POLITECNICO DI TORINO

DIPARTIMENTO DI INGEGNERIA STRUTTURALE, EDILE E GEOTECNICA (DISEG)

CORSO DI LAUREA MAGISTRALE IN SISTEMI EDILIZI

TESI DI LAUREA IN SISMICA

FORECAST OF POST-DISASTER SCENARIO WITH MACHINE LEARNING APPLICATIONS: EXTENT OF DEBRIS PREDICTION AND FRAGILITY CURVES EVALUATION

RELATORE: Prof. Ing. Gian Paolo CIMELLARO

CANDIDATO:
Putignano Domenico
Mat: 266842

Anno Accademico 2019 – 2020

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 - EVALUATION OF DEBRIS EXTENTION IN A POST-DISASTER SCENARIO

1. INTRODUCTION

After a seismic event, a large amount of debris is generated from the damage of structures and it can be a critical obstacle to emergency operations and evacuations (Hirokawa and Osaragi 2016). Nevertheless, the analysis of post-disasters data can be used to reduce the social and economic losses by leading a procedure that can predict the optimal path for rescue operations and can increase the possibility for better action in terms of human security and safety.

These data, mostly in form of pictures, videos, etc., are collected after every disaster by reconnaissance teams formed by professional engineers, academic researchers, graduate students, etc. This information can be used to identify potential gaps in existing research or the applications of engineering knowledge, defining the object of further investigations to change codes and laws.

In literature, there are several studies focused on the damages caused by a large amount of debris, e.g. studies about strategies in debris management after seismic events ((Garcia et al. 2016), (Rafee et al. 2008)), but few studies have focused on Machine Learning (ML) applications to forecast the extension of building-rubble after earthquakes. The purpose of this study is to forecast the extension of the debris (EOD) and the amount of rubble, giving a prediction of the practicability of roads in an urban area after an earthquake. Input data are magnitude, distance from the epicenter, year of construction, building height, number of stories, construction materials, and direction of debris extension. This would ultimately allow civil protection agencies to plan their rescue operations while reducing the risk of getting stuck by extended rubble. The dimensions of debris extension are extracted from pictures, and ML algorithms are used to forecast the volume of debris produced. Using the methodologies of modern ML, it is possible to evaluate the accuracy of these predictions and continuously increase it adding new data that could be collected in future events.

This document is organized into eight sections: in section 2 there is a literature review of ML application in civil engineering; in section 3 there is a literature review of the used ML algorithms;

in section 4 there is a description of input dataset; in section 4 there is a dataset analysis and a description of preprocessing phase; in section 6 there are algorithm configurations used in training phase; in section 7 results of the test phase are reported for comparison among algorithms; in section 0 there are the concluding remarks.

2. MACHINE LEARNING IN CIVIL ENGINEERING

2.1. RESEARCH METHOD

The research of considered articles was done on ISI Web of Science (WoS) database, based on its importance in the scientific community and on its larger amounts of registered publications.

For the research process, WoS gives the possibility to apply several filters simultaneously; in particular, filters about (i) document type, (ii) topic, (iii) area of application, (iv) year of publication, and (v) times cited were used. Scientific journal articles and conference papers were selected as principal document types and, the initial sample of data was obtained using as topics "Machine Learning". Table 1 shows the number of obtained results applying filter about the document type and the area of application.

Table 1 – results of the first research

Database	Filter	Keyword	Results
ISI Web of Science	Topic	Machine Learning	150544
	Document type	Scientific Journal Articles	73972
	Area of application	Civil Engineering	1139
ISI Web of Science	Topic	Machine Learning	150544
	Document type	Conference Papers	73444
	Area of application	Civil Engineering	296

The elevated number of results, obtained without the filter about the application area, shows that ML has a wide application in today's scientific society, but, at the same time, the reduced number of civil engineering applications means that this field of research can have a significant increase in the future.

The first research collected several articles with a low correlation to the purpose of this work and a more detailed filter has been applied to the topic field. In particular, “Machine Learning algorithm*” AND “Machine Learning model*” AND “Machine Learning application*” has been set in the research field. This filter is used to reduce the selected articles to the ones that treat applications of ML to civil engineering, by the use of algorithms or models. Table 2 reports the number of collected works after the second research.

Table 2 – Results of the second research

Database	Filter	Keyword	Results
ISI Web of Science	Topic	Machine Learning Machine Learning Model* Machine Learning Algorithm* Machine Learning Application*	7626
	Document type	Scientific Journal Articles	4434
	Area of application	Civil Engineering	113
ISI Web of Science	Topic	Machine Learning Machine Learning Model* Machine Learning Algorithm* Machine Learning Application*	7626
	Document type	Conference Papers	3097
	Area of application	Civil Engineering	12

The number of articles decreases significantly with the described filter but, at the same time, the field of research is restricted to the only application of ML.

After this selection, other steps have been applied to individuate the most appropriate articles for a literature review.

2.1.1. Selection methodology

In this section, the selection of literature papers to be reviewed has been investigated; this procedure included a five-step process. From each step, new information about papers was obtained and used for the selection of suitable ones.

Step 1. The research on the WoS browser starts using keywords and filters described in the previous paragraph; applying filters, from the 148567 total results, only 115 documents are considered for the next steps. In 'Appendix A' all the 115 works are reported.

Step 2. Selected articles are organized for the year of publication and times cited. The former results are used to define the actual research frontier, the latter ones are used to define the most productive authors and their works.

Step 3. The detection of diffused topics is done using a Citation Network (CN) among selected works in *Step 1*. Moreover, a brief manual screening of titles and abstracts was conducted. Papers in which the keywords of research were used with a different meaning were omitted. This step gives a clear picture of treated issues.

Step 4. This step consists of an accurate reading of each selected paper, to define which are the most representative and to avoid repetitions. Features considered to the selection are the number of ML algorithms applied in each work and an exhaustive description of these methods. Some topics were omitted due to an incomplete treatment among papers. After this step, the final list of treated issues is obtained: (i) damage of structures; (ii) material compositions, (iii) electrical request of cities, and (iv) management of post-disaster. This step selects 8 documents.

Step 5. The last step consists of a bibliometric analysis of selected papers, to identify new documents about each algorithm. Articles that compare used algorithms and define positive and negative features of them were preferred. CNs among authors and cited authors are used to identify relevant researchers in the investigation field. After this step 18 new articles were selected.

Next, 2 books about ML algorithms and 5 theses about ML applications were added to paper references, chosen from the author's knowledge. In conclusion, 25 different works are considered in this review.

2.1.2. Bibliometric analysis

The literature review is a key step to analyze a new research topic and, according to Araújo (2006), the central part of bibliometrics is the quantitative method to classify scientific production.

After the first selection of articles in *Step 1*, metadata of papers were imported as text files, to be read by BibExcel (v.2016-02-20, (Persson et al. 2009)). This software can process the browser results and investigate all the bibliographic aspects. In particular, input data were converted into a dialog-format and several analyses about topics, authors, or cited papers can be done; each analysis creates an output file in *txt* format. Subsequently, the output file can be converted into '*net*' file, to obtain a CN using NetDraw (v.2.16, (Borgatti 2002)). CNs aim to understand relations among considered works and identify possible groups of connected topics or author collaboration. In this paper, relations about topics, authors, cited authors, and journals of publication were investigated. The use of CNs avoids the selection of repetitive topics and speeds up the research.

The final results of the bibliometric analysis are explained as two numeric indexes:

- the frequency obtained directly by BibExcel output; this index defines the number of repetitions of the concerned row data;
- the centrality degree of each node in a network, obtained by analysis tools in NetDraw; according to Bordin et al. (2014), this index can be defined as the number of node connections in that network.

In *Step 3*, CNs are used to identify the most important topics of selected articles. In particular, topics treated at least three times are considered. Table 3 shows the most diffused ones in the 115 selected articles.

Table 3 – Most cited topics

Frequency	Topic
35	Machine learning
11	Structural health monitoring
7	Damage detection
7	Support vector machines
7	Artificial neural networks
6	Neural networks

Frequency	Topic
5	Data mining
5	Support vector machine
5	Deep learning
4	Prediction
4	Compressive strength
4	Artificial intelligence

Subsequently, the same topics are organized into a CN to identify the connection among them.

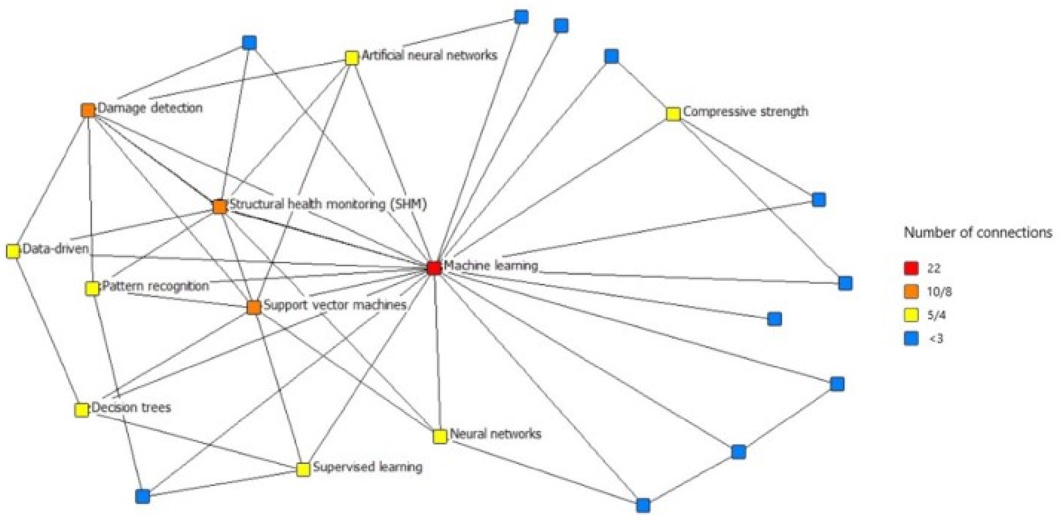


Figure 1 – Connections among most cited topics

Figure 1 shows that in civil engineering, the most diffused topics about the ML applications are the 'Structural Health Monitoring (SHM)', the 'Damage detection', and the 'Support vector machines' algorithms. However, there are additional research topics that have not been investigated sufficiently showing some gaps in the literature.

In *Step 5*, bibliometric analysis is used to define the most productive authors in the ML field and two different analyses are conducted. The former selects the authors among the 115 starting papers, and the latter identifies the most cited authors connected with the same papers. Both analyses aim to define the pillars in the research field treated. Table 4 reports the results of the first author investigation, by identifying their affiliation and area of research. Each author has three collaborations among selected papers and only Lin ZB does not belong in the civil engineering field.

Table 4 – Most productive authors

Frequency	Author	Affiliation	State	Area of research
3	Chou JS	National Taiwan University	Taiwan	Civil Engineering
3	Gonzalez I	Royal Institute of Technology	Sweden	Civil Engineering
3	Hoang ND	Duy Tan University	Vietnam	Civil Engineering
3	Karoumi R	Royal Institute of Technology	Sweden	Civil Engineering
3	Lin ZB	Nanjing University	China	Acoustic
3	Pan H	Jinggangshan University	China	Civil Engineering
3	Stevens DK	Utah State university	USA	Civil Engineering

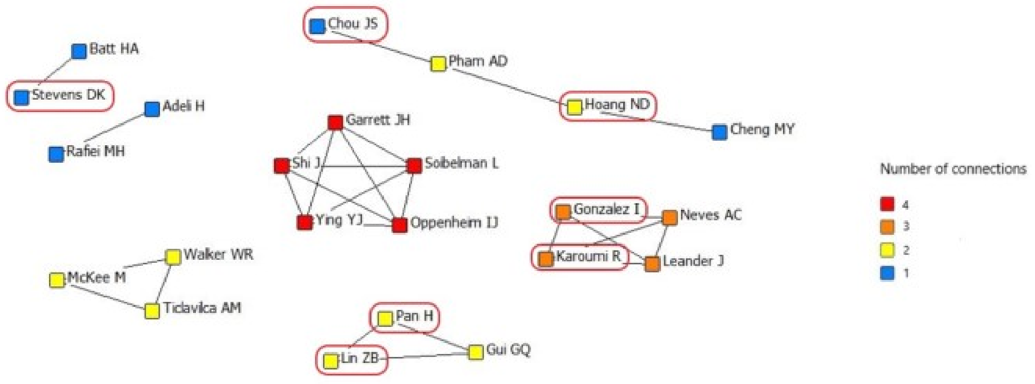


Figure 2 – Connection among most relevant authors

Figure 2 shows the collaboration net among all the authors of the 115 papers, with more than two works. This analysis confirms that the culture of a specific area influences the collaboration teams and the preferences of leadership style. In this network, seven different teams can be distinguished, connected to different nationalities of authors. In the graph, the authors present in Table 4 are circled. Table 5 reports the results of the cited author investigation. Every cited article of 115 starting papers has been considered in this analysis. The most cited author is Breiman L., a professor of UC Berkeley, with 21 different collaboration in selected articles. The author concerned his study on the statistic field and proposed the Random Forests algorithm (explained in the next section). Moreover, this analysis shows that there are several connections between civil engineering and other fields of research, as informatics, mechanical engineering, and statistic, necessary to improve the knowledge in each area.

