets that derive from experimental tests is not a feasible option since it is practically impossible to either get tens of thousands of experimental results related to the shear capacity of RC beams or columns, and where it is also impossible to get data sets on full-scale structures that were tested in laboratories for a specific loading condition or at a laboratory with a seismic table. The number of these type of tests across the world is limited thus cannot be used to form a data set of 500 or 1000 data. Therefore, experimentally validating 3D detailed models and using them for generating relatively large data sets was found to be a realistic approach [42, 64–66], thus the respective data sets derived from these research projects are used herein for the needs of this manuscript. It is also important to note that the all data sets were split into 15-85% for testing and training with the proposed ML algorithms.

4.1 Predicting the shear capacity of RC beams without stirrups

The first data set consists of data points on the shear capacity of RC beams without stirrups. This structural problem is still open, especially when dealing with the currently available design codes, which cannot provide sufficient accuracy [64].

4.1.1 Existing knowledge

Predicting the shear capacity of slender RC beams without stirrups through models derived from experimental data remains the primary source of input for the design of RC structural members. The main drawback of this approach is the limited number of available experimental data, which cannot cover the entire spectrum in beam geometries, load scenarios, boundary conditions, material properties, and reinforcement ratios, to allow the development of an extensive database. Furthermore, predicting the shear capacity of deep beams or the effect of fiber-reinforced polymer (FRP) rebars within the concrete domain make this problem even more unmanageable due to the arch action that needs to be accounted for in the first case and the poly-parametric influence of FRP in the overall structural behavior in the second case. The existing design codes are either unable or need improvement when predicting the shear capacity of these RC members [41, 64].

In the last two decades, researchers began to explore the use of AI and ML algorithms by investigating the possibility of developing models through training that would be able to predict the shear strength of RC beams with and without stirrups [67–69]. According to these studies, the main objective was to develop prediction models by training AI algorithms based on existing experimental tests in an effort to derive an improved method for calculating the shear capacity of RC

beams with and without stirrups. Even though these studies used different ML algorithms based on ANN with different input and hidden layer nodes, the restriction of having a limited number of available experimental database did not allow the development of a model that could be used for the shear capacity prediction for any geometry or material characteristic. It is well known that even a large number (1,000-2,000) of experiments is not adequate for ANN training, especially when dealing with deep learning. For this reason, one of the main problems that researchers have to deal with, when using AI algorithms and training ML models, is the need for more extensive databases that can be used to develop accurate and objective prediction models.

For the needs of this paper, AI algorithms and 3D detailed modeling of RC structures were used. In [64], an extensive database of RC slender beams without stirrups was developed, where the training and testing were performed with different ML algorithms. Based on the proposed AI algorithms, the numerically generated data set is used to train the models for predicting their shear strength. The Reconan Multirun simulation software was used to generate and analyze approximately 36,000 slender RC beams without stirrups, where the obtained results were used for the training of the predictive models through higher-order regression. The developed predictive models were validated through the use of experimental data found in the literature that was also used to compare the prediction abilities of the design codes ACI318-14 (2014) [70], and ACI318-19 [71]. The closed-form solution proposed in [64] is used herein as a baseline model to highlight the accuracy of the four new proposed AI models in this study.

4.1.2 Numerical simulation of RC structures

The generation and modeling of thousands of RC beam models through the use of 3D detailed numerical simulations of hexahedral finite elements (FEs) for discretizing concrete and the embedded steel reinforcement is highly challenging and computationally demanding task. To allow for thousands of analyses to be performed in a reasonable amount of time, the adopted simulation approach should be characterized by three main properties: Modelling objectivity, numerical stability, and computational efficiency.

The first property is of significant importance since the adopted numerical modeling method must have the ability to objectively capture the capacity of RC structural members by accounting for the most important physical phenomena that occur during the loading of these structural members (microcracking and macrocracking of concrete, capturing the 3D stress field within the concrete domain, steel rebar-concrete interaction, discretizing the exact geometry of the structural member, steel yielding and steel rupturing). To account for these properties, the concrete domain was modeled with the 20-noded isoparametric hexahedral finite elements, whereas the steel reinforcement was modeled with embedded bar elements. Finally, vertical push-over analyses were performed by analyzing 3-point bending tests to obtain the shear capacity of each RC beam numerically.

4.1.3 Modeling of concrete and steel rebars

The adopted concrete material model taking into account the use of the hydrostatic and deviatoric stress components [72], using two moduli of elasticity (bulk K and shear G) and an equivalent external stress σ_{id} to describe the constitutive relations is presented in [73]. The normal and octahedral shear stresses (σ_0, τ_0) and strains (ϵ_0, γ_0) are used to form the concrete material relationships that use the bulk and shear moduli (K, G) describing the non-linear stress–strain relationships (σ_0-$\epsilon_{0(h)}$ and τ_0-$\gamma_{0(d)}$) combined with the use of the equivalent external stress σ_{id}. The constitutive relations have the following form:

$$\epsilon_0 = \epsilon_{0(h)} + \epsilon_{0(d)} = (\sigma_0 + \sigma_{id})/(3K_S) \tag{3}$$

$$\gamma_0 = \gamma_{0(d)} = \tau_0/(2G_S) \tag{4}$$

where K_S and G_S are the secant forms of bulk and shear moduli, respectively. At the first stages of loading, when the deviatoric stress is less than 50% of the ultimate strength, it is assumed that the concrete material will behave elastically, therefore, the elastic constitutive material matrix of the uncracked material is used [72]. The constitutive model of the uncracked concrete is given through the following expression:

$$C = \begin{bmatrix} 2G_t+\mu & \mu & \mu & 0 & 0 & 0 \\ \mu & 2G_t+\mu & \mu & 0 & 0 & 0 \\ \mu & \mu & 2G_t+\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & G_t & 0 & 0 \\ 0 & 0 & 0 & 0 & G_t & 0 \\ 0 & 0 & 0 & 0 & 0 & G_t \end{bmatrix} \tag{5}$$

where $\mu = K_t - 2G_t/3$. When the deviatoric stress exceeds 50% of the ultimate strength, the parameters $K_t = K_t(\sigma_0, \tau_0, f_c)$ and G_t=$G_t(\sigma_0,\tau_0,f_c)$, where f_c represents the uniaxial compressive concrete strength, are updated according to the current state of stress accounting for the concrete material deterioration due to microcracking.

The ultimate strength of concrete is expressed with the value of the ultimate deviatoric stress by using the expressions of Willam and Warkne [74].

$$\tau_{0u} = \frac{2\tau_{0c}(\tau_{0c}^2 - \tau_{0e}^2)\cos\theta + \tau_{0c}(2\tau_{0e} - \tau_{0c}) \times SQ}{4(\tau_{0c}^2 - \tau_{0e}^2)\cos^2\theta + (2\tau_{0e} - \tau_{0c})^2}, \tag{6}$$

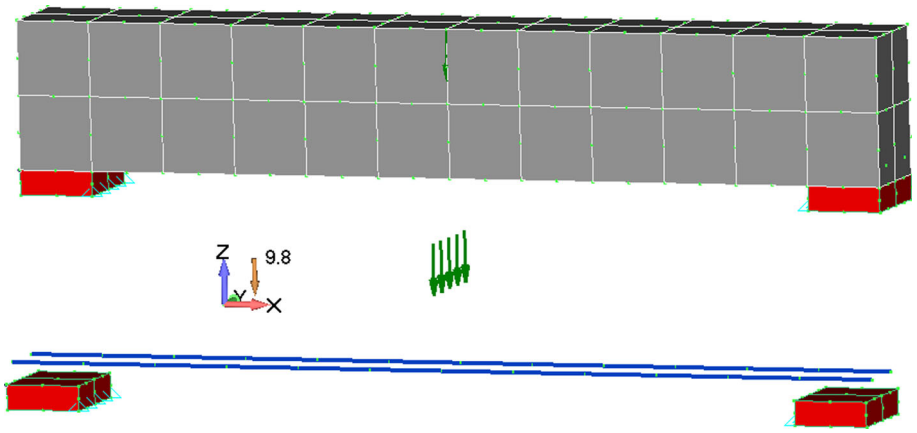


Fig. 5 Hexahedral and embedded rod element meshes of a simply supported slender RC beam with a net span of 150 cm and a 20×30 cm section

where

$$SQ = \sqrt{4(\tau_{0c}^2 - \tau_{0e}^2)\cos^2\theta + 5\tau_{0e}^2 - 4\tau_{0c}^2\tau_{0e}^2}$$

Furthermore, the steel reinforcement is modeled as embedded beam elements using the Menegotto, and Pinto [75] material model that accounts for the Bauchinger effect.

4.1.4 Numerical tests

Each slender beam model is simulated with 20-noded isoparametric hexahedral finite elements, whereas the steel rebars were modeled using 2-noded rod elements. A typical model of an RC beam can be seen in Fig 5, with the embedded rebar elements and the applied forces at the midspan, while the concrete cover was equal to 4 cm for all the considered models [64]. A total of 95 different beam geometries were created, as seen in Table 1, while 35,849 numerical tests were obtained according to the multirun analysis performed in [64].