Table 5 – Most cited authors

Frequency	Author	Affiliation	State	Area of research
21	Breiman L	UC Berkeley	USA	Statistics
19	Vapnik V.	Columbia University	USA	Statistics
16	Chou JS	National Taiwan University	Taiwan	Civil Engineering
15	Yeh IC	Tamkang University	Taiwan	Civil Engineering
15	Cheng MY	National Taiwan University	Taiwan	Civil Engineering
12	Bishop C. M.	Microsoft	UK	Informatics
12	Worden K	University of Sheffield	UK	Mechanical Engineering
11	Huang GB	Nanyang Technological University	Singapore	Electronic Engineering
10	Figueiredo E	Universidade Lusofona	Portugal	Civil Engineering
10	Kohavi R.	Microsoft	UK	Informatics
9	Yang XS	Middlesex University	UK	Civil Engineering
9	Samui P	NIT Panta	India	Geotechnical Engineering
9	Cha YJ	University of Manitoba	Canada	Civil Engineering
9	Adeli H	The Ohio State University	USA	Civil Engineering
9	Pal M	NIT Kurukshetra	India	Civil Engineering
9	Zhou J	Central South University	China	Mining Engineering
8	Deo RC	University of Southern Queensland	Australia	Artificial Inteligence
8	Friedman JH	Stanford University	USA	Statistics
8	Smola AJ	Amazon	Germany	Machine Learning
8	Rafiei MH	Johns Hopkins University	Italy	Infrastructures Engineering
8	Chau KW	Hong Kong Polytechnic University	China	Civil Engineering
8	Haykin S.	McMaster University	Canada	Cognit Syst Lab

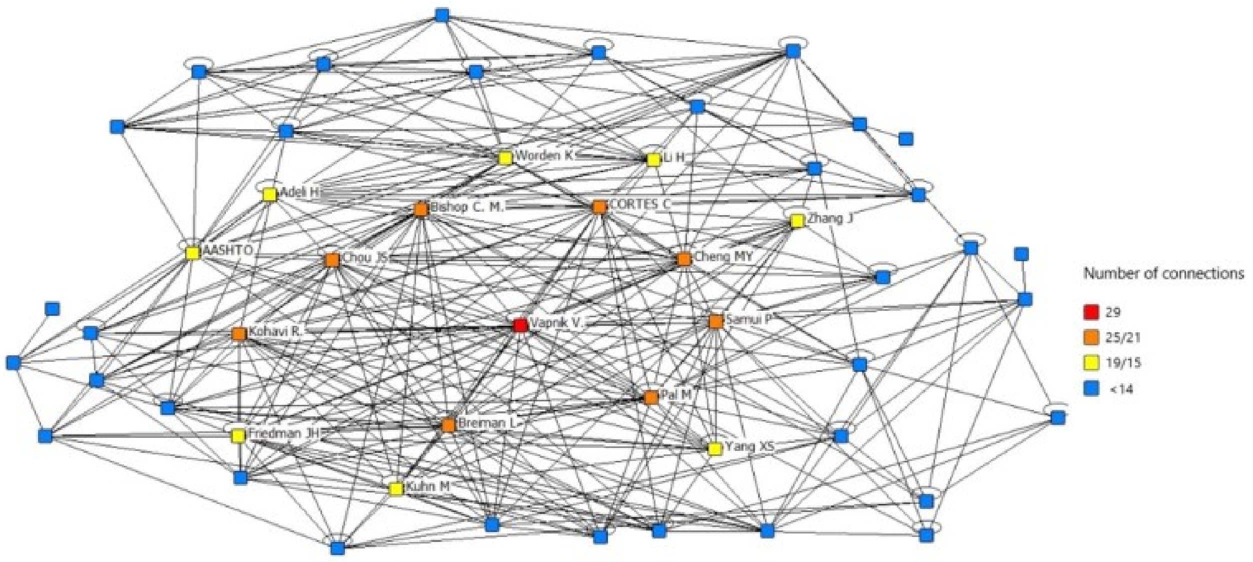


Figure 3 – Connection among most-cited authors

Figure 3 is a CN among cited authors. It results that Vapnik V. is the most cited author about the ML application on civil engineering (29 citations). His most important works are selected in this review to explain the SVM algorithm (view next section). From the analysis of cited articles, of the selected sample, results that the mean value of citations among only the 115 articles is 0,29, i.e. considered studies are not both connected and there is no awareness of other works in the field. On the contrary, the mean value of citation papers of each work in the sample is 42,1, i.e. there are lots of citations outside the considered sample.

Another bibliographic aspect, to consider for the selection of papers, is the scientific journal of publication. Using BibExcel, most diffused journals can be detected. Table 6 shows obtained results and defines the "*Journal of computing in civil engineering*" as the most relevant in civil engineering, with 41 different publications (the 36% of the entire sample of articles) and an impact factor of 2,554.

Table 6 – Frequency of scientific journals publications

Frequency	**Journals**	**Impact Factor**
41	Journal of computing in civil engineering	2,554
8	Journal of hydrologic engineering	1,438
7	Journal of bridge engineering	1,840
3	Journal of materials in civil engineering	1,984
3	Journal of construction engineering and management	2,734
3	Journal of environmental engineering	1,657
3	Journal of transportation engineering part a: systems	0,641
3	Journal of transportation engineering	1,520
3	Journal of water resources planning and management	3,404

The last analysis is done using browser tools of WoS, and it is about the trend publication of articles of ML in civil engineering. Studying all the articles of the sample, it results that the first publication was in 1996, and up to 2001, only 4 articles were published (3,45% of total). An important increase starts in 2011 with a peak in 2018, when 22 articles were published (19% of total). Figure 4 shows the trend of publication described.

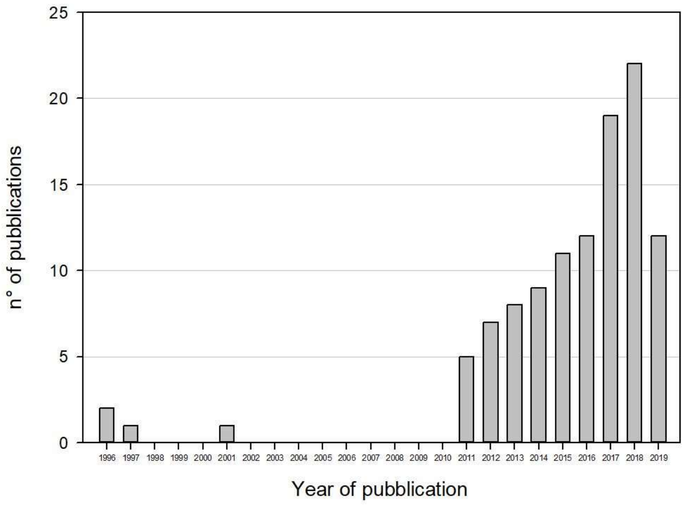


Figure 4 – Trend of publications (1996-2019)

2.1.3. Gap in literature

The bibliometric analysis on treated topics in selected articles shows that the structural field is the most diffused, and the seismic approach is preferred to others. At the same time, SHM applications can be improved by future works, focusing on the post-earthquake stability of structures and communication routes to help the rescue organizations. The application of augmented reality can be also connected to this topic.

An accurate interpretation of Figure 1 defines a range of topics that were not researched sufficiently in civil engineering and gaps in literature can be defined. Possible future works can improve the BIM approach of buildings and infrastructures, in agreement with the new design process of the last years. In this regard, is important to refine the knowledge of BIM software to obtain better results in the prediction of performance and the clash detection process. BIM software and ML can be used also to control the daily construction progress during the realization phase.

Other possible applications of ML algorithms concern the geotechnical engineering and ground detection to improve the foundation structures design. Moreover, the ability of ML to provide information can be used to forecast the needs of users in an urban area.

3. MACHINE LEARNING ALGORITHMS

Machine Learning is a computational technique with application in any field of research to improve human capacity. This field has started growing in the last ten years together with the computational power of computers. Today, ML algorithms have spread in everyday life and help human activities as detecting spam, recognizing people inside pictures, movie selection, etc. ML aims to give computers the ability to learn from input data and to make predictions, automating the decision-making processes.

In the current study, ML algorithms are used to forecast the debris extension, using post-disaster pictures dataset, that is treated with a supervised learning approach by regressor algorithms. In literature, there are several algorithms available based on different approaches and the best one to apply to each issue can be detected from general features of the input dataset and with an error comparison. The category of the algorithm to use can be detected from the graph in Figure 5, basing on the issue to treat and the dimension of the dataset.

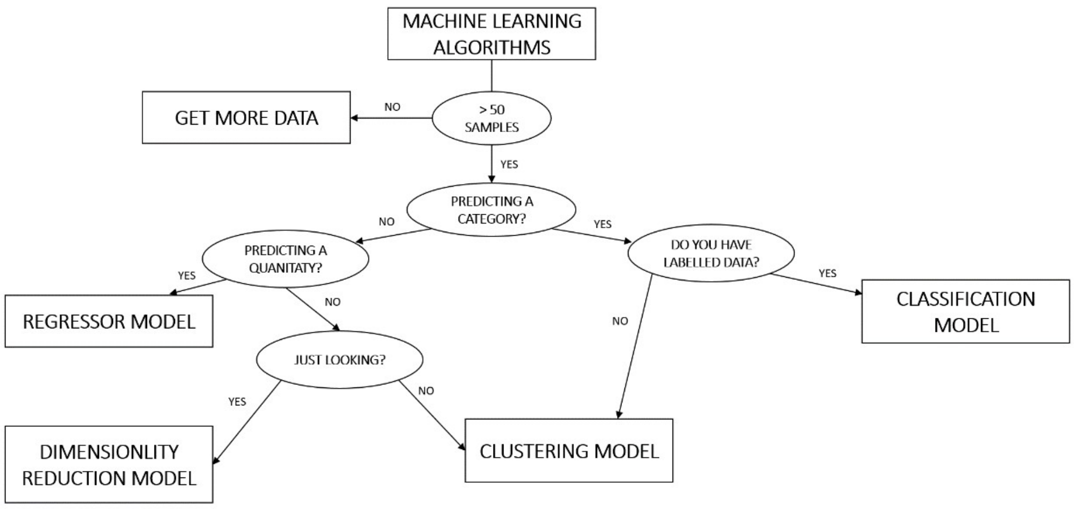


Figure 5 – Algorithm chart

In the next paragraphs, supervised learning algorithms used in this study, are presented.

3.1. k-Nearest Neighbors Regressor (KNR)

KNR algorithm aims to predict an output data point starting from the closest data in the training sample, its "nearest neighbors"; the "*k*" number defines how many points are used to evaluate the prediction (Goldberger et al. 2004).

The objective of KNR is the minimization of distances among considered "*k*" neighbors; if *d* is the evaluated distance among input data (x_i , y_i) and training sample (x_0 , $y(x_0)$), the prediction is done by the mean values of distances as in Equation (1) :

$$y(x_0) = \frac{1}{K}\sum_{i=1}^{k} d_i \tag{1}$$

According to Johannesen et al. (2019), there are four available distances: (i) Manhattan/city block distance, (ii) Euclidean distance, (iii) Minkowski distance, and (iv) Chebychev distance. The recommended set of KNR is $k \le 5$ neighbors and evaluation of distances done using Euclidean distance (Equation (2)) to avoid overfitting.

$$d(x,y) = \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2} \tag{2}$$

Figure 6 shows how the prediction can change varying the value of *k* from 1 to 3. In the first case, the target value is the same as the considered neighbor; in the latter, the prediction is given by the mean value of the 3 neighbors.

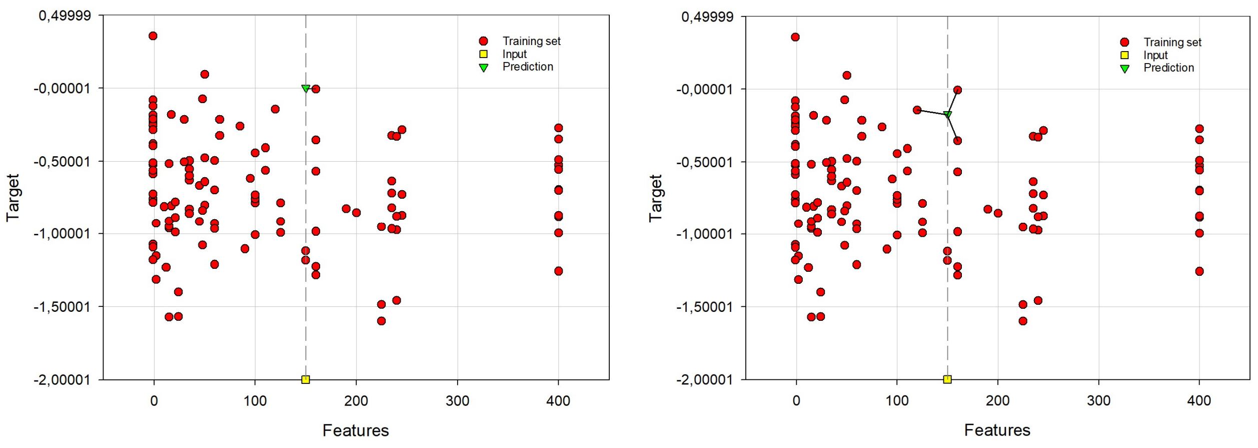


Figure 6 – KNR with k=1 (left) and KNR with k=3 (right)

The limitation of the KNR model is that with a large training sample the prediction can be slow and the results are not accurate.

3.2. Linear regression models (LM)

Linear models used as regressors, fit the input data with a straight line (Figure 7) and uses loss function for predictions. They are widely used for their straightforward and robustness and chosen when the number of predictors (*p*) is higher (Big Data); moreover, several models allow the automatic selection of features in a dataset when an unsupervised learning process is required.

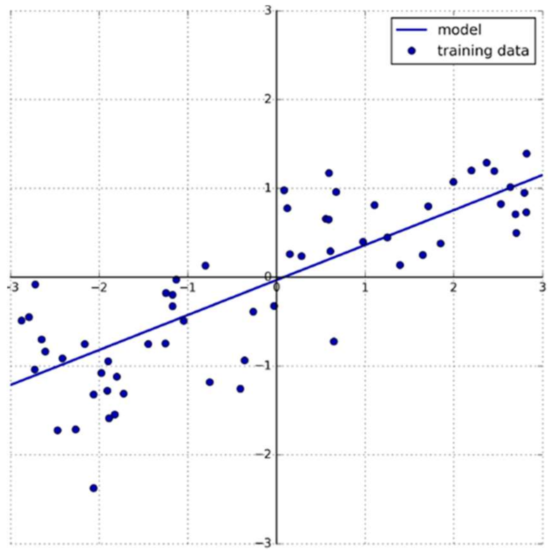


Figure 7 – Predictions of a Linear model

In literature, several linear regression models are presented; the algorithms selected for EOD detection are (i) Ridge Regression; (ii) LASSO model; (iii) Elastic Net.

3.2.1. Ridge Regression

In Ridge Regressor (Faber et al. 2017) to estimate the predictors β_j, a penalty function is added as in Equation (3):

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda_2\underbrace{\sum_{j=1}^{p}\beta_j^2}_{L_2-penalty}=RSS+\lambda_2\sum_{j=1}^{p}\beta_j^2 \qquad (3)$$

where λ_2 is a tuning parameter. If $\lambda_2=0$, there is no penalty and the model gives the same results of RSS. Instead, if $\lambda_2 \rightarrow \infty$, the penalty is high and many coefficients will be close to zero, so there will be less *variance* in the model, but higher *bias*. The research of the optimal bias-variance trade-off is defined as "continuous shrinkage". The limitations of this regression model are that the parameters can not be removed and their significance is difficult to be physically interpreted.

3.2.2. LASSO model

In the LASSO model (Tibshirani 1996) (**L**east **A**bsolute **S**hrinkage and **S**election **O**perator), the Ridge Regressor limitations are removed, so the final equation is modified as follow:

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda_1\underbrace{\sum_{j=1}^{p}|\beta_j|}_{L_1-penalty}=RSS+\lambda_1\sum_{j=1}^{p}|\beta_j| \tag{4}$$

A new L_1-penalty function, i.e. the sum of the absolute values of the predictors β_j, is included (Equation (4)). As in Ridge Regressor model, when λ_1 increases the predictors tend to zero, however because of the presence of the absolute value in the penalty function some predictors can become exactly null, and therefore they can be removed from the model. As in Ridge Regression, if $\lambda_1=0$, the L_1-penalty function disappears and the model gives the same results of RSS. Instead, if $\lambda_1 \rightarrow \infty$, all the predictors β_j are null and they are removed from the model.

In summary, the LASSO model does both continuous shrinkage and variable selection due to the L_1-penalty function, and several works in literature support this method for the extraction of the regression coefficient β_j (Knight and Fu 2000), for the stability with input data that trends to infinity (Meinshausen and Bühlmann 2006), and for the high accuracy in sparsity conditions (Zhang and Huang 2008).

Limitations of LASSO are presented by (Zou and Hastie 2005). It is less accurate when the number of predictors β_j is more than the number of observations (n) because in that case, LASSO selects at most *n* variables before it saturates. Furthermore when there is a group of predictors that are highly correlated the LASSO tends to select only one variable from the group and does not care which one is selected.

3.2.3. Elastic Net (EN)

Elastic Net (EN) is a hybrid model that linearly combines the L_1 and L_2 penalties of the Ridge Regressor and the LASSO model (Zou and Hastie 2005). The objective function that the Elastic Net regression is trying to solve is the following:

$$L(\lambda_1,\lambda_2,\beta_j) = | y_i - \beta_j x_{ij} |^2 + \lambda_2 \sum_{j=1}^{p} \beta_j^{\,2} + \lambda_1 \sum_{j=1}^{p} | \beta_j | \quad (5)$$

where the λ_1 part generates a sparse model and the λ_2 part removes the limitation on the number of selected variables and stabilizes the λ_1 regularization path.

The solution of Equation (5) is equivalent to the optimization problem reported in Equation (6) :

$$\hat{\beta} = \arg\min | y_i - \beta_j x_{ij} |^2 \quad \text{s.t.} \quad (1-\alpha)\sum_{j=1}^{p} | \beta_j | + \alpha \sum_{j=1}^{p} \beta_j^2 \le t \ \text{ for some } t \qquad \text{with } \alpha = \frac{\lambda_2}{\lambda_1 + \lambda_2} \quad (6)$$

For α=1 and α=0, EN becomes respectively a Ridge Regression and a LASSO model. When $0 < \alpha < 1$ EN has characteristics of both models and avoids LASSO limitations, generating a reduced model by predictors removal.

3.3. Decision Trees (DT)

Decision Trees (DT) is an algorithm based on a decision graph in which predictions are based on "tests", i.e. sequences of if/else questions that build a tree. Tests are chosen over all possible questions about the treated issue and those which predict high information are selected (Chongchong et al. 2018). Each test aims to divide input data into two subsets, called "node"; in a tree structure the former node is called "root" and the latter ones are called "leaves". If/else questions that connect a root node to a leaf are called "branch". Figure 8 shows the growth of a tree.

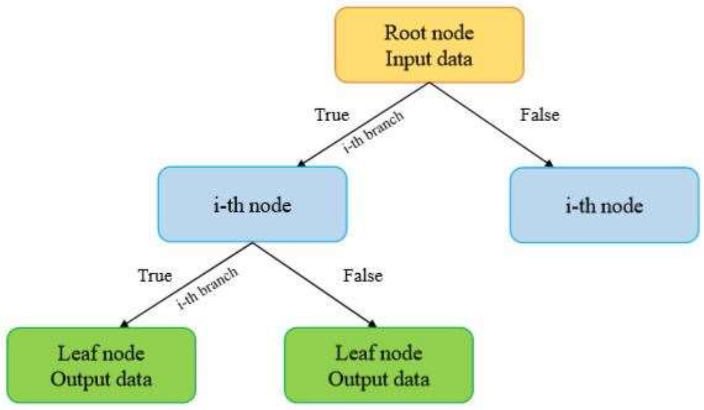


Figure 8 – DT algorithm

The growth of a tree will continue until a node contains a group of homogeneous data. With uncontrolled growth, the DT model tends to overfit and for this reason, a pruning strategy can be chosen from two possible approaches:

- stopping the growth of the tree early (also called "pre-pruning");
- removing or collapsing nodes that contain little information (also called "post-pruning" or just pruning)

An implementation of DT algorithm is presented by Di Girolamo et al. (2020). In their work, authors introduced the idea of Classification and Regression Tree (CART), i.e. a method that, starting from the growing of a tree, defines a dynamical model of the problem to do predictions. The first step concerns in the creation of a tree (T_j) for each prediction required; then, a model as Equation (7) is associated with each leaf l_{ij}:

$$x(k+j) = A'_{ij}\, x(k) + \sum_{\alpha=1}^{j} B'_{ij,\alpha}\, u(k+\alpha-1) + f'_{ij} \tag{7}$$

where $A'_{ij}; B'_{ij,\alpha}$ and f'_{ij} is defined with least square methodology. The last step consists in the derivation of a dynamical model, obtained with an extension of the state-space of input and variables. DT has two advantages over the other algorithms: the resulting model can be visualized easily and understood by non-experts, and the algorithms are completely invariant to the scaling of the data. The weakness of this method lies in the risk of overfitting data providing poor generalization, despite the use of pre-pruning or post-pruning. Therefore, in most applications, both pruning approaches are used.

3.4. Random Forests (RF)

Random Forests (RF) is a combination of tree predictors where each tree growth depends on a random vector and follows the DT growing rules (Breiman 2001). If *x* is the input data and Θ_k is the random vector of the root nodes of each tree in the forest, the output prediction is the $h(x, \Theta_k)$ function. The result of the RF model is the average over *k* predictions of trees.

Defined *X, Y* as random vectors that randomize the training set, $E_{X,Y}(Y - h(X,\Theta_k)^2$ the mean-squared generalization error for any function $h(x, \Theta_k)$, *PE*(tree)* the average generalization error of a single tree, and *PE*(forest)* the average generalization error of the forest, it can be demonstrated that (Theorem 11.2 (Breiman 2001)):

$$PE^*(forest) \leq \bar{\rho} PE^*(tree) \tag{8}$$

where $\bar{\rho}$ is the weighted correlation between the residual $Y - h(X,\Theta_i)$ and $Y - h(X,\Theta_j)$.

Equation (8) shows that RF decreases the error of a single Decision Tree, and the overfitting is reduced as more trees are added to the RF model. Furthermore, the distortion of the output by noisy input data is less than 5%.

3.5. Support Vector Regression (SVR)

SVR (Smola 1996) is a regression model based on the Support Vector Machine (SVM) algorithm presented by Vapnik (1995), (1998). In the SVM, the prediction is based on the identification of a hyperplane fitted to the input data and the evaluation of distances among input points and the plane. A generic formulation of the hyperplane is reported in Equation (9) :

$$f(x,\alpha) = (w \cdot x) + b \tag{9}$$

where *w* and *b* are parameters induced from the training sample, *α* is the Lagrangian multipliers used to define the optimization of *w* and *b*, *x* is the input data, and *f(x,α)* is the set of real function that contains the regression function *f(x,α₀)*. The hyperplane divides input data space into regions that identify different classes. Figure 9 shows the hyperplane function in a training set with a single variable.