4.1.5 Database assembly

A database of ten independent variables was created, which are divided in two groups. The first group consists of the variables corresponding to the geometry and the meshing of the beams, which is a time-consuming procedure and was initially created to constitute a basis for the analysis. In particular, for each beam, $L(mm)$ stands for net span, $b(mm)$ for width, and $d(mm)$ for effective depth. Additionally, f_c, E_c, f_t, β, E_s, f_y and ρ represent the uniaxial compressive concrete strength, concrete's Young modulus, tensile strength as a percentage of the compressive strength, the shear remaining strength at the surface of the cracks, steel's Young modulus, yielding stress, and the reinforcement ratio, respectively. The statistical characteristics of all the independent variables are presented in Table 2, where the coefficient of variation c_v (standard deviation σ to the mean μ) is shown below:

$$c_v = \frac{\sigma}{\mu} \tag{7}$$

The second group of variables was generated utilizing Eq. 8,

$$X_i^{'} = (\max(X_i) - \min(X_i)) \times \mathcal{U}(0, 1) + \min(X_i) \tag{8}$$

where $\mathcal{U}(0, 1)$ stands for the uniform distribution in the domain (0,1), X_i corresponds to the random variable i and their lower and upper limits are indicated with *min* and *max*, respectively.

Table 1 RC slender beam geometry [64]

a/a	Parameter	min	max
1	L (cm)	150	870
2	b (cm)	20	60
3	h (cm)	30	135
4	L/h	5.0	11.6
5	h/b	1.0	4.5

Minimum and maximum magnitudes of the 95 beam models

4.1.6 Distributed XGBoost-HYT-CV training numerical performance

The proposed distributed XGBoost-HYT-CV algorithm is used in this section to train and test the RC slender beam data set. Figure 6 demonstrates the model's learning curves without a drop-out and with a 10% drop-out effect. In the case of no drop-out (Fig. 6 a), a number of significant spikes of the objective function are observed, even though there is generally a decreasing tendency of MAPE with the number of epochs. These sudden increases in the objective function can be interpreted as an effect of averaging the weights, which is an inevitable source of error in distributed parallel computations. However, smoother learning curves are achieved with a 10% drop-out (Fig. 6 b), alleviating the phenomenon

Table 2 Statistical metrics of independent variables

	L	d	b	f_c	E_c	f_t	β	E_s	f_y	ρ
Average	5380.8	626.8	279.8	37.9	29961.5	0.058	0.034	2.00×10^5	500.1	0.0066
σ	2236.0	254.4	63.7	11.6	2896.4	0.023	0.009	5.77×10^3	57.7	0.0041
Median	5700.0	560.0	250.0	36.9	29951.4	0.057	0.034	2.00×10^5	499.9	0.0055
Minimum	1500.0	260.0	200.0	20.0	25000.0	0.020	0.020	1.90×10^5	400.0	0.0010
Maximum	8700.0	1310.0	600.0	60.0	35000.0	0.100	0.050	2.10×10^5	600.0	0.0200
c_v	0.416	0.406	0.228	0.305	0.097	0.404	0.252	0.029	0.115	0.627

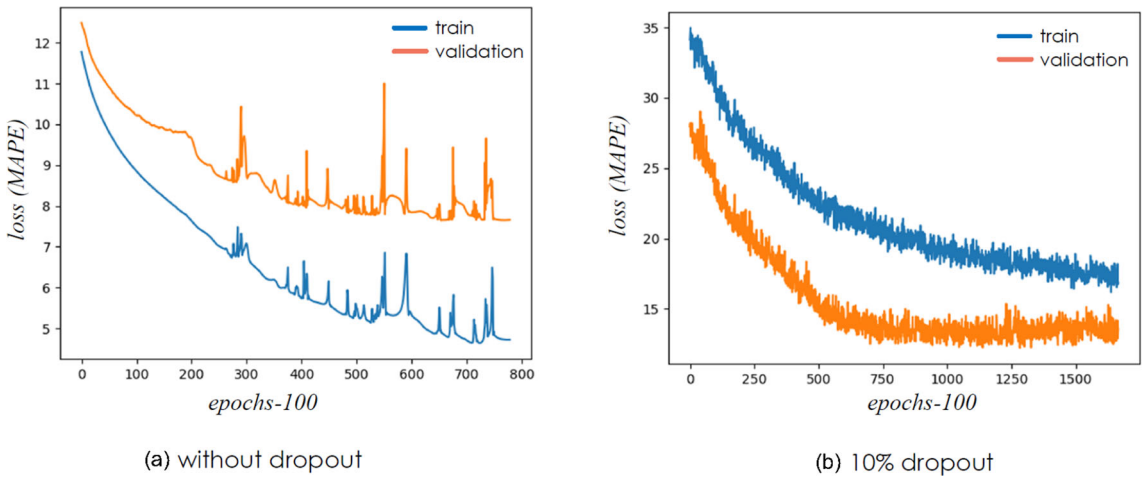


Fig. 6 Drop-out effect on the loss function during training in parallel for 100 epochs

of extreme spikes. Furthermore, it is worth noting that the validation curve for 10% drop-out is lower than the training curve, indicating the generalizability of the results.

A variety of experiments are performed to attain an optimal drop-out ratio. Figure 7 presents results for a drop-out equal to 0.01. As depicted in Fig. 7 b, it stabilizes around a constant value when the validation curve becomes higher than the training curve. This behavior of the loss function signifies over-training of the network resulting in an over-fitting of the derived predictive model.

4.1.7 Comparison between the DANN-MPIH-HYT and XGBoost-HYT-CV

To perform this comparison, an ANN with 10 layers and 1000 neurons per layer is trained, corresponding to more than 9 million weights that need to be optimized. The best loss (MAPE) attained with the DANN-MPIH-HYT learning model was 5.94%. The data set under study comprised values with an increment of 10 kN and an average of 98.22 kN, thus, giving an inevitable error of $\frac{10/2}{98.22} = 5.09\%$ which occurs on average [49]. Therefore, any improvement of the 5.94% loss cannot be practically attained. The corresponding MAPE for the tuned XGBoost-HYT-CV model was 5.825%, verifying the advanced accuracy of the proposed XGBoost-HYT-CV compared to the computationally demanding distributed deep learning algorithm. It is also important to note here that according to the best ML algorithm used in [64], the corresponding MAPE was 16.6%, which highlights the optimum performance and numerical superiority of the proposed ML algorithms.

4.1.8 POLYREG-HYT for automatic formula development

The predictive models that derive from DANN-MPIH-HYT and XGBoost-HYT-CV are black-boxes for the user that cannot be interpreted. On the other hand, closed-form formulae to be utilized for analysis and design purposes are much more desirable. The PR algorithm [64] for formula development extracted an analytical expression for the shear capacity V_c, comprising ten terms, as given in Eq. 9. The corresponding terms for f_c, d, b, L, Es, ρ, and f_t are used to calculate the required predictive formula for V_c.

$$\begin{aligned} V_c = {} & 0.168 \times d - 0.0196 \times \rho \times b \times d \\ & - 1.53 \times 10^{-10} \times L^3 + 0.0197 \times \rho \times f_c \times L \\ & - 809.8 \times \rho^2 \times d + 1.558 \times 10^{-04} \times \rho \times E_s \times d \\ & + 5.74 \times 10^{-09} \times b \times L^2 - 9.60 \times 10^{-12} \times E_s \times L^2 \end{aligned}$$

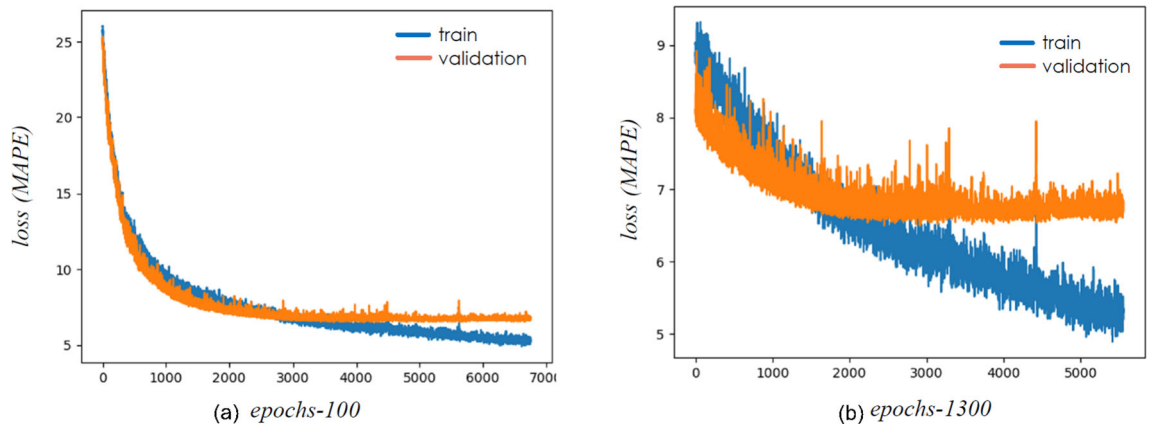


Fig. 7 Results for low drop-out=0.01 for different epochs

$$- 6853.2 \times \rho^2 \times f_c + 6.78 \times 10^{-03} \times f_t \times f_c \times d - 36.172 \quad (9)$$

It is important to note that even though the training and testing of this predictive model were performed exclusively on a data set generated by nonlinear numerical analyses, the model validation was performed on the out-of-sample, 100 experimental data found in the literature [64]. The following sections will summarize the closed-form formulae derived using the proposed ML algorithms and those previously implemented by the authors [64].

4.1.9 Comparison of ML models

Table 3 presents all ML models' performance metrics. It can be observed that XGBoost-HYT-CV attains the best accuracy for all studied metrics in the out-of-sample test set, followed by random forests with tuning. For the case of the test data set, the predictive model derived from XGBoost-HYT-CV decreases the MAMPE compared to the LR by 80.5%, while for the case of the polynomial regression, the XGBoost-HYT-CV reduces the MAMPE value by a 71.3%. The RF-HYT algorithm performs better than the LR and the polynomial regression delivers a 43% larger MAMPE than the XGBoost-HYT-CV. The performance of XGBoost-HYT-CV is also depicted in Fig. 8, while the corresponding error analysis shown in Fig. 9. Furthermore, the proposed RF-HYT is found to derive a 9.55% MAPE, and it is easy to observe the significant improvement in terms of MAPE on the validation data set when compared to the 16.6% derived from the random forests reported in [64]. The same applys for the newly proposed POLYREG-HYT that was found to outperform the older polynomial regression used in [64].