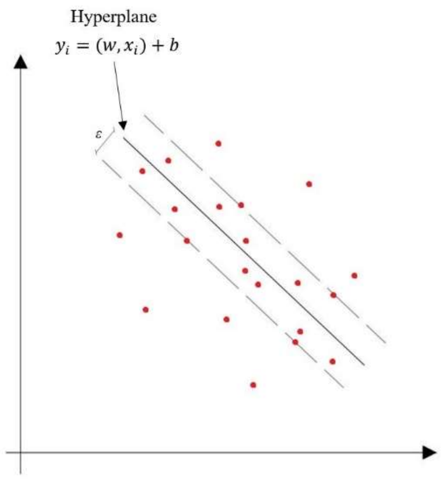


Figure 9 – SVR algorithm

SVR algorithm is obtained from the introduction of the distance measurement between input and prediction. SVR aims to find a regression function that has at most a fixed deviation among target and inputs (ε) and it is as flat as possible. The regression function is given by the minimization of the empirical risk as shown in Equation (10):

$$\Phi(w,\xi^*,\xi) = \frac{\|w\|^2}{2} + C\left(\sum_{i=1}^{j}\xi_i^* + \sum_{i=1}^{j}\xi_i\right) \tag{10}$$

where *C* is a fixed value, and ξ^*, ξ are slack variables representing upper and lower constraint of the *j* outputs. This method results useful to a pixel defragmentation analysis of images (Dibike et al. 2001).

3.6. Artificial Neural Network (ANN)

According to Bilal et al. (2016), ANN defines a wide group of ML algorithms based on a network of connected neurons that output a prediction applying several filters to input features. Generally, in each ANN algorithm, networks can be divided into three groups of layers: (i) the input layer, where the input features are collocated, (ii) hidden layers that contained neurons (functions), and (iii) the output layer. Neurons are weights function obtained from a random process or a backward propagation and their formulation changes for each algorithm.

3.6.1. Multilayer Perceptron Regressor (MLP)

A multilayer perceptron (MLP) is an Artificial Neural Networks (ANN) method in which neurons are called 'perceptrons' and have the formulation reported in Equation (11):

$$f(x) = w_2 \cdot g(w_1^T x + b_1) + b_2 \tag{11}$$

where w_1 and w_2 are the weights, b_1 and b_2 are the bias, and $g(x)$ is the activation function ((Khadem and Hossein-Zadeh 2014),(Qi et al. 2018)).

The network structure is composed of an input layer, two hidden layers, and an output layer as reported in Figure 10.

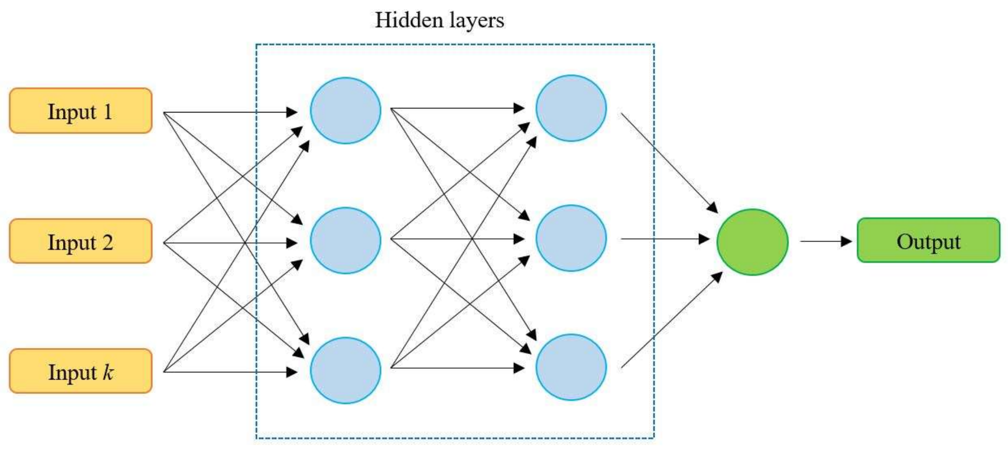


Figure 10 – MLP structure

Neurons try to simulate the same procedures that occur in the human brain. Each processing unit estimates a weighted sum of its inputs and uses an activation function to extract the single output value. The process starts with random initialization of weights that are continuously updated by a backward propagation in the neurons. The iterative process starts with the definition of the error on the output value:

$$e_j(n) = d_j(n) - y_j(n) \tag{12}$$

where *d* is the target values, and *y* is the perceptrons prediction. Minimizing the error in Equation (12) by a cost function (Equation (13)) and applying the Gradient Descent in Equation (14), a cycle of iteration is completed.

$$\varepsilon(n) = \frac{1}{2}\sum_j e_j^2(n) \tag{13}$$

$$y_{i+1} = y_i - \eta \frac{\partial \varepsilon(n)}{\partial v_j(n)} \tag{14}$$

where η is the learning rate and v_j is a local variable. The end of the process is set by a maximum number of iterations or by a specific cutting point.

4. DATA COLLECTION

4.1. Data sources

The first part of the study involved collecting around 10.000 pictures from four different sources:

- The Earthquake Engineering Institute (EERI);
- Datacenterhub.org;
- Db.concretecoalition.org;
- The Geotechnical Extreme Events Reconnaissance (GEER).

To apply the ML algorithms, it was necessary to manually select pictures in which the extent of the rubble was visible and could be measured by comparison with other known elements appearing in the pictures. Following this approach, a final database of 310 pictures from 25 different earthquakes events was created; each picture shows at least a portion of the building that was damaged or collapsed and the extent of debris. This database was used to train and test the ML algorithm. The 25 different earthquakes covered and the number of pictures selected by each of them is shown in Table 7.

Table 7 – Number of pictures of events

Earthquake	Year	No. of Pictures
Central Italy	2009	65
Cephalonia (Greece)	2011	22
Christchurch (New Zealand)	2011	13
Ecuador	2016	54
India	2001	28
Loma Pietra (US)	1989	5
Mexico Central	2017	20
Nepal	2015	34
Northern Iran	2017	1
Northridge (US)	1994	2
North-west Armenia	1988	5
South Napa Valley (US)	2015	6
Southern Taiwan	2016	24
Turkey	1999	10
Alaska	1964	3
Algeria	1980	1
Armenia	1988	1
California	1994	4
Chile	2010	4
Haiti	2010	3
Honduras	2009	1
Indonesia	2005	1
Japan	2012	1
Korea	2017	1
Oklahoma (USA)	2016	1

4.2. Dataset features

The learning process of ML algorithms starts from an input dataset made up of several data samples identified by features and an output value; learning consists in the extraction of weights of features needed to predict given outputs. The accuracy and the dimension of the dataset influence the prediction results. After a careful analysis, the following attributes have been identified as they are closely connected to the amount of debris made by each building:

- *Material*: the buildings considered in this study are made of reinforced concrete or masonry and the behavior of the two materials is different in term of debris extension;

- *Stories*: : seismic forces applied to buildings and damage are dependent from the number of slabs of the building;
- *Year of construction*: construction technologies and methodologies can vary during the years because of both different standards and construction techniques;
- *Magnitude*: the stronger the ground motion, the higher the possibilities to damage buildings;
- *Distance from epicenter*: the smaller the distance among buildings and epicenter, the greater the effects on structures;
- *Building height*: tallest buildings could produce a bigger amount of rubble,
- *Direction of EOD*: for each image, vertical or horizontal EOD can be evaluated.

A valuation of dataset composition can be done analyzing the feature correlation with the EOD of the samples in the dataset. Table 8 and Figure 11 shows the results obtained with a Random Forest validation process.

Table 8 – Correlation value of features

Features	**Correlation**
Height of building	0.417
Epicenter distance	0.208
Magnitude	0.118
Stories of building	0.094
Year of construction	0.081
Direction of evaluation	0.046
Material	0.036

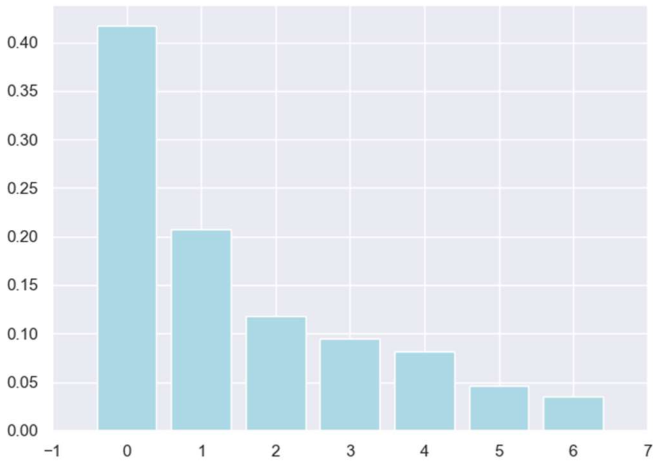


Figure 11 – Correlation chart

4.3. Debris extension

The main purpose of this study is to forecast the EOD after earthquakes based on different parameters of the structures. Unfortunately, this information is not present in the reports that are usually written after each seismic event. From each picture, a dimension whose size is well known is identified and compared with debris extension; the comparison was done using PhotoFilter, a photo editor, evaluating real distances from pixel coordinates. The sample id.061 test is reported; in Figure 12 the width of a car (P) is used as a reference and the dimension of debris (p) is evaluated.

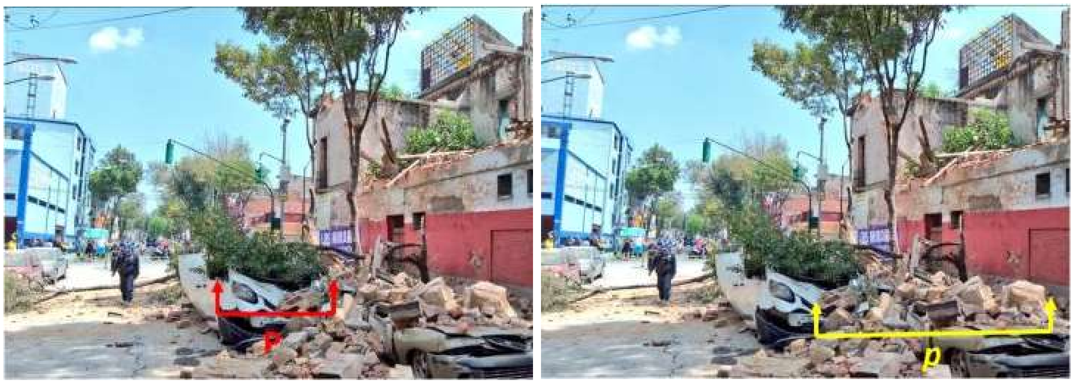


Figure 12 – Depth (P) of a car and picture dimension (p) of debris

To predict the EOD along one axis the proportion in Equation (15) is used:

$$d = p \times \frac{P}{D} \tag{15}$$

where:

- $P = x_{1r} - x_{2r}$ with x_{1r} and x_{2r} pixel coordinates of the reference element;
- $p = x_{1d} - x_{2d}$ with x_{1d} and x_{2d} pixel coordinates of the debris extension;
- D is the value in meters of the principal measure of the reference element;
- d is the value in meters of the debris extension.

The parameters used to evaluate the rubble shown in Figure 12 are summarized in Table 9.

Table 9 – input and output data in sample id.061

Coefficient	**Value**
x_{1r}	502
x_{2r}	334
x_{1d}	720
x_{2d}	371
p	349
P	168
D (m)	1.8
d (m)	3,629

To process the obtained EOD for each picture, results are normalized to the height of the building and their log_{10} value is considered. This post-processing is done to:

- make the training data less sensitive to the scale of the features (e.g the height of a building heavily influences the amount of rubble);
- minimize the variance of the dataset;
- compare results with other models available in the literature;
- make the optimization well-conditioned: most of the ML optimizations are solved using Gradient Descent and the speed of convergency depends on the scaling of the features.

Unfortunately, it is not possible to collect all the parameters from each picture due to the lack of information. Therefore, Table 10 is summarized how many samples (pictures) provide information for each feature.

Table 10 – Number of samples for features

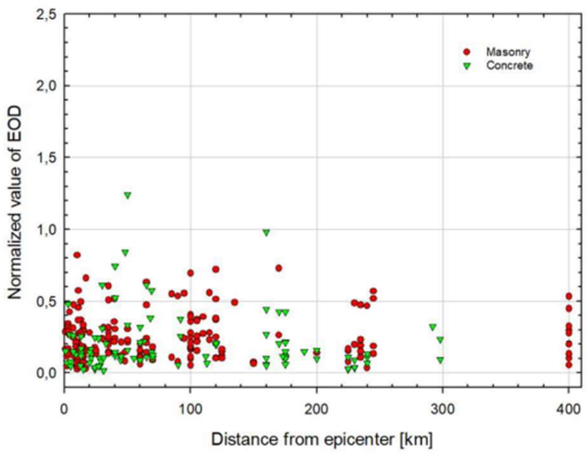
Feature	**# samples**
Material	310
Stories	310
Year of construction	66
Magnitude	310
Distance from epicenter	278
Height	310
Direction of EOD	310

Undefined values of features are replaced with the medium value of the considering features; this is done to reduce dispersion in ML algorithm based on distances among data as Linear models, KNR, and SVR.

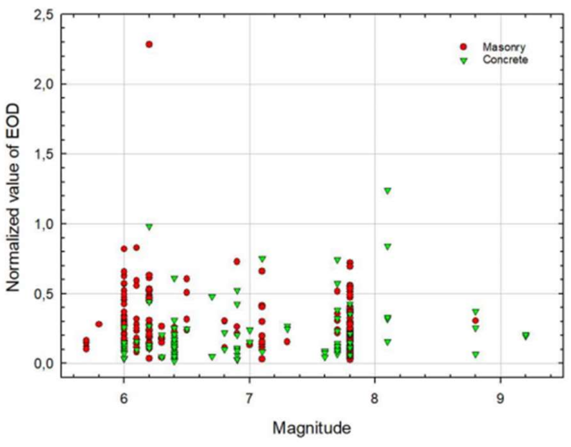
5. DATA VISUALIZATION AND PREPROCESSING

In this paragraph, data are divided according to the input features of the dataset. In each graph on the y-axis is reported the EOD value and on the x-axis is reported the corresponding feature. Moreover, masonry and concrete EOD results are divided using labels.

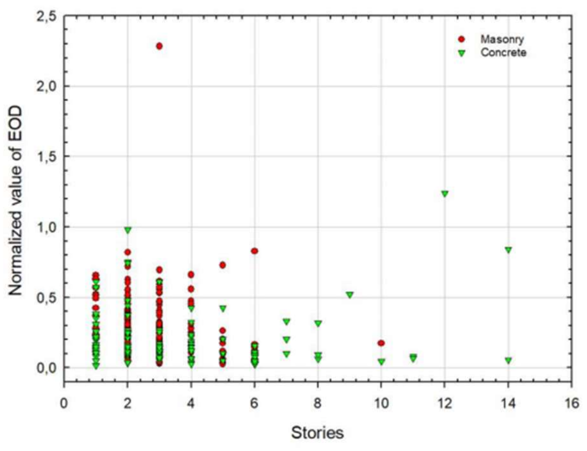
Figure 13 reported the EOD value for respectively: (a) the epicenter distance, (b) the magnitude, (c) the stories of the building, (d) the year of construction, and (e) the building heigh. The direction of EOD evaluation is not reported in these graphs because it is not representative of the issue but is used to improve the accuracy of algorithms.



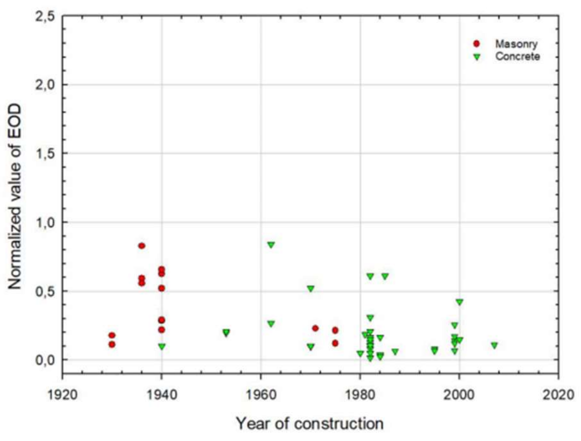
(a)



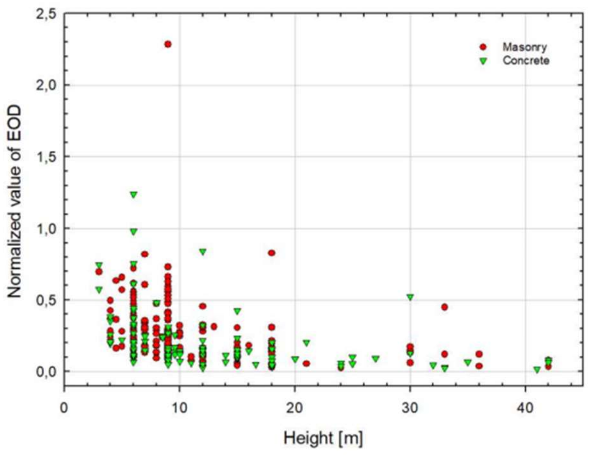
(b)



(c)



(d)



(e)

Figure 13 – Features charts: (a) the epicenter distance, (b) the magnitude, (c) the stories of buildings, (d) the year of construction, (e) the building heigh

The distance from the epicenter (a) and the magnitude of the considered earthquake (b) are the features that could be identified more easily, so the data size is the largest; the stories of buildings (c) shows that the dataset mostly concentrates on buildings between 1 and 6 stories; the year of construction (d) was most difficult to identify because few earthquake reports reported this information, so the data size corresponding to this feature is the smallest; the building height (e) can be obtained from reports or the number of stories. To improve the accuracy of predictions, a logarithmic distribution of building heights is considered.

5.1. t-Distributed Stochastic Neighbour Embedding (t-SNE)

The issue analyzed in this study is a multi-features regression in 7 dimensions that can not be visualized in a 3D or 2D space. The visualization of the data distribution among features may induce a validation from the user, based on the sparsity of data. In particular, a sparse dataset may produce a well generalized ML model to be applied at different samples for predictions. To avoid this limitation and to have an optimization of interpretation of predictions, the t-SNE tools can be used (Van der Maaten and Hinton 2008).

The t-SNE method aims to create a map of analyzed data, where distances among points in the real space ("*t*" dimensional) are respected, i.e. data points are grouped with their neighbor data, so different groups are identified. This model applies two times a probability distribution of distances among points, using Barnes-Hut approximation, and minimize the Kullback-Leibler divergence by Gradient Descent. Figure 14 shows the application of t-SNE to the dataset of the current case study. Looking at the graph it is possible to say that the dataset can be divided into five different categories of samples; moreover, the R-squared value hardly could have a high score because for the same y-value on the graph we have more than one data on the x-axis.

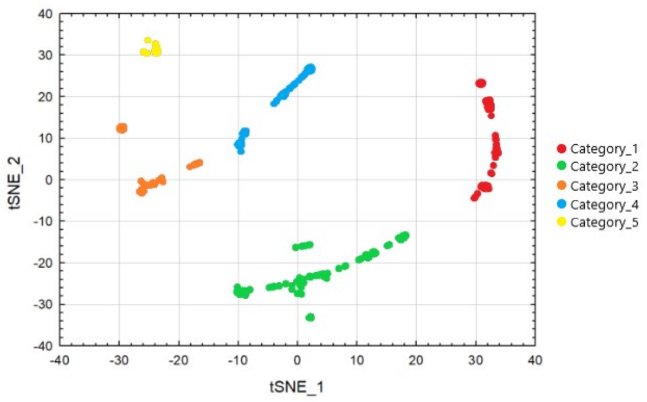


Figure 14 – t-SNE chart

5.2. Preprocessing Data

Overfitting, or high variance, is when an algorithm, which is used to fit a training set, can have high accuracy for the train set but a lower one for the predictions. In other words, overfitting occurs when the algorithm may accurately fit the training set but fail to generalize to new examples. For each algorithm, there are different pre-processing approaches to the dataset that can help to avoid overfitting or other prediction error. In particular, for distance-based methods, scaler processing can be used. For Python implementation, Scikit Learn proposes several possibilities and in this work, the following are used:

- StandardScaler: this pre-processing ensures that for each feature the mean value is 0 and the variance is 1, bringing all the features to the same magnitude. In this way it is possible to avoid outliers point that could create problems for the accuracy;
- MinMaxScaler: this procedure shifts the data such that all features are exactly between 0 and 1. For a two-dimensional dataset, this means all data is contained within the maximum and the minimum value of the feature.

6. DATA TESTING

In this chapter, the algorithm tuning and the comparing models to find the best ML algorithm are reported.

For each different model, several parameters or coefficients are needed to tune the algorithm to obtain the best result. In the next paragraphs, the parameter necessary to find the best fit regression are described. For further information, it is recommended to consult the documentation of Scikit-learn (2018) since in the concerned study only the parameters and the options of used algorithms are described. A self-tuning process has been used for the definition of the best parameters set for each algorithm; a wide range of values has been trained for each parameter and the set with the lowest MSE value has been chosen. Moreover, 100 random splits between training and test set have been considered for three size ratio (85-15%, 80-20%, and 70-30%). Finally, the set with higher accuracy has been chosen for each algorithm.

Next paragraphs reports the best set of parameters individuated.

6.1. k-Nearest Neighbour Regressor (KNR)

For the KNR algorithm, the parameters to tune are two:

- the *k-value* of considered neighbors, i.e. the number of points taken into consideration to predictions for the *X* test;
- the distance evaluation.

The best set is: size-ratio=85%-15%, *k=6,* and distance= ‘manhattan’.

6.2. Linear Models

For Ridge Regression and LASSO model, the only parameter to tune is α, i.e. the regularization strength of respectively *L_2-penalty* and *L_1-penalty* coefficient. In Elastic Net, there is also the *l1_ratio* parameter used to define the ratio between λ_1 and λ_2. The best set for each algorithm is obtained from a size-ratio of 85%-15% and parameters are:

- Ridge Regression: α=10
- LASSO: α=0,001
- ElasticNet: α= 0,01 and *l1_ratio*=0,5

6.3. Decision Trees (DT)

The coefficients used for DT are 4:

- *min_samples_split*: gives the minimum number of samples required to split an internal node;
- *min_samples_leaf*: is the minimum number of samples required to be at a leaf node;
- *max_depth:* the maximum depth of a tree;
- *max_features:* the maximum number of features to consider when looking for the best split.

The best set is: size-ratio=80%-20%, *min_samples_split*=20, *min_samples_leaf*=2, *max_depth*=5, and *max_features*=2.

6.4. Random Forest (RF)

The coefficients used for RF are 4:

- *min_samples_split*: gives the minimum number of samples required to split an internal node;
- *min_samples_leaf*: is the minimum number of samples required to be at a leaf node;
- *max_depth:* the maximum depth of a tree;
- *max_features:* the maximum number of features to consider when looking for the best split.

The best set is: size-ratio=85%-15%, *min_samples_split*=5, *min_samples_leaf*=2, *max_depth*=6, and *max_features*=2.

6.5. SVR

The tuned parameters for SVR are 4:

- *C*: specifies the strength of the regularization;
- *ε:* specifies the epsilon-tube within which no penalty is associated with the loss function;
- *kernel*: specifies the kernel type used in the algorithm;
- *gamma*: a kernel coefficient;
- *degree:* specifies the degree of the polynomial kernel function only with a 'poly' kernel;

The best set is: size-ratio=85%-15%, *C*=1000, *ε*=0,1, *kernel*=linear, *gamma*='scale', and *degree*=2.

6.6. MLP regressor

ANN required several parameters to tune due to their nature of the multilayer connective framework. In the concerned study the parameters set are 7:

- *alpha*: is the regularization parameter;
- *hidden_layer_sizes*: is the number of neurons in the i-th hidden layer;

- *activation*: defines the activation function for the hidden layer;
- *solver:* defines the solver for weight optimization;
- *max_iter:* the maximum number of iteration.

The best set is: size-ratio=80%-20%, *alpha*=0,001, *hidden_layer_sizes*=10, *activation*='relu', *solver*='lbfgs', and *max_iter*=350.

6.7. Error evaluation

6.7.1. R-squared value

The R-squared value, also known as the "coefficient of correlation", evaluates how much the scatter points are distant from the regression fit line calculated by the algorithm, providing a measure of the future prediction accuracy of new samples. The range of variation of R-squared is [0;1], where 1 is the best possible score and 0 is the worst, corresponding to a model that always predicts the *y* value, disregarding the input features. In this study, the R-squared value can be negative since the model is predicting descending values while the true data are growing and absolute values of R-squared are used for the comparison.

Defined $\widehat{y_i}$ as the predicted value of the i-th sample and y_i as the corresponding true value, the R-squared (R^2) coefficient of correlation estimated over $n_{samples}$ has the formulation in Equation (16).

$$R^2\left(y,\widehat{y}\right)=1-\frac{\sum_{i=0}^{n_{samples}-1}\left(y_i-\widehat{y_i}\right)^2}{\sum_{i=0}^{n_{samples}-1}\left(y_i-\overline{y}\right)^2} \tag{16}$$

where:

$$\overline{y}=\frac{1}{n_{samples}}\sum_{i=0}^{n_{samples}-1}y_i \tag{17}$$

6.7.2. Mean Squared Error (MSE)

The MSE of an estimator measures the average of the squares errors, i.e. the average squared difference between the predicted values and the input values. It is used to measure the quality of an estimator since it evaluates both the variance and the bias. MSE is always a non-negative and values close to zero correspond to high accuracy.