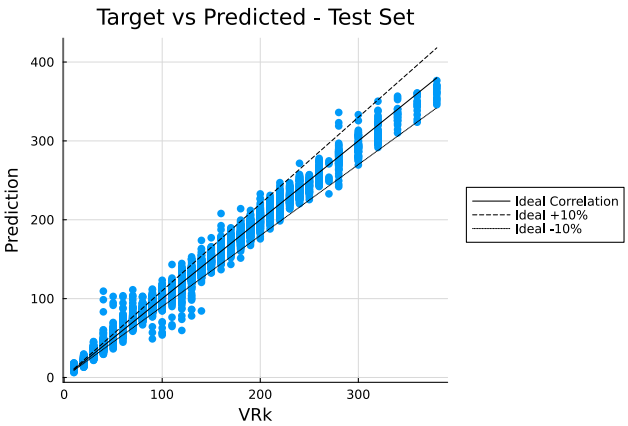


Fig. 8 Experimental vs predicted characteristic shear capacity V_{RK} with XGBoost-HYT-CV

Table 3 Comparison of performance metrics for RC beams without stirrups

Method	Data set	Pearson	MAPE %	MAMPE %	MAE	RMSE	Alpha	beta
Linear regression	Train	0.874	30.030	22.065	26.397	34.683	0.764	28.184
Linear regression	Test	0.877	30.847	22.362	26.215	34.318	0.757	29.283
POLYREG-HYT	Train	0.937	22.721	15.003	17.948	24.922	0.878	14.588
POLYREG-HYT	Test	0.937	23.381	15.208	17.829	24.963	0.879	14.840
XGBoost-HYT-CV	Train	0.999	2.156	1.511	1.808	2.480	0.996	0.494
XGBoost-HYT-CV	Test	0.995	5.825	4.369	5.122	7.445	0.978	2.643
RF-HYT	Train	0.988	8.527	6.921	8.280	11.262	0.958	4.980
RF-HYT	Test	0.985	9.550	7.665	8.986	12.205	0.952	5.778

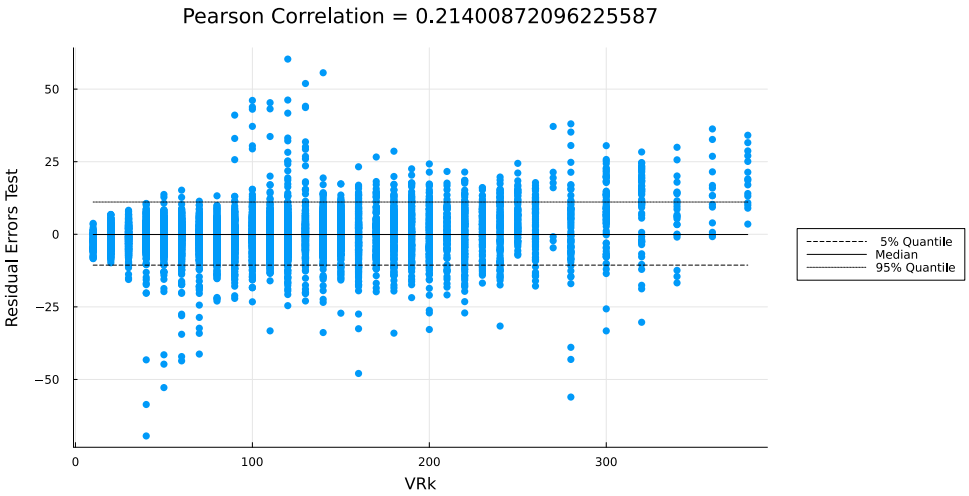


Fig. 9 Residual errors of XGBoost-HYT-CV model vs given V_{RK}

Table 4 Computational time for training with the RC beams without stirrups data set

Method	Time (minutes)	MAPE (Test)
Linear regression	0.001	30.847%
POLYREG-HYT	17	23.381%
RF-HYT	70	9.550%
XGBoost-HYT-CV	108	5.825%
DANN-MPIH-HYT	278	5.940%

Before investigating the proposed ML algorithms, it must be noted here that the computational performance was also evaluated for the four proposed algorithms. The RC slender beams without stirrups data set was used in this case where the training was performed on a personal computer with an i7-10510U CPU (2.30 GHz). The corresponding computational times for training the predictive models can be seen in Table 4, where XGBoost-HYT-CV was found to outperform all the proposed algorithms in terms of accuracy and it is almost 3 times faster compared to the DANN-MPIH-HYT which is an AI algorithm based on a deep learning artificial network framework. The training rounds for both models were set to 100. For the case of RF-HYT, the number of tune rounds were 100, while for the POLYREG-HYT the polynomial degree was equal to 3 and the number of formula features was also 100.

4.2 Predicting the fundamental period of RC structures with and without soil-structure interaction

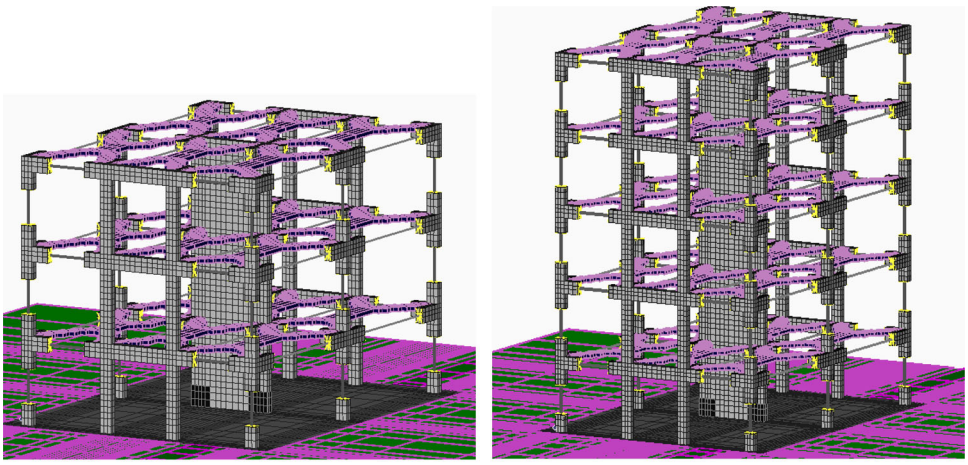
The use of a data set with fundamental periods of RC framed structures will be used in this section to test the numerical performance of the proposed ML algorithms. This data set consists of results obtained for both fixed and flexible-base models, as will be explained below.

4.2.1 Existing knowledge

One of the most important dynamic characteristics of structures is the fundamental period, which is associated with the dynamic performance of the structures as well as for estimating the enforced seismic load. Furthermore, during a seismic excitation, the interaction between the superstructure (building) and the substructure (soil) can become necessary as it affects the stress–strain distribution within the superstructure, altering the initial expected structural performance and the fundamental period of the structure [76, 77]. In general, it has been found that SSI can increase the fundamental period and the overall damping of the system. Thus it is essential to consider its effect on the structural response and to eliminate unsafe designs and unexpected damage development [78, 79] during earthquake excitations.

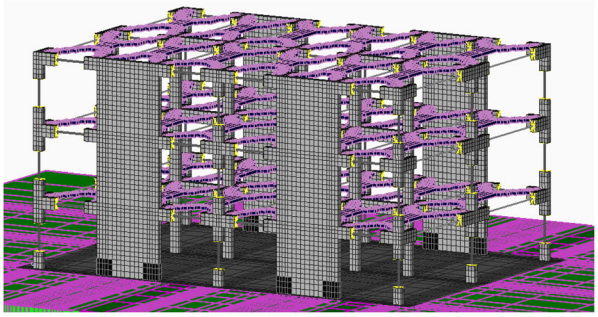
It is well documented that the current design codes need to consider the SSI effect on the fundamental period of a structure, while the effect of the shear walls in the stiffness distributions is not adequately considered, especially in the current "Eurocode 8" [80], amongst others, as demonstrated in [42, 66]. Thus the objective is to create a data set comprising all accurate modal analysis results obtained from various numerical models. To demonstrate the superiority of the proposed deep learning algorithm, a data set of 790 modal analysis results is used to develop a new, more accurate and robust predictive model. Some of the models are depicted in Fig. 10, where the SSI effect is considered with a discretization of the soil domain using hexahedral elements. In addition, the use of shear walls and different types of soil domains, and asymmetrically positioned shear walls were also considered during the validation procedure.