Defined $\widehat{y_i}$ as the predicted value of the i-th sample and y_i as the corresponding true value, MSE over $n_{samples}$ is estimated in Equation (18).

$$MSE\left(y,\hat{y}\right)=\frac{1}{n_{samples}}\sum_{i=0}^{n_{samples}-1}(y_i-\hat{y}_i)^2 \tag{18}$$

7. RESULTS

In this chapter, obtained results of R-squared value and MSE of each algorithm are reported in column graphs and comparison among methods is done. In particular, different bar colors are used to different test size:

- Blue bars refer to the score with a test size data about 15% of the dataset;
- Yellow bars refer to the score with a test size data about 20% of the dataset;
- Red bars refer to the score with a test size data about 30% of the dataset.

As explained in the previous paragraph, each algorithm is run with three different size-ratio of train/test set, with 100 random sample partitions, and with a self-tuning process of parameters.

The results here reported for the comparison are the best ones for each algorithm.

The MSE value is used for the first comparison and Figure 15 and Figure 16 show obtained results respectively for the training set and the test set.

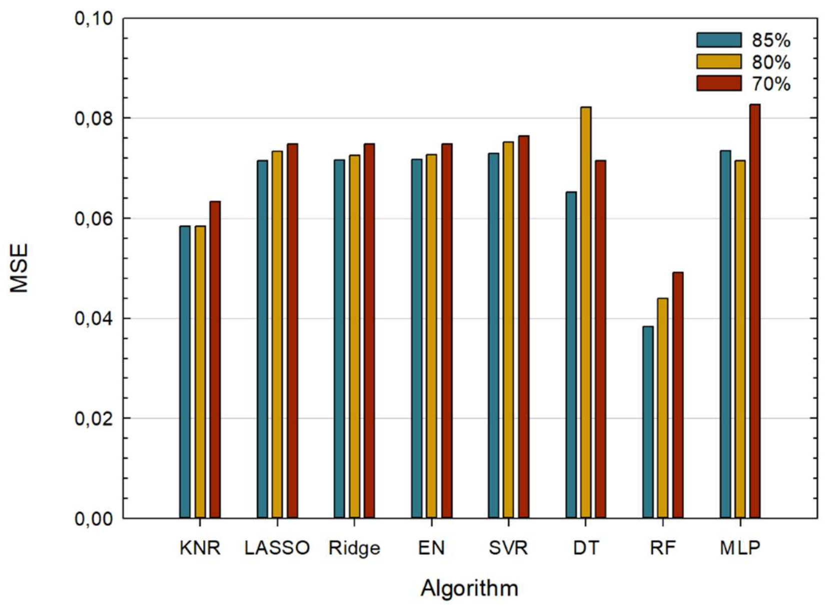


Figure 15 – MSE comparison on the train set

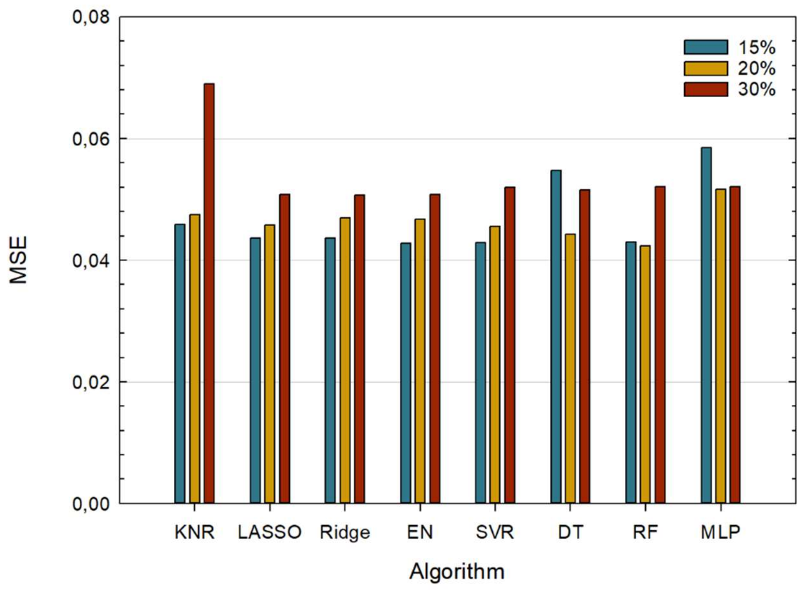


Figure 16 – MSE comparison on the test set

In previous charts, it may be seen that the linear regression models and the SVR gives similar results for both training and test set; moreover, for KNR, linear models, SVR and RF, the lower the training set size, the higher the MSE value. DT and MLP has a higher error respectively for the 80-20% and 30-70% sizes. RF has the lower MSE value, with a mean value of 0,044.

According to this comparison, Randon Forest is the most accurate ML algorithm for the treated dataset to predict the EOD and a 85-15% partition of the input dataset has been chosen for its lower error evaluation.

Figure 17 shows the R-squared value of this configuration.

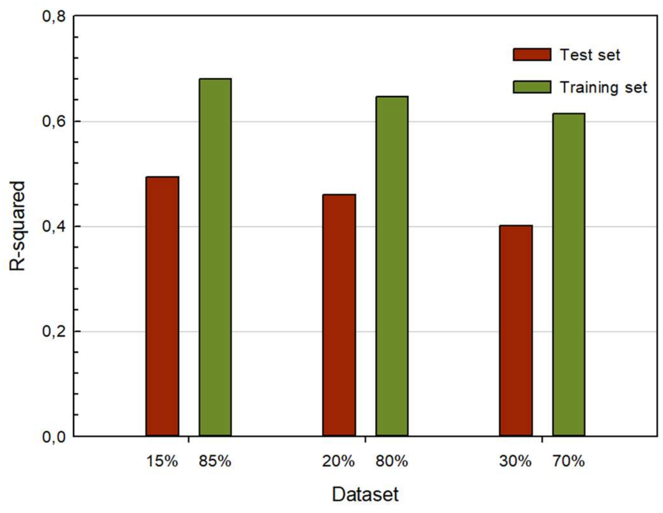


Figure 17 – R-squared value of RF

The low difference between test and training confirms that RF has great accuracy in the prediction of EOD and that the parameter configuration selected avoids overfitting and underfitting. In Table 11 the numeric evaluation of the accuracy is reported; red values are the best results for R-squared and MSE evaluations.

Table 11 – R-squared and MSE scores

Algorithm		**Training set**			**Test set**		
		15%	**20%**	**30%**	**15%**	**20%**	**30%**
KNR	R^2	0,525	0,531	0,427	0,382	0,395	0,453
	MSE	0,058	0,058	0,063	0,046	0,048	0,069
LASSO	R^2	0,410	0,412	0,398	0,487	0,417	0,461
	MSE	0,072	0,073	0,075	0,044	0,046	0,051
Elastic Net	R^2	0,408	0,417	0,398	0,497	0,405	0,460
	MSE	0,072	0,073	0,075	0,043	0,047	0,051
Ridge Regression	R^2	0,409	0,417	0,398	0,487	0,403	0,462
	MSE	0,072	0,073	0,075	0,044	0,047	0,051
Decision Trees	R^2	0,445	0,355	0,391	0,475	0,357	0,529
	MSE	0,065	0,082	0072	0,055	0,044	0,052
Random Forest	R^2	0,679	0,647	0,614	0,493	0,459	0,402
	MSE	0,038	0,044	0,049	0,043	0,042	0,052
SVR	R^2	0,398	0,395	0,384	0,495	0,419	0,448
	MSE	0,073	0,075	0,077	0,043	0,046	0,052
MLP	R^2	0,357	0,394	0,372	0,514	0,518	0,309
	MSE	0,074	0,072	0,083	0,059	0,052	0,052

8. CONCLUDING REMARKS

The objective of this study is to assess the extension of debris (EOD) from pictures, caused by earthquake damage on civil structures, using ML algorithms. The starting input data is composed of 310 pictures of post-disaster and related data, divided into 7 features, from 25 different earthquakes. Eight ML algorithms are trained and their performances in terms of EOD prediction are evaluated using R-squared and Mean Squared Error (MSE).

During the training process, some parameters, different for each algorithm, are altered and tested to achieve the best score.

Looking at the results, the main features of the present research can be summarized as follows:

- RF gives the best MSE score on both training and test set and the best R-squared value; the R-squared evaluation shows that the algorithm does not suffer from overfitting; it is defined as the most accurate ML algorithm for the EOD evaluation with the used dataset;
- Linear models and SVR gives are less accurate than RF;
- KNR decreases significantly its accuracy as the dimension of the training set is reduced;
- MLP results not appropriate to the treated dataset since it is not enough large.

In conclusion, Random forest is the best choice for predicting the EOD and the performances suggest that there are some potential uses of ML methodologies in this field. It is appropriate to underline that collecting more pictures and data from other Earthquakes in the future could significantly improve results.

CHAPTER 2 - OBJECT DETECTION FOR THE EXTRACTION OF STRUCTURAL FEATURES

9. INTRODUCTION

Object detection is a relevant machine learning application that improves image processing during every computer vision research. The main aim is to identify relevant objects in images or videos and classify them with labels. Recent applications make possible a real-time detection of videos that improve automation as robot vision and autonomous driving. Moreover, object detection is used to improve security on the roads or to boost the productivity of industries.

Most common applications are based on supervised learning and start from a dataset of images that contains specific categories of objects to detect. For each image in the dataset, the location of objects is defined with a rectangular box or with a 'mask'.

Object detection may be used to detect a specific instance, i.e. human faces, buildings, or several categories that include the most common natural and artificial objects.

Datasets can be realized manually by the user or taken from open sources on the net, i.e. COCO dataset (Lin et al. 2014) and OpenImage Dataset (Kuznetsova et al. 2018). After the training phase, the algorithm can detect the selected categories of objects in images and/or videos, and the corresponding locations. Today, object detection is a field in continuous development and new methods and algorithms are a challenging task. There exist several applications that merge different machine learning algorithms to obtain an efficient training phase and feature extraction on the dataset. Convolutional Neural Networks (CNN), Random Forest (RF), Support Vector Machine (SVM), K-Nearest Neighbor (KNR), and Linear models (LM) are the most commonly used algorithms.

With post-processing of detection output, information is extracted from images and may be used in computer analysis. This work aims to define an automatic extraction process of features of building façade, to evaluate fragility curves of masonry structures; in particular, the opening ratio and the location of windows or doors will be detected. The post-processing may define an equivalent frame model (EFM) to realize a FEM representation of the structure that will be used to evaluate the capacity curve and the fragility curves of the building. The main issue is the identification of a standard object detection method to extract opening locations from different points of view in the images. Several object detection methods have been tested and the more accurate is identified for the detection phase. This document is organized in four sections: in section 10 there is a literature review of the most common object detection methods and error evaluation approaches; in section 11 there is the individuation of the most accurate method and the feature extraction with the post-processing phase; in section 12 there is a description of fragility curves evaluation and the EFM definition; in section 13 there are the concluding remarks.

10. OBJECT DETECTION

In the last ten years, several object detection methods have been published basing on machine learning algorithms described in section 3. In particular, CNN, RF, SVM, KNR, and LM are used. Each algorithm has been applied with different architecture to obtain a faster and more accurate method for each particular issue.

Nguyen et al. (2020) published a state of the art of the most relevant methods for object detection of the last seven years.

10.1. CNN Models

The most commonly used category of algorithms is the CNN; depending on the number of connected layers, CNN returns greater accuracy but, at the same time, needs more computational resources and evaluating time. Layers of each CNN can be classified into four categories:

- *Convolutional layer (CONV)*: is the main layer and it is composed of kernels (filters) that contain learnable weights for the feature extraction. Its depth is the same as the input layer (i.e. treating RGB images, depth is 3), but its height and width are smaller than input. CONV layers slide on the image to find features and the accuracy and the computational cost depend on the stride of the sliding.

- Max pooling layer: is the layer that performs downsampling by a reduction of input size. It works sliding a filter on the input and reduces the computational cost and the probability of overfitting.
- Fully connected layer (FC): is the layer that connects previous layers to all neurons. It is a vector that adds bias to input for neurons.
- Softmax layer: is the layer that predicts the class object of a region proposal. It evaluates the probability of each region to be an object and output the class with the highest probability.