According to the latest research study on developing a closed-form formula for predicting the fundamental period

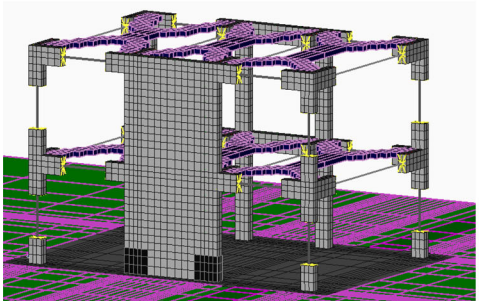


(a) 8.9x12.5x9m – ρ = 42.03%

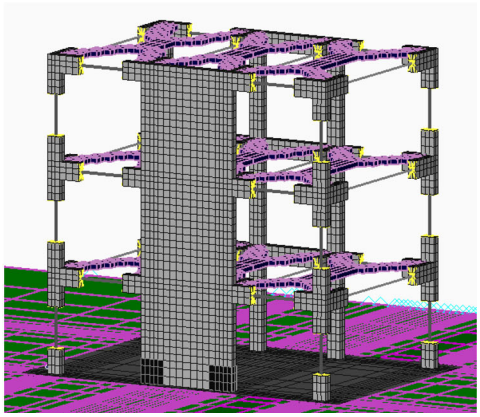
(b) 8.9x12.5x15m – ρ = 42.03%



(c) 12.25x17.4x9m – ρ = 69.05%



(d) 6.25x8.9x6m – ρ = 54.72%



(e) 6.25x8.9x9m – ρ = 54.72%

Fig. 10 RC building models with soil. **a** 3-storey and **b** 5-storey with a shear wall in the middle, **c** 3-storey shear walls at the perimeter and **d** 2-storey and **e** 3-storey with a single shear wall (asymmetric cases). [42]

of RC structures with and without SSI [42], a 20-term formula was proposed according to Eq. 10.

$$
\begin{aligned}
T = {} & (0.0292939 \times H) - (0.000150825 \times \rho \times H) \\
& + \left(0.00000582242 \times H \times B^2\right) \\
& + \left(0.00000330369 \times \rho \times H^2\right) \\
& + (0.000215881 \times H \times L) \\
& - \left(1.89375x10^{-15} \times {E_s}^2 \times D\right) \\
& + (0.00000323855 \times L \times H \times D) \\
& - (0.00000646154 \times \rho \times B \times H) \\
& - (0.0000000000925478 \times H \times E_s \times D) \\
& - (0.0000000000406192 \times \rho \times E_s \times D)
\end{aligned}
$$

$$
\begin{aligned}
&+ \left(0.000000194394 \times D \times \rho^2\right) \\
&+ (0.0037148 \times B) \\
&+ (0.000000358861 \times \rho \times H \times D) \\
&+ \left(0.0000000000662381 \times E_s \times D^2\right) \\
&- \left(0.000000278639 \times D^3\right) \\
&- (0.000000000113737 \times L \times E_s \times D) \\
&- \left(0.0000016727 \times B^3\right) \\
&+ (0.0000309934 \times L \times D) - (0.00178654 \times L) \\
&+ \left(0.000000645744 \times L^3\right) + 0.00239996 \qquad (10)
\end{aligned}
$$

where H the building's height (m), ρ the percentage shear walls (%), E_s the soils' modulus of elasticity (kPa), L the length of the building parallel to the oscillating direction (m), B is the width of the building perpendicular to the oscillating direction (m), D is the soil depth (m).

The fundamental periods of the structures were computed by Reconan FEA software code.

4.2.2 Database assembly

Using 790 variations of the type of structures shown in Fig. 10, the data set was constructed numerically. The statistical characteristics of the features (D_s: Soil depth, E: Young Modulus of Soil, H: Height of building, L: building length, B: out of plane length, ρ: shear wall ratio at ground floor), and the target (T: fundamental period) variables, are reported in Table 5.

In the following, the data set was split into the training (80%) and testing (20%) sets, while a part of the train set (20%) was kept for validating the deep learning during training.

4.2.3 Performance of ML models

Table 6 presents all ML models' performance metrics. It can be observed that the XGBoost-HYT-CV attains the best accuracy for Pearson, MAMPE, MAE, and RMSE metrics. On the other hand, RF-HYT with tuning gives higher accuracy for MAPE, alpha, and beta metrics and similar accuracy with XGBoost-HYT-CV for the other metrics. POLYREG-HYT also attains a high accuracy, having the advantage of producing a closed-form formula. It is noteworthy to mention here that the MAMPE error of the XGBoost-HYT-CV on the testing data set was 4.03% compared to the LR and POLYREG-HYT methods that resulted in a 13.4% and 7.4% MAMPE, respectively. The corresponding MAMPE value achieved by the RF-HYT was 4.3%. The results reported in Table 6 confirm that the proposed ML algorithms can derive an accurate predictive model with the XGBoost-HYT-CV outperforming all other ML algorithms.

To highlight the significant improvement and the numerical superiority of the proposed ML algorithms compared to those presented in [42], the respective MAPE of the optimum predictive model was equal to 5.68%. XGBoost was also implemented by Gravett et al., [42], where the corresponding MAPE was equal to 30.6%. This numerical result performed with the data set considered in [42], revealed that the XGBoost is the least suitable method, while the proposed XGBoost-HYT-CV, achieves MAMPE 5.06%, which is significantly smaller than the corresponding 30.6% derived from the XGBoost as presented in [42], but is also lower than 5.68%, which was the lowest MAPE obtained by the ML generated predictive models in [42]. Furthermore, the proposed RF-HYT was found to have the minimum MAPE (4.8%) out of the four predictive models of Table 6 for the considered test data set. According to the reported MAPE of the random forests used by Gravett et al., [42], the corresponding numerical error was equal to 11.46%. This improvement is attributed to the hyperparameter tuning that leads to more accurate predictive models.

4.2.4 Sensitivity analysis

Performing a sensitivity analysis has a twofold purpose. The first one is to assess the impact of a particular input on the target variable, and the second is to evaluate the reliability of the predictive model. The sensitivity analysis curves for ρ, the significance levels (25% and 75% median), and the best performing models XGBoost-HYT-CV and RF-HYT are given in Figs 11 and 12, respectively. It can be observed a decreasing pattern followed by a plateau for $\rho > 40$ for both models considered. This indicates that the models' accuracy was not achieved by chance. Still, they capture the proper relationship between input–output, highlighting the proposed ML algorithms' ability to derive high-accurate predictive models. Finally, Fig. 13 Finally illustrates the ability of the XGBoost-HYT-CV visually in predicting the values found in the test data set.

4.3 Predicting the fundamental period of steel structures with and without soil-structure interaction (SSI)

The next data set that is used to evaluate the ML algorithms consists of fundamental period results of steel framed structures, including the SSI effect.

4.3.1 Existing knowledge

Current building design codes use empirical oversimplified relations to predict the fundamental period of structures [65,

Table 5 Statistical properties of characteristic parameters

Metric	D_s (m)	E (kPa)	H (m)	L (m)	B (m)	ρ (%)
Mean	23.518	25 000	15.44	13.201	11.99	22.125
StD	20.24	82 400	9.58	6.86	5.048	32.416
Median	22.5	300	12	12.25	12.25	0
Minimum	0	65	3	3.4	3.4	0
Maximum	60	300 000	30	34.4	34.4	85.294

Table 6 Comparison of performance metrics for fundamental period of RC structures with and without SSI

Method	Data set	Pearson	MAPE %	MAMPE %	MAE	RMSE	Alpha	Beta
Linear regression	Training	0.987	13.827	7.688	0.037	0.049	0.974	0.012
Linear regression	Testing	0.986	13.366	7.405	0.037	0.050	0.967	0.017
POLYREG-HYT	Training	0.994	7.053	5.172	0.025	0.034	0.988	0.006
POLYREG-HYT	Testing	0.993	7.449	5.234	0.026	0.036	1.003	0.0004
XGBoost-HYT-CV	Training	0.999	3.550	2.566	0.012	0.018	0.977	0.011
XGBoost-HYT-CV	Testing	0.996	5.061	4.032	0.020	0.028	0.974	0.011
RF-HYT	Training	0.998	3.354	3.069	0.015	0.021	0.987	0.005
RF-HYT	Testings	0.995	4.781	4.362	0.022	0.030	0.987	0.004

81], requiring only the height of the structure and do not account for the actual 3D geometry or for any SSI effect. An estimation of the fundamental period of steel structures according to EC8 EC1998, is given below:

$$T_1 = C_t\,(H)^{0.75}$$

where:

$C_t = 0.085$ for moment-resistant space steel frames

$C_t = 0.075$ for eccentrically braced steel frames

As well as according to ASCE 7-05 [82]:

$T_1 = 0.0724\,(H)^{0.8}$ for steel moment-resisting frames

$T_1 = 0.0731\,(H)^{0.75}$ for eccentrically braced steel frames

Furthermore, Cinitha [83] proposed a formula that takes into account the geometry of the structure and particularly the plan area of the building ($L \times B$) as shown below:

$$T_1 = C_0\,(L \times B)^{0.3289 \times \alpha}\,, \tag{11}$$

with

$C_0 = 0.0247e^{0.1305 \times H}$

$\alpha = 0.4473e^{-0.0441 \times H}$

In [66], a total of 576 numerical models using Reconan FEA code created and analyzed to develop a data set of 1,152 fundamental periods. The data set was used to train the predictive model through the ML algorithm presented in Sect. 2.1 and formulate a 40-feature closed-form formula, which was also validated using out-of-sample data. This data set is also used in investigating the numerical response of the proposed in this study XGBoost-HYT-CV parallel training algorithm. The proposed closed-form formula found in [66]

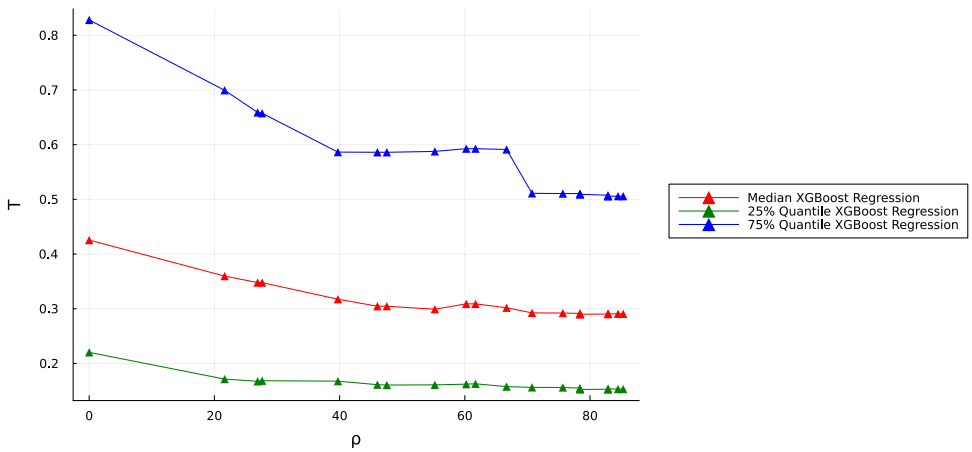


Fig. 11 Sensitivity curves for ρ with XGBoost-HYT-CV

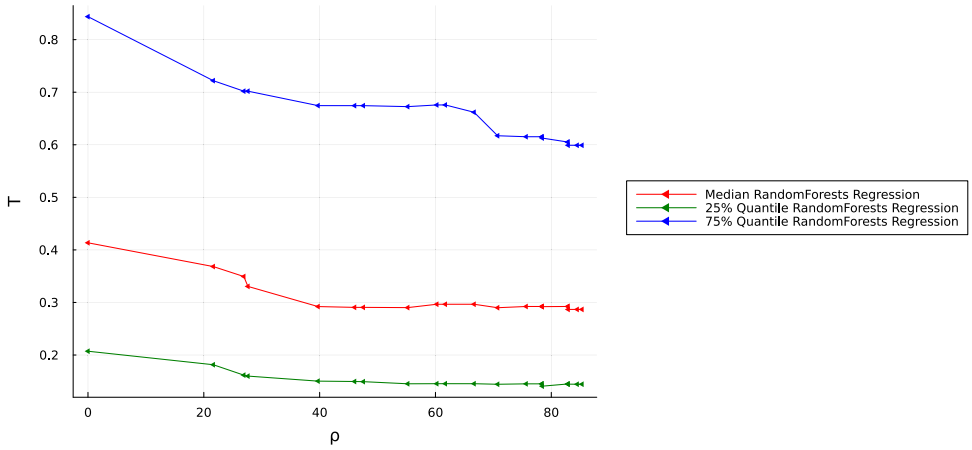


Fig. 12 Sensitivity curves for ρ with RF-HYT

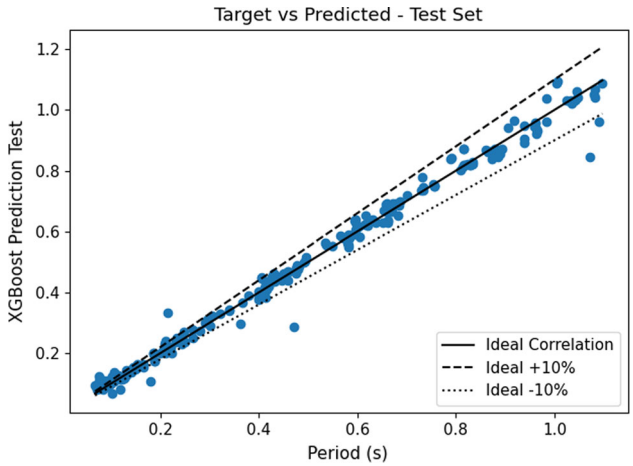


Fig. 13 Numerical vs predicted fundamental period of RC structures on the test data set with XGBoost-HYT-CV

is given in Eq. 12, which resulted in a 2.7% MAPE.

$$
\begin{aligned}
T = {} & 0.194630 \times lH^2 + 0.0580556 \times CO^2 \times B \\
& - 9.39027 \times InvCO \times InvB \times lB \\
& - 8.49213 \times InvL \times CO \times H \\
& - 41.8498 \times InvCO \times lL \times H \\
& - 8.14564 \times InvE \times E \times H \\
& - 0.800465 \times CO \times B \times H + 114.808 \\
& \times InvCO \times InvB \times H \\
& + 46.6778 \times InvCO \times InvB^2 \\
& + 0.0631499 \times B^2 \times H \\
& + 4.20803 \times lB \times CO \times H - 0.144945 \\
& \times lL \times H \times L \\
& + 0.847694 \times B \times H \times InvL + 9.37930 \\
& \times InvL^2 \times H \\
& - 1.08930 \times InvCO^2 \times L + 4.04342 \times InvL \\
& - 0.251627 \times InvL \times CO \times B - 0.00783561 \\
& \times InvB \times lCO \times lE \\
& + 0.523388 \times lL^2 \times InvCO + 0.0947335 \\
& \times InvH \times lH \times L \\
& + 46.8309 \times InvE \times H \times lDs + 0.00764850 \\
& \times lH \times B \\
& + 0.000161108 \times lL \times L \times lE - 20.5554 \\
& \times InvE \times CO \times Ds \\
& - 0.00474725 \times InvL^2 \times InvDs \\
& + 2.73101 \times InvL \times InvH \times CO \\
& + 0.403996 \times InvCO \times lB \times L \\
& - 0.0105914 \times lL \times L \times B \\
& - 0.228100 \times lB^2 \times CO + 0.00265642 \times InvL \times H^2 \\
& - 2.58386 \times InvB \times InvH \times CO \\
& + 5.84142 \times InvCO \times H \times L \\
& + 29.5168 \times InvCO \times H + 0.849560 \times InvL \\
& \times lH \times CO \\
& - 2.14776 \times InvB \times lH \times lCO \\
& + 1.34222 \times lB \times lH \times InvH \\
& - 0.00333495 \times lE \times L \times InvH \\
& - 2.64111 \times InvB^2 \times InvH \\
& + 71.1358 \times InvH \times Ds \times InvE \\
& - 17.9194 \times InvE \times lE \times lL \\
& - 1.16636
\end{aligned}
\tag{12}
$$

According to the features that were investigated in [66], the variables that were used to construct the above equation are the following:

T is the fundamental period (s)
D_s is the depth of soil (m)
E is the soils Young's Modulus (kPa)
H is the building height (m)

Table 7 minimum and maximum parameter values for model development

Parameter	Minimum	Maximum
Soil depth [m]	1	37.5
Soil E [kPa]	65 000	700 000
Height [m]	3.5	35
Length (along x-axis) [m]	5	15
Width (along y-axis) [m]	3	6

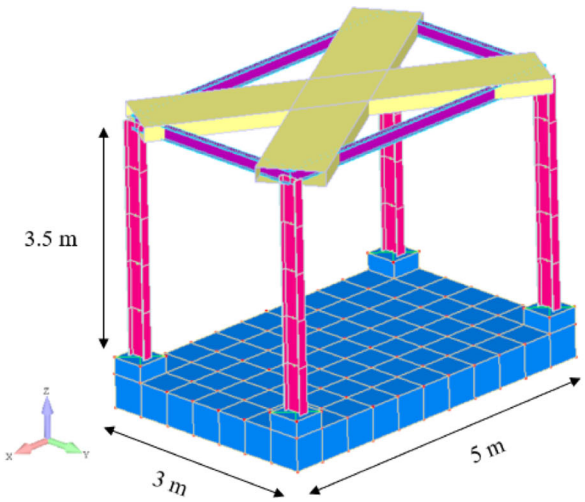
L is the length of the building parallel to the oscillating direction (m)
B is the width of the building perpendicular to the oscillating direction (m)
CO is the orientation of the columns (either a 1 or 2)
$lParameter$ is $\ln(Parameter + 1)$ i.e. $lD_s = \ln(D_s + 1)$
$InvParamete$ is $\frac{1}{Parameter+1}$ i.e. $InvD_s = \frac{1}{D_s+1}$

4.3.2 Database assembly

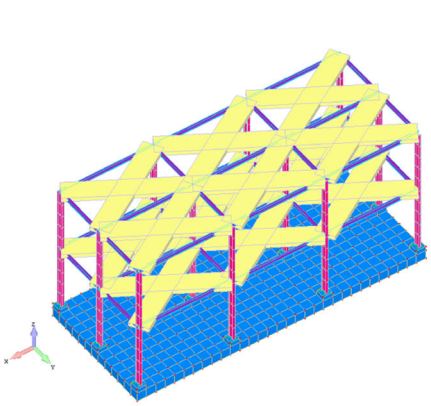
Figure 14 (a) shows the smallest structural model used to create a data set of 1,152 fundamental mode results, while Fig. 14 (b,c) shows a multi-span 2-storey frame with and without the soil domain that was discretized with 8-noded isoparametric hexahedral elements. Table 7 shows the minimum and maximum values of the variables considered during the training phase. This pilot project is based on the use of one to 10-storey buildings with single and multiple spans founded on soft, medium, and hard (fixed base) soil. Finally, a relevant mass was assumed to equal the mass of a 150 mm thick slab with a 2.0 kN/m^2 live load (residential buildings).