10.2. Detection Methods

Using different algorithms of machine learning, several object detection methods have been developed. In general, methods are divided into two classes:

- *Two-stage frameworks*: the detection of objects is done in two-step; the first is the extraction of a region of interest (RoI), i.e. the possible locations of the object in the image, the second is the classification of each RoI;
- *One-stage frameworks*: the detection of objects occurs in one step; each method learn bounding-box coordinates and class probability from image pixels.

The 'two-stage frameworks' have better accuracy than 'one-stage frameworks', but they require a high computational cost and time-machine for the training and the evaluation phases. For this reason, 'one-stage frameworks' are used for real-time object detection.

Each method is based on an architecture model that defines the connection between layers or other method's settings.

10.2.1. Architecture model

A pioneering work of this field is the AlexNet architecture (Krizhevsky et al. 2012): it consists of five CONV layers and three FC layers that work using a 11x11,5x5 and 3x3 kernels. ZF-Net (Zeiler and Fergus 2014) is an improvement of AlexNet; it is composed of 13 layers divided into CONV, LRN, FC, and softmax layers. CONV layers are followed by a ReLU activation function. This model uses a 7x7 kernels. To avoid the problems about needed resources and time, ResNet (He et al. 2016) introduces the ‘residual block’, an architecture that uses skip-connections between layers, and DenseNet (Huang et al. 2018) proposes the ‘dense block’, which consists in layers densely connected where the input of a layer is the output of all previous ones.

10.2.2. Two-stage frameworks

The first step of each ‘two-stage framework’ method is the extraction of RoIs. A ‘Region of Interest’ (RoI) is a sample within a dataset identified for a particular purpose. For the object detection issue, an RoI is defined on a 2D dataset and identifies the boundaries of an object. RoIs are extracted from a feature map and must be processed to become bounding boxes. A region can be defined by a vector of 4 integers in two ways:

- two coordinates of one corner, width, and height of the box;
- two coordinates of two opposite corners of the box.

To classify an RoI as positive, i.e. to consider the proposal region as an object to be classified among labels, an IoU ratio must be fixed. The ‘Intersection over Union’ (IoU) is an index that measures the overlap between two boundaries. In the object detection, IoU is used to evaluate the precision of the bounding-box predictions as it is described in Figure 18 and Equation (19).

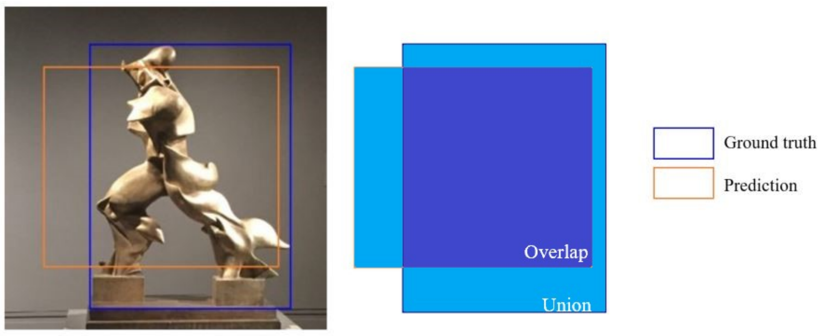


Figure 18 – IoU definition adapted

$$IoU = \frac{area \quad of \quad overlap}{area \quad of \quad union} \tag{19}$$

Below, relevant ‘two-stage frameworks’ are reported.

10.2.2.1. R-CNN

R-CNN (Girshick et al. 2014) is a method based on a CNN end-to-end to classify a region proposal into an object or background. As it is only a classifier, its accuracy depends on the previous region proposal module. Selective Search or Edge Box are adopted for this issue. The CNNs are used to extract high-level features and the classification phase is done by several SVM algorithms. R-CNN requires 2000 region proposals (RoI) for every image to be analyzed.

10.2.2.2. Fast R-CNN

Fast R-CNN (Girshick 2015) was introduced to avoid limitations of R-CNN, in particular, to reduce the computational time. This method uses CNNs to create a feature map from the input image that is overlaid with the RoI. With a max-pooling layer a fixed-size vector of features is extracted from each RoI and location and classes of objects are evaluated by an FC layer and two softmax layers. Equation (20) defines the multi-task loss *L* used on each RoI to train classification and bounding-box regression.

$$L(p,u,t^u,v) = L_{cls}(p,u) + \lambda[u \geq 1]L_{loc}(t^u,v) \tag{20}$$

where *p* is the probability distribution (per RoI), *u* is the ground-truth class (per RoI), *v* is the ground-truth bounding-box regression target, and L_{cls} and L_{loc} are defined respectively in Equation (21) and Equation (22).

$$L_{cls}(p,u) = -\log(p_u) \tag{21}$$

$$L_{loc}(t^u,v) = \sum_{i \in \{x,y,w,h\}} smooth_{L_1}(t_i^u - v_i) \quad with \quad smooth_{L_1}(x) = \begin{cases} 0.5x^2 & if\ |x| < 1 \\ |x| - 0.5 & otherwise \end{cases} \tag{22}$$

In the method presented by Girshick, the minimum value of IoU to label an RoI as positive is 0.5.

10.2.2.3. Faster R-CNN

Faster R-CNN, presented by Ren et al. (2017), is an implementation of Fast R-CNN. The main aim is to improve both the accuracy and speed of previous methods, sharing convolutional features among RoI individuation and classification phases. Faster R-CNN is composed of two modules: the first is the Region Proposal Networks (RPN) and the second is the Fast R-CNN.

RPN is a fully-connected convolutional network with images as input and RoI and objectness scores as outputs. Objectness measures the probability of a region to be an object versus background. Generally, RoI has a rectangular shape, but RPN can treats also several shapes. To extract RoIs, RPN generates a feature map using a CNN and slides a CONV layer on this map; a rectified linear unit (ReLU) activation function is used to increase speed and convergence of this process. Each sliding window (CONV layer) generates a different vector, fed into a regression (*reg*) and softmax (*cls*) layers that evaluate the coordinates of RoIs and the probability of being object into the box. Figure 19 shows the RPN architecture.

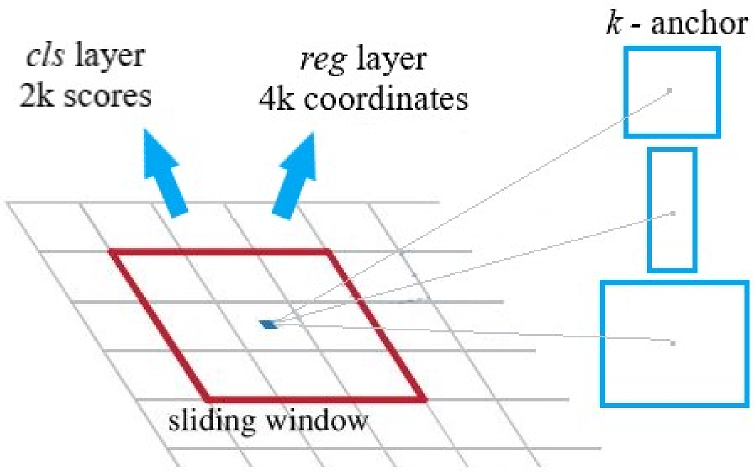


Figure 19 – RPN architecture

Each sliding-window location can predict a *k* number of region proposals that are called *anchors*. This means that the *reg layer* has *4k* outputs (i.e. the coordinates of *k* boxes), and the *cls layer* has *2k* outputs (i.e. the probability to be an object or background). The authors proposed to use *k*=9, to evaluate 3 different scales and 3 different aspect ratio. Each anchor in the sliding window can be labeled as positive, negative, or neither positive nor negative: for this classification IoU approach is used. According to the authors, an anchor is labeled as positive (object) if its IoU ratio is the highest or if it is greater than 0.7, as negative (background) if its IoU ratio is less than 0.3; when the anchor is neither positive nor negative, it does not contribute to the training.

After the individuation, RoIs are shared between RPN and Fast R-CNN using CONV layers. In particular, the model of '*Alternating Training*' is adopted from the authors. This model consists of an iteration process of four steps:

- Step 1: training of an RPN network starting from pre-trained weights or random weights;
- Step 2: training of a Fast R-CNN network using RoI obtained in step 1;
- Step 3: initializing RPN network training, fixing the shared CONV layer, and tuning only the layers unique to RPN;
- Step 4: a unified network has been formed and it can be run for more iterations.

The unified network obtained from *Alternating Training* is the unit-based of the Faster-RCNN. A schematic architecture is reported in Figure 20.

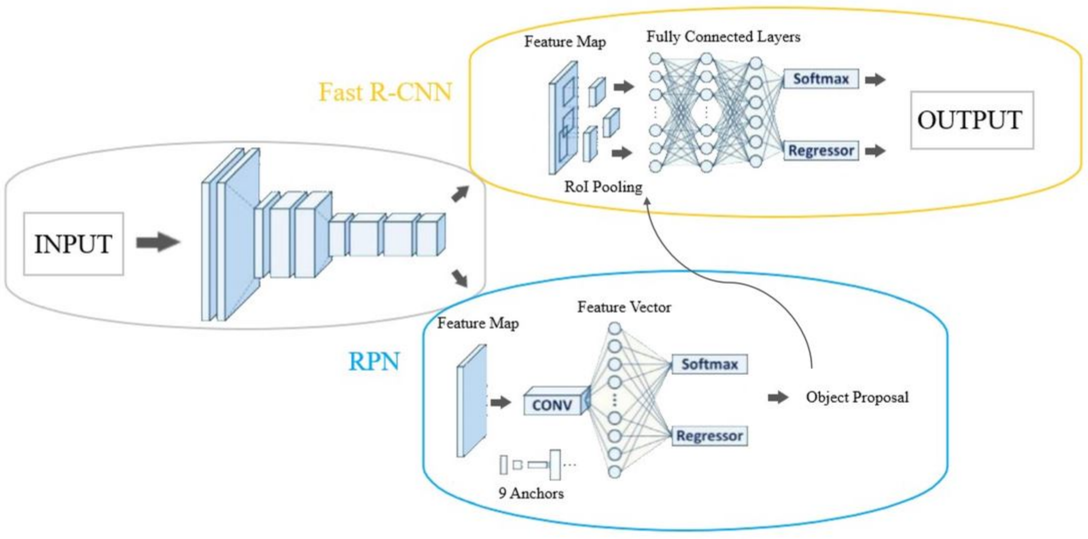


Figure 20 – Faster-RCNN architecture

10.2.2.4. Mask R-CNN

Mask R-CNN (He et al. 2017) is an implementation of Faster R-CNN, that introduced the object instance segmentation. The main aim is to classify each pixel of images into object categories, adding a third output at the Faster R-CNN. To each RoI, a new Fully Convolution Network (FCN) is applied that works in parallel with the *cls layer* and outputs a *mask.* A *mask* is a *m* x *m* matrix build on each positive RoI that identifies the real boundaries and shape of each object in the box; in the Mask R-CNN, this matrix has not resorted to an FC layer like other object segmentation methods. The pixel-to-pixel process of classification works with a feature map for each RoI and needs an alignment with the pixel position in the image. Usually, the feature map is extracted from a RoIPool layer that *quantizes* a floating-number RoI, divides each RoI into spatial bins which are themselves quantized, and aggregates features to each bin. Quantization generates a misalignment between pixels and, to avoid this problem, the authors introduced the '*RoIAlign layer*'. This layer removes the quantization process and uses bilinear interpolation to evaluate features at four sampled locations in each RoI bin. During training, the multi-task loss function in Equation (23) is used on each RoI.

$$L = L_{cls} + L_{box} + L_{mask} \tag{23}$$

where L_{mask} is the average binary cross-entropy loss.

10.2.2.5. SNIPER

Scale **N**ormalization for **I**mage **P**yramids with **E**fficient **R**esampling (Singh et al. 2018) is a pixel-based method that generates *chips* in the image of *k* x *k* pixels at equal intervals of *d* pixels, at multiple scales. The image is also resized to each scale. Positive chips cover all the positive instances in the images and the background portions are not covered. Possible positive chips are selected from the region proposal pre-obtained.

10.2.3. One-stage frameworks

10.2.3.1. YOLO

The YOLO (**Y**ou **O**nly **L**ook **O**nce) family methods were the first that introduced a unique end-to-end CNN for the bounding-box detection and the classification of objects. Every YOLO method is composed of three main parts:

- Backbone: a CNN that creates a feature map at different scales;
- Neck: layers that combine features for detection;
- Head: layers that predict boxes and object labels.

The difference between versions is in the architecture that combines these parts.