4.3.3 Comparison of ML models

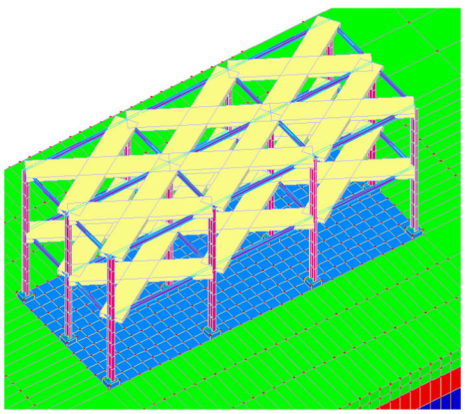
Table 8 summarizes the numerical performance of the ML algorithms from both training and testing data sets. As can



(a) Initial model



(b)



(c)

Fig. 14 2-storey steel building. Triple span in the long direction, double in the short direction **a** fixed base with raft foundation **b** flexible base with hexahedral soil mesh. [66]

Table 8 Comparison of performance metrics for fundamental period of steel structures with and without soil-structure interaction

Method	Data set	Pearson	MAPE %	MAMPE %	MAE	RMSE	alpha	beta
Linear regression	Train	0.738	64.933	47.315	0.628	0.719	0.544	0.605
Linear regression	Test	0.709	69.197	48.735	0.655	0.740	0.520	0.642
POLYREG-HYT	Train	0.999	1.478	0.809	0.011	0.028	0.999	0.001
POLYREG-HYT	Test	0.998	1.172	0.981	0.019	0.019	0.999	0.003
XGBoost-HYT-CV	Train	1.000	0.213	0.121	0.002	0.003	1.000	0.0002
XGBoost-HYT-CV	Test	0.998	2.740	2.345	0.032	0.060	0.984	0.014
RF-HYT	Train	0.994	9.707	5.369	0.071	0.123	0.952	0.059
RF-HYT	Test	0.986	17.422	9.624	0.129	0.183	0.922	0.098

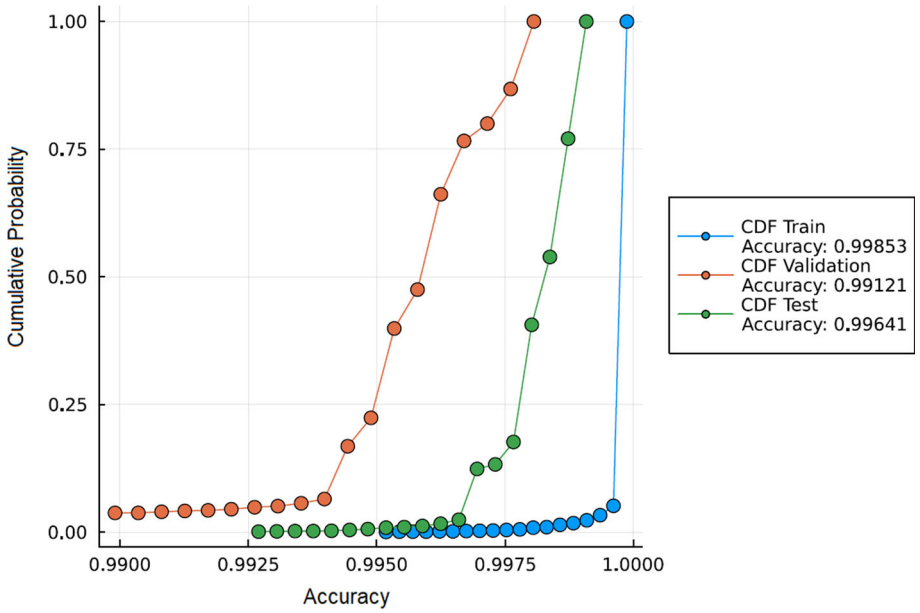


Fig. 15 Cumulative distribution functions for train, test and validation sets

be seen, POLYREG-HYT achieves the best error metrics for this problem, outperforming the previous accuracy reported in [66] decreasing the MAPE by 57%. XGBoost-HYT-CV achieves the second best accuracy for all error metrics with 2.7% MAPE values which is almost 20 times lower compared to the LR method. When compared with the random forests, XGBoost-HYT-CV is found to attain a four times lower MAPE value. For the case of the MAMPE and MAE metrics, the difference is similar or larger compared to the corresponding MAPE values. It must be noted that the RF-HYT method was tuned exhaustively for the needs of this problem but still did not achieve the accuracy of the XGBoost-HYT-CV method.

4.3.4 Statistical reliability of model selection

In Fig. 15, the cumulative distribution functions (CDFs) are presented for the models exhibiting higher accuracy than the median validation one. This figure shows that CDF is located on the right of the corresponding validation CDF in the entire test which indicates that the best model was selected following a robust procedure. Furthermore, the trained CDF indicates higher accuracy when using the training set. However, this is balanced in the sense of ANNs, as the test set still exhibits higher accuracy. CDFs are a valuable tool for understanding and evaluating the model development procedure. Therefore, using CDFs after training and testing a predictive model is highly recommended to achieve a robust empirical process for evaluating the hyper-parameter tuning.

4.3.5 Impact of data-set volume evaluation

In Fig. 16, the XGBoost-HYT-CV results are presented when training is performed sequentially keeping the 25% − 100% of the observations, with 20 intermediate rounds. In addition, a linear model is fitted within the graph for all and 50% of the obtained results. The observed flattening of the curve during the last iterations indicates that there are few or no accuracy

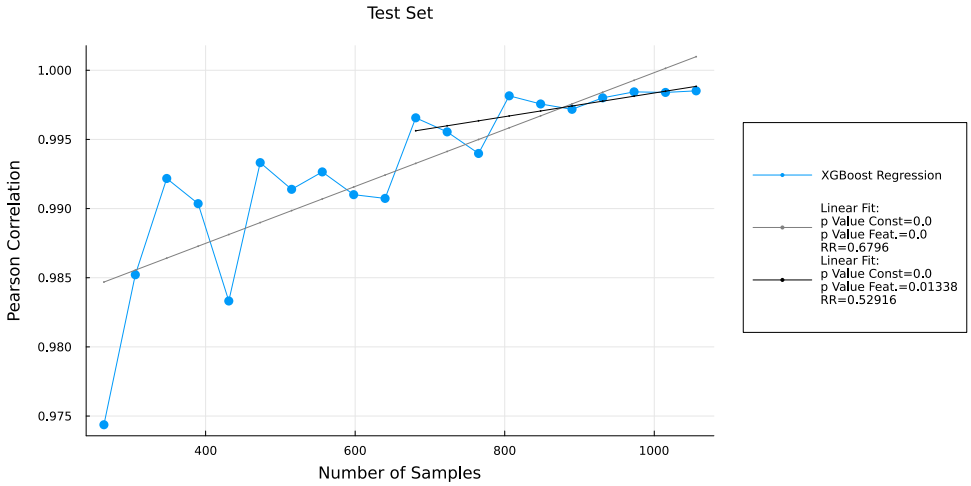


Fig. 16 Impact of data-set volume

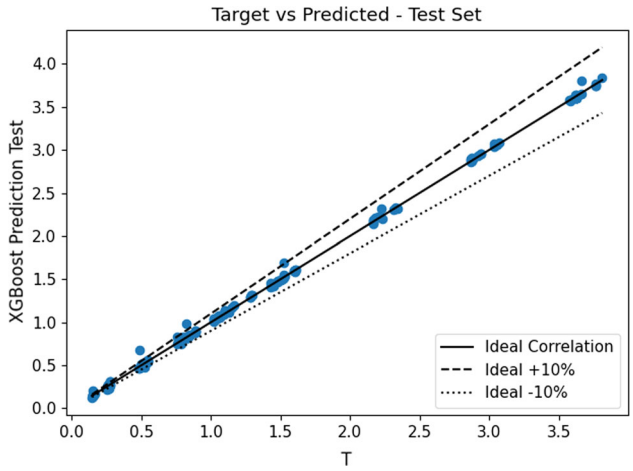


Fig. 17 Numerical vs predicted fundamental period of steel structures on the test data set with XGBoost-HYT-CV

enhancements that the model can attain at this stage. It is noteworthy to notice that the tuning of the optimal parameters obtained by the proposed tuning algorithm for all 20 intermediate models of XGBoost-HYT-CV was a computationally demanding task. However, this process is a good indicator of the adequacy of the number of samples of the selected training data set. Figure 17 illustrates the numerical vs predicted fundamental period of steel structures on the test data set obtained from the XGBoost-HYT-CV. It is easy to observe the accuracy of the derived predictive model as it derived from the numerical analysis.

5 Conclusions

Four enhanced ML algorithms are presented in this research work, illustrating their numerical performance and ability to extract predictive models. The methods proposed herein with hyperparameter tuning are namely the MPI & Horovod-based parallel training DANN-MPIH-HYT, the polynomial regression POLYREG-HYT, the extreme gradient boosting XGBoost-HYT-CV and the random forest RF-HYT. The proposed ML algorithms incorporate the use of a hyperparameter tuning algorithm that enhances the predictive model training procedure, decreases the computational effort while increasing the accuracy of the obtained predictive models. Similar improved numerical performances were observed when implementing the four proposed methods for the data sets published in [31, 63] further highlighting their computational response.

As it was described in Sects. 2 and 3, the novelty presented in this article is found in the new distributed algorithm as a baseline for the ML models, and the POLYREG-HYT's feature selection algorithm which is novel. A holistic framework for analyzing engineering datasets was also presented, taking into consideration of all the appropriate parts, like error analysis, sensitivity analysis, and comparison among the predictive models during training. Furthermore, all the data sets and algorithms that were developed for the needs of this research work are provided for free through an open-source link.

The DANN-MPIH-HYT algorithm managed to achieve accurate predictive models for the case of RC slender beams. The proposed method's parallel scalability was also demonstrated by using the 64 Tesla GPUs of Cyclone supercomputer, where the speed-up was found to be practically linear demonstrating the efficiency of the proposed algorithmic structure for parallel training. Additionally, the proposed parallel algorithm achieved a 5.94% MAPE when used on the RC slender data set, which is ideal compared to the computed inevitable average error derived from the data set's nature. This also represents a 3-fold decrease of the corresponding MAPE 16.6% achieved in previous studies.

The proposed POLYREG-HYT algorithm, with the proposed feature selection algorithm for the case of combina-

torial optimization, was parametrically investigated and was found to be able to develop predictive accurate models. Furthermore, the obtained average MAMPE of this algorithm was found to be 7.14% on average in the parametric investigation of the four structural-related problems. Last but not least, this algorithm can produce closed-form expressions for the required quantities in a computationally efficient manner.

The two additional ML algorithms proposed and parametrically investigated using the four test cases considered, namely the XGBoost-HYT-CV and the RF-HYT, were found numerically superior, compared to the others, with XGBoost-HYT-CV being the best in practically all categories related to error metrics. Although RF-HYT achieved significantly high accuracy (average MAMPE 7.22% and MAPE 10.58%), the XGBoost-HYT-CV was found to achieve an average MAMPE value of 3.58% and a corresponding average MAPE value equal to 4.54% for all test cases considered. This finding demonstrates the numerical superiority of XGBoost-HYT-CV (5.825%) which can give, on average, 2.5 times lower MAPE than RF-HYT and a slightly more accurate MAPE than the deep learning algorithm DANN-MPIH-HYT (5.94%). However, after comparing the computational efficiency of the two methods, it was found that when using the same CPU to train on the RC slender beams without stirrups data set, XGBoost-HYT-CV was almost 3 times faster than DANN-MPIH-HYT which is attributed to the large number of weight factors combinations and training set that is required when deep learning is engaged.

In order to evaluate the overall numerical response of the proposed ML algorithms, the derived errors from this research work were compared to the errors reported when using the previously published ML algorithms. According to the comparison performed on MAPE and MAMPE values, it was found that the proposed ML algorithms outperform their predecessors in all test cases considered.

Many research and industrial applications of ML algorithms focus on selecting the best possible model for a particular data set. However, the statistical reliability of model selection to attain a robust model as well as cleaning the data set to avoid errors and useless resource allocation are both essential and should also be investigated. To this extent, a three-stage automatic process was proposed in this study to address this significant data set-related issue, while error analysis revealed substantial patterns for the ML models contributing to the evaluation of the final model.

Furthermore, the ML black-box model that derives from algorithms that do not produce a closed-form formula can be scrutinized with a random perturbation of the features at various significance levels, using the proposed sensitivity analysis that was presented as a part of the overall evaluation of the derived predictive models.

It is important to note here that the proposed ML algorithms do not constitute a fit-all-solution, since the data sets that were used to investigate their numerical response referred to structural related problems. Therefore, engineers are always advised to deploy a set of different ML algorithms especially when dealing with a newly developed data set. This will ensure the development of objective and extendable predictive models.

Finally, the current demand for data-centric AI is considered, with an algorithm for evaluating the data set volume's impact on each model's performance. These procedures are cleaning the data set, QR-factorization to avoid multicollinearity, generation of CDFs, error analysis, and impact of data-set volume, which should be implemented when training, testing, and validating any predictive model..

Acknowledgements Parts of the runs were performed on the MeluXina (https://docs.lxp.lu/) as well as Cyclone (https://hpcf.cyi.ac.cy/) Supercomputers.

Funding This work received financial support from the EuroCC Project (GA 951732) and EuroCC 2 Project (101101903) of the European Commission. Open access funding provided by University of Pretoria.

References

1. Lagaros ND, Charmpis DC, Papadrakakis M (2005) An adaptive neural network strategy for improving the computational performance of evolutionary structural optimization. Comput Methods Appl Mech Eng 194(30):3374–3393. https://doi.org/10.1016/j.cma.2004.12.023
2. Sundar V, Shields MD (2016) Surrogate-enhanced stochastic search algorithms to identify implicitly defined functions for reliability analysis. Struct Saf 62:1–11. https://doi.org/10.1016/j.strusafe.2016.05.001
3. Roy A, Manna R, Chakraborty S (2019) Support vector regression based metamodeling for structural reliability analysis. Probab Eng Mech 55:78–89. https://doi.org/10.1016/j.probengmech.2018.11.001
4. Singh K, Kapania R (2021) Alga: active learning-based genetic algorithm for accelerating structural optimization. AIAA J 59(1):330–344. https://doi.org/10.2514/1.J059240
5. Böhringer P, Sommer D, Haase T, Barteczko M, Sprave J, Stoll M, Karadogan C, Koch D, Middendorf P, Liewald M (2023) A strategy to train machine learning material models for finite element simulations on data acquirable from physical experiments. Comput Methods Appl Mech Eng 406:115894
6. Samaniego E, Anitescu C, Goswami S, Nguyen-Thanh VM, Guo H, Hamdia K, Zhuang X, Rabczuk T (2020) An energy approach

to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications. Comput Methods Appl Mech Eng 362:112790
7. Saha S, Gan Z, Cheng L, Gao J, Kafka OL, Xie X, Li H, Tajdari M, Kim HA, Liu WK (2021) Hierarchical deep learning neural network (hidenn): an artificial intelligence (AI) framework for computational science and engineering. Comput Methods Appl Mech Eng 373:113452
8. Birky D, Ladd J, Guardiola I, Young A (2022) Predicting the dynamic response of a structure using an artificial neural network. J Low Freq Noise Vib Active Control 41(1):182–195
9. Oishi A, Yagawa G (2017) Computational mechanics enhanced by deep learning. Comput Methods Appl Mech Eng 327:327–351. https://doi.org/10.1016/j.cma.2017.08.040
10. Ge Y, He Z, Li S (2023) A machine learning-based probabilistic computational framework for uncertainty quantification of actuation of clustered tensegrity structures. Comput Mech 72:431–450
11. Huang D, Fuhg JN, Weißenfels C, Wriggers P (2020) A machine learning based plasticity model using proper orthogonal decomposition. Comput Methods Appl Mech Eng 365:113008
12. Weber G, Pinz M, Ghosh S (2022) Machine learning-enabled self-consistent parametrically-upscaled crystal plasticity model for ni-based superalloys. Comput Methods Appl Mech Eng 402:115384
13. Frankel A, Hamel CM, Bolintineanu D, Long K, Kramer S (2022) Machine learning constitutive models of elastomeric foams. Comput Methods Appl Mech Eng 391:114492
14. Fuhg JN, Hamel CM, Johnson K, Jones R, Bouklas N (2023) Modular machine learning-based elastoplasticity: Generalization in the context of limited data. Comput Methods Appl Mech Eng 407:115930
15. Liu W, Karniadakis G, Tang S (2019) A computational mechanics special issue on: data-driven modeling and simulation–theory, methods, and applications. Comput Mech 64:275–277
16. Asteris PG, Nikoo M (2019) Artificial bee colony-based neural network for the prediction of the fundamental period of infilled frame structures. Neural Comput Appl 31(9):4837–4847
17. Charalampakis AE, Tsiatas GC, Kotsiantis SB (2020) Machine learning and nonlinear models for the estimation of fundamental period of vibration of masonry infilled rc frame structures. Eng Struct 216:110765
18. Asteris PG, Koopialipoor M, Armaghani DJ, Kotsonis EA, Lourenço PB (2021) Prediction of cement-based mortars compressive strength using machine learning techniques. Neural Comput Appl 33(19):13089–13121
19. Li H, Kadav A, Kruus E, Ungureanu C (2015) Malt: distributed data-parallelism for existing ml applications. In: Proceedings of the Tenth European Conference on Computer Systems, pp 1–16
20. Chen C-C, Yang C-L, Cheng H-Y (2018) Efficient and robust parallel dnn training through model parallelism on multi-gpu platform, arXiv preprint arXiv:1809.02839
21. Shallue CJ, Lee J, Antognini J, Sohl-Dickstein J, Frostig R, Dahl GE (2018) Measuring the effects of data parallelism on neural network training, arXiv preprint arXiv:1811.03600
22. Li H, Kafka OL, Gao J, Yu C, Nie Y, Zhang L, Tajdari M, Tang S, Guo X, Li G, Tang S, Cheng G, Liu WK (2019) Clustering discretization methods for generation of material performance databases in machine learning and design optimization. Comput Mech 64:281–305
23. Park JH, Yun G, Chang MY, Nguyen NT, Lee S, Choi J, Noh SH, Choi Y-r (2020) Hetpipe: Enabling large dnn training on (whimpy) heterogeneous gpu clusters through integration of pipelined model parallelism and data parallelism. In: 2020 USENIX Annual Technical Conference (USENIX ATC 20), pp 307–321
24. Abueidda D, Koric S, Sobh N (2020) Topology optimization of 2D structures with nonlinearities using deep learning. Comput Struct 237:106283. https://doi.org/10.1016/j.compstruc.2020.106283
25. Gorshenin A, Kuzmin V (2020) Analysis of configurations of LSTM networks for medium-term vector forecasting. Informatika i ee Primeneniya 14(1):10–16. https://doi.org/10.14357/19922264200102
26. Do N, Taberner A, Ruddy B (2019). Design of a linear permanent magnet transverse flux motor for needle-free jet injection. https://doi.org/10.1109/LDIA.2019.8770975
27. Miller R, Moore B, Viswanathan H, Srinivasan G (2017) Image analysis using convolutional neural networks for modeling 2D fracture propagation, vol 2017, pp 979–982. https://doi.org/10.1109/ICDMW.2017.137
28. Kohar CP, Greve L, Eller TK, Connolly DS, Inal K (2021) A machine learning framework for accelerating the design process using CAE simulations: an application to finite element analysis in structural crashworthiness. Comput Methods Appl Mech Eng 385:114008
29. Mourlas C, Papadrakakis M, Markou G (2017) A computationally efficient model for the cyclic behavior of reinforced concrete structural members. Eng Struct 141:97–125
30. Markou G, Roeloffze W (2021) Finite element modelling of plain and reinforced concrete specimens with the Kotsovos and Pavlovic material model, smeared crack approach and fine meshes. Int J Damage Mech. https://doi.org/10.1177/1056789520986601
31. Ababu E, Markou G, Bakas N (2022) Using machine learning and finite element modelling to develop a formula to determine the deflection of horizontally curved steel i-beams
32. Markou G (2020) v2.00, Reconan, F.E.A. - User's Manual
33. Zinonos Z, Gkelios S, Khalifeh AF, Hadjimitsis DG, Boutalis YS, Chatzichristofis SA (2022) Grape leaf diseases identification system using convolutional neural networks and lora technology. IEEE Access 10:122–133. https://doi.org/10.1109/ACCESS.2021.3138050
34. Gkelios S, Sophokleous A, Plakias S, Boutalis Y, Chatzichristofis SA (2021) Deep convolutional features for image retrieval. Expert Systems with Applications 177:114940, https://doi.org/10.1016/j.eswa.2021.114940, https://www.sciencedirect.com/science/article/pii/S095741742100381X
35. Dimopoulos T, Bakas N (2019) Sensitivity analysis of machine learning models for the mass appraisal of real estate. case study of residential units in nicosia, cyprus. Remote Sens 11(24):3047
36. Bakas N, Koutsantonis D, Plevris V, Langousis A, Chatzichristofis S (2022) Inverse transform sampling for bibliometric literature analysis, in: The Thirteenth International Conference on Information, Intelligence, Systems and Applications. Ionian University, Corfu, Greece, 18–20 July 2022, IISA 2022. http://easyconferences.eu/iisa2022/
37. Plevris V, Solorzano G, Bakas N (2019) Literature review of historical masonry structures with machine learning, in: 7th ECCOMAS Thematic Conference on Computational Methods in Structural Dynamics and Earthquake Engineering, ECCOMAS, Crete, Greece, pp 1547–1562. https://doi.org/10.7712/120119.7018.21053
38. Bakas NP, Langousis A, Nicolaou M, Chatzichristofis SA (2019) A gradient free neural network framework based on universal approximation theorem, arXiv preprint arXiv:1909.13563
39. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
40. Xu B, Chen T (2014) Xgboost.jl. https://github.com/dmlc/XGBoost.jl
41. AlHamaydeh M, Markou G, Bakas N, Papadrakakis M (2022) Ai-based shear capacity of frp-reinforced concrete deep beams without stirrups. Eng Struct 264:114441
42. Gravett D, Taljaard V-L, Bakas N, Markou G, Papadrakakis M (2021) New fundamental period formulae for soil-reinforced concrete structures interaction using machine learning algorithms and anns

43. Liu X, Zhao D, Xiong R, Ma S, Gao W, Sun H (2011) Image interpolation via regularized local linear regression. IEEE Trans Image Process 20(12):3455–3469
44. Weisstein EW (2002) Least squares fitting–polynomial. http://mathworld.wolfram.com/LeastSquaresFittingPolynomial.html
45. Helwig NE (2017) Regression with polynomials and interactions. http://users.stat.umn.edu/~helwig/notes/polyint-Notes.pdf
46. Chen T, Guestrin C (2016) XGBoost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, KDD '16, ACM, New York, NY, USA, pp 785–794. https://doi.org/10.1145/2939672.2939785
47. Sadeghi B (2013) Decisiontree.jl
48. Bezanson J, Edelman A, Karpinski S, Shah VB (2017) Julia: a fresh approach to numerical computing. SIAM Rev 59(1):65–98
49. Bakas N, Markou G, Langousis A, Lavdas S, Chatzichristofis S (2023) NBML: computer software for data analysis and predictive modelling with artificial intelligence algorithms. https://github.com/nbakas/nbml/blob/main/docs/__nbml__.pdf
50. Bakas NP, Plevris V, Langousis A, Chatzichristofis SA (2022) ITSO: a novel inverse transform sampling-based optimization algorithm for stochastic search. Stoch Env Res Risk Assess 36(1):67–76
51. Plevris V, Bakas NP, Solorzano G (2021) Pure random orthogonal search (pros): A plain and elegant parameterless algorithm for global optimization. Appl Sci 11(11):5053
52. Ruder S (2016) An overview of gradient descent optimization algorithms, arXiv preprint arXiv:1609.04747
53. T. S. d (2016) The scikit-learn developers, Scikitlearn.jl (2007–2016)
54. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: Machine learning in Python. J Mach Learn Res 12:2825–2830
55. Buitinck L, Louppe G, Blondel M, Pedregosa F, Mueller A, Grisel O, Niculae V, Prettenhofer P, Gramfort A, Grobler J, Layton R, VanderPlas J, Joly A, Holt B, Varoquaux G (2013) API design for machine learning software: experiences from the scikit-learn project. In: ECML PKDD workshop: languages for data mining and machine learning, pp 108–122
56. Mazumder M, Banbury C, Yao X, Karlaš B, Rojas WG, Diamos S, Diamos G, He L, Kiela D, Jurado D et al (2022) Dataperf: benchmarks for data-centric ai development, arXiv preprint arXiv:2207.10062
57. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, GimelsheinN, Antiga L et al (2019) Pytorch: an imperative style, high-performance deep learning library, arXiv preprint arXiv:1912.01703
58. Paszke A, Gross S, Chintala S, Chanan G, Yang E, DeVito Z, Lin Z, Desmaison A, Antiga L, Lerer A (2017) Automatic differentiation in pytorch
59. Sergeev A, Balso MD (2018) Horovod: fast and easy distributed deep learning in TensorFlow, arXiv preprint arXiv:1802.05799
60. Clarke L, Glendinning I, Hempel R (1994) The mpi message passing interface standard, in: Programming environments for massively parallel distributed systems, Springer, pp 213–218
61. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
62. I. Anthos I., P. Stavroula J., P. Michael F., C. Dimos C., Bakas N, George M, Ashley Megan v D W (2023) Prediction of reinforced concrete column limit states using machine learning algorithm, in: COMPDYN, ECCOMAS, Athens, Greece
63. Asteris PG (2016) The fp4026 research database on the fundamental period of RC infilled frame structures. Data Brief 9:704–709
64. Markou G, Bakas NP (2021) Prediction of the shear capacity of reinforced concrete slender beams without stirrups by applying artificial intelligence algorithms in a big database of beams generated by 3d nonlinear finite element analysis. Comput Concr 28(6):533–547
65. Taljaard V, Gravett D, Mourlas C, Bakas N, Markou G, Papadrakakis M (2021) Development of a new fundamental period formula for steel structures considering the soil-structure interaction with the use of machine learning algorithms
66. van der Westhuize A, Markou G, Bakas N (2022) Development of a new fundamental period formula for steel structures considering the soil-structure interaction with the use of machine learning algorithms
67. Amani J, Moeini R (2012) Prediction of shear strength of reinforced concrete beams using adaptive neuro-fuzzy inference system and artificial neural network. Scientia Iranica 19(2):242–248
68. Pérez JL, Cladera A, Rabuñal JR, Martínez-Abella F (2012) Optimization of existing equations using a new genetic programming algorithm: application to the shear strength of reinforced concrete beams. Adv Eng Softw 50:82–96
69. Keskin RS, Arslan G (2013) Predicting diagonal cracking strength of RC slender beams without stirrups using ANNS. Comput Concr 12(5):697–715
70. Institute AC (2014) Building Code Requirements for Structural Concrete (ACI 318-14): Commentary on Building Code Requirements for Structural Concrete (ACI 318R-14): an ACI Report, American Concrete Institute. ACI
71. Institute AC (2019) Building Code Requirements for Structural Concrete (ACI 318-19): Commentary on Building Code Requirements for Structural Concrete (ACI 318R-19): an ACI Report, American Concrete Institute. ACI
72. Markou G, Papadrakakis M (2013) Computationally efficient 3D finite element modeling of RC structures. Comput Concr 12(4):443–498
73. Kotsovos MD (2015) Finite-element modelling of structural concrete: short-term static and dynamic loading conditions. CRC Press, Boca Raton
74. Willam KJ (1975) Constitutive model for the triaxial behaviour of concrete. Proc Intl Assoc Bridge Struct Eng 19:1–30
75. Menegotto M, Pinto P (1973) Method of analysis for cyclically loaded reinforced concrete plane frames including changes in geometryand non-elastic behavior of elements under combined normal force and bending, Proceedings. IABSE Sympoium on Resistance and Ultimate Deformability of Structures Acted on by Well-Defined Repeated Loads
76. Markou G, AlHamaydeh M, Saadi D (2018) Effects of the soil-structure-interaction phenomenon on RC structures with pile foundations
77. Markou G, Genco F (2019) Seismic assessment of small modular reactors: nuscale case study for the 8.8 mw earthquake in chile. Nucl Eng Des 342:176–204
78. Mourlas C, Khabele N, Bark H, Karamitros D, Taddei F, Markou G, Papadrakakis M (2020) The effect of soil-structure interaction on the nonlinear dynamic response of reinforced concrete structures. Int J Struct Stab Dyn 20(13):2041013
79. Gravett Z, Markou G (2019) State-of-the-art investigation of wind turbine structures founded on soft clay by considering the soil-foundation-structure interaction phenomenon - optimization of battered rc piles. Eng Struct 235:112013
80. Eurocode 8 (2004) Design of structures for earthquake resistance - Part 1-1: General rules and seismic action, Publications Office of the EU - European Union
81. Jiang R, Jiang L, Hu Y, Jiang L, Ye J (2020) A simplified method for fundamental period prediction of steel frames with steel plate shear walls. Struct Design Tall Spec Build 29(7):1–15. https://doi.org/10.1007/978-3-030-87312-7_38

82. ASCE-16 (2016) Minimum Design Loads for Buildings and Other Structures, The American Society of Civil Engineers
83. Cinitha A (2012) A rational approach for fundamental period of low and medium rise steel building frames. Int J Modern Eng Res 2(5):3340–3346