YOLOv3 (Redmon and Farhadi 2018) is a method based on a CNN algorithm that predicts, for each image, a 3D tensor divided into N x N grid cells and was written using the *Darknet* framework. Boxes are identified as in YOLO9000, i.e. by four coordinates for each box (t_x, t_y, t_w, t_h), and e an objectness score evaluate the precision of box prediction. Those with a score under a fixed threshold are ignored by the algorithm; usually, the threshold is set to 0.5.

For the classification, YOLOv3 uses a multilabel classification to predicts the possible classes of the bounding-box. The feature extractor is a hybrid method between Darknet-19 (used in YOLOv2) and modern network stuff. This network is called Darknet-53 and is made up of 53 CONV layers connected by shortcut connections. YOLOv3 uses 3 different scales during detection, obtained by downsampling input image dimension by 32, 16, and 8; this process improves the classification accuracy of small objects.

YOLOv5 (Jocher et al. 2020) is the last update of YOLO family methods, published on the 25th of June 2020. This version avoids some problems of the previous version YOLOv3 and YOLOv4 and today it is considered the leader in real-time object detection. YOLOv5 is the first YOLO method with a naïve release written in PyTorch, that makes support and deployment easier than the Darknet framework. In this version, the developers added a process of 'Data augmentation' based on scaling, color space adjustment, and mosaic augmentation of dataset images. Mosaic augmentation is the real innovation of YOLOv5, because improves the detection accuracy, helping the method to identify small objects generally not well detected.

Another improvement is the '*Auto learning bounding boxes anchors*' process: in the previous versions, boxes are predicted from a fixed list of anchor dimensions; with YOLOv5 the anchor detection is supported by *K-means* or *genetic algorithms* that adapted boxes to the input dataset. Moreover, YOLOv5 uses a CSP Backbone to extract image features. This model is based on a DenseNet, which connects layers of a CNN to bolster feature propagation and reduce the number of network parameters. In particular, CSP ResNext50 and CSP Darknet53 have been used for YOLOv5. The main consequence of this approach is the boosting of inference speed.

10.2.3.2. RetinaNet

RetinaNet (Lin et al. 2002) is a one-stage method that uses two different techniques for object detection: Feature Pyramid Network (FPN) backbone and focal loss. FPN backbone is adopted from *ResNet* architecture and is used to extract the feature map combining low-resolution, high-resolution, and semi-weak characteristics; focal loss improves the accuracy of detection changing weights in the loss function, adding the modulating factor $(1-p_i)^\gamma$. The resulting formulation of loss is reported in Equation (15):

$$\sum_{i=1}^{k}(y_i \log(p_i)(1-p_i)^\gamma) + (1-y_i)\log(1-p_i)p_i^\gamma) \qquad (24)$$

Figure 21 describes a scheme of the RetinaNet architecture, showing two subnetworks attached to FPN, the first for the classification of objects, and the second for the regression of boxes.

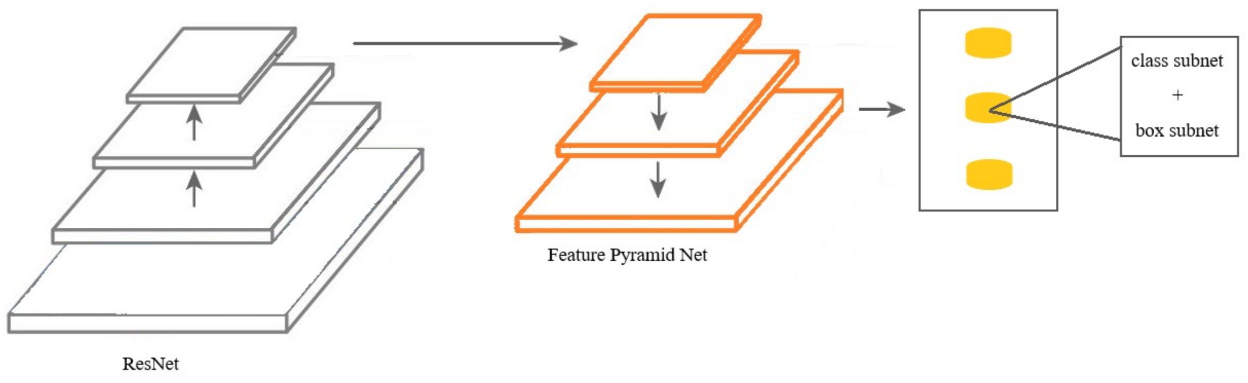


Figure 21 – RetinaNet architecture

10.2.3.3. SSD

Single-**S**hot **D**etector (Liu et al. 2020) is a high-speed detector usually used for real-time object detection. It is based on a VGG-16 network, made up of 16 CONV layers, concatenated with 3 FC layers and a softmax layer that return detection results. In the SSD, each CONV layer detects objects using a certain scale with *k* anchors and outputs *4k* offsets and *c* objectness scores; detection is done using a *m* x *n* features map. The entire process needs *kmn(c+4)* filters.

In the training process, the loss function is defined as a sum of the localization loss (L_{loc}) and the confidence loss (L_{conf}) as in Equation (25):

$$L(x,c,l,g) = \frac{1}{N}(L_{conf}(x,c) + \alpha L_{loc}(x,l,g)) \qquad (25)$$

where x is the box index, c is the class confidence, l is the predicted box parameter, and g is the ground truth box parameter. In the SSD, the L_{loc} is a Smooth L1 loss between the predicted box and ground truth box, and the L_{conf} is the softmax loss over multiple class confidences.

10.2.3.4. CenterNet

CenterNet (Zhou et al. 2019) is an anchor-free method based on a single CNN without FPN. To localize an object, CenterNet considers the center point of the bounding box and regresses other properties from image features. Using a keypoint estimator $\hat{Y}$, it can be obtained a heatmap, offset value $\hat{O}$, and size value $\hat{S}$; center points are defined as the peak points of the heatmap, and bounding boxes are evaluated as reported in Equation (26):

$$\left(\hat{x}_i + \delta\hat{x}_i - \hat{w}_i / 2, \quad \hat{y}_i + \delta\hat{y}_i - \hat{h}_i / 2, \quad \hat{x}_i + \delta\hat{x}_i + \hat{w}_i / 2, \quad \hat{y}_i + \delta\hat{y}_i + \hat{h}_i / 2\right) \tag{26}$$

where $P_c = (\hat{x}_i, \hat{y}_i)$ are the detected center point of class *c*, $\hat{O}_{\hat{x}_i, \hat{y}_i} = (\delta\hat{x}_i, \delta\hat{y}_i)$ is the offset prediction, and $\hat{S}_{\hat{x}_i, \hat{y}_i} = (\hat{w}_i, \hat{h}_i)$ is the size prediction.

The detection phase estimates a 3D bounding-box for each object and requires three additional attributes for each center point: depth, 3D dimensions, and orientation. In particular, depth *d* is a scalar obtained from the L1 loss function, 3D dimensions are three scalars, and orientation is a single scalar. According to the authors, CenterNet can be tested using several network architecture like ResNet-101, ResNet-18, Deep Layer Aggregation-34 (DLA-34), and Hourglass-104 which returns the best result.

10.2.3.5. RCNet

RCNet (Zhang et al. 2016) is a method based on a random CNet architecture that incorporates a Gradient Boosting Machine (GBM) into the framework. A CNet is a multilayer architecture in which the features extraction is done using three kinds of layers:

- *CONV layer*: a layer that connects the input feature map with the output feature map using filters;
- *Nonlinearity layer*: a pointwise function applied to each component of the feature map. Usually, this layer consists of a ReLU function;
- *Pooling layer*: a layer that involves executing a max operation over the activation function.

After the features extraction, a back-propagation of a soft-max loss function is applied to the network; the goal of the process is to minimize *J(θ)* function in Equation (27):

$$J(\theta)=\sum_{i=1}^{N}\left(\frac{1}{2}\|h(x_i,\theta)-y\|^2\right)+\lambda\sum_{l}^{L}sum(\|\theta^{(l)}\|^2) \tag{27}$$

where the target output *y* is a *1-of-K* vector with *K* number of outputs, *L* is the number of layers, *l* is the index of the layer number, $h(x_i,\theta)$ is the CNet function, θ is the learned parameter, λ is a regularization term, and *N* is the number of the training sample.

In the CNet architecture, each feature extraction stage can be considered as a complete learning process and all the extracted parameters are shared between stages; this approach generates overfitting. The innovation of the RCNet consists in the introduction of a filter bank ($W_{sfilter}$) and a random selection of a single filter (W_{filter}) from each stage.

10.2.4. Evaluation of error

After the training phase, the validation phase occurs to evaluate the accuracy of the features extraction and the prediction of labels and bounding-boxes. With this error evaluation, methods can be compared and the more accurate can be individuated for a particular dataset.

The first evaluation of method accuracy is obtained from the value of the used loss function; in particular, the lower the value of the loss, the higher the accuracy of the method. The loss function evaluates the difference between the true values and the estimated values and it considers several factors as the quality of the dataset, the number of features, and the accuracy of features extraction. Monitoring of the loss function value during the training phase can help the user to avoid overfitting or to stop the iteration process when it does not converge.

According to Hui (2018), there are other two popular error indexes used for object detection: '*Average Precision*' (AP) and '*mean Average Precision*' (mAP). Both indexes are based on the concept of '*precision*' and '*recall*':

- *'precision' (p)* measures the percentage of correct prediction;
- *'recall' (r)* measures the capacity to find the true positive on *K* total prediction.

Both indexes are over 0 to 1; below mathematical formulations are reported:

$$Precision = \frac{TP}{TP + FP} \tag{28}$$

$$Recall = \frac{TP}{TP + FN} \tag{29}$$

with:

TP= True Positive

TN= True Negative

FP= False Positive

FN= False Negative

In the evaluation of these indexes on bounding-boxes, it must be set a minimum value of IoU to define true and false predictions; generally, this value is $IoU \geq 0.5$.

Average Precision (AP_v) is defined as the area under the precision-recall curve and its value is between 0 and 1 as *p* and *r*. Equation (30) shows the mathematical formulation of AP:

$$AP_v = \int_0^1 p(r)dr \tag{30}$$

where *v* is the minimum IoU for positive prediction expresses as a percentage.

Often precision-recall curve is smoothed, replacing each precision value for recall $\hat{r}$ with the maximum precision value of any recall $\geq \hat{r}$ as in Equation (31).

$$p_{\text{int}erp}(r) = \max_{\hat{r} \geq r} p(\hat{r}) \tag{31}$$

After this process, the curve will decrease monotonically with a zig-zag pattern. In Figure 22 the orange curve is the original relation and the green curve is the result of the smoothing process.

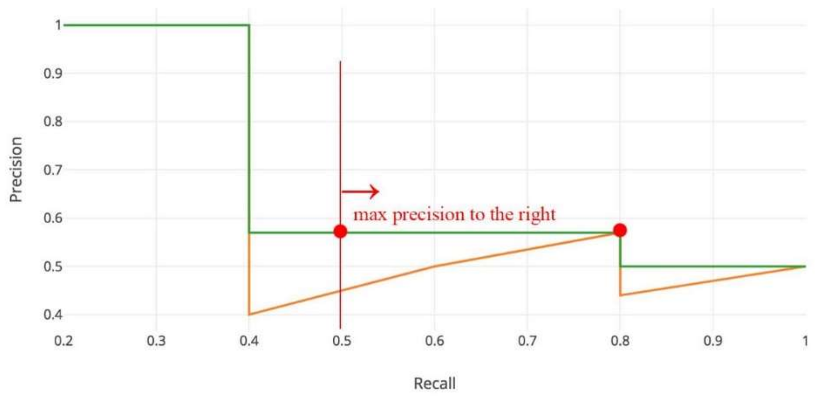


Figure 22 – Precision-Recall curve

The smoothing process gives an AP value less suspectable to small variations of precision.

Mean Average Precision (mAP) is the definition of AP proposed by COCO dataset developer. It is evaluated as the average of AP for IoU from 0.5 to 0.95 with a step size of 0.05 (AP@[.50:.05:.95]).

10.3. Object Detection in Civil Engineering

Object detection is a field of computer vision-based used in civil engineering for the prevention and evaluation of risk.

In the last ten years, the relevant works about object detection can be divided into 4 categories:

- Structural Health Monitoring (SHM);
- Detection of the existing structure;
- Seismic field;
- Street object detection.

10.3.1. Structural Health Monitoring

The SHM is a field of civil engineering that monitors the performance of an existing structure and detect possible damages.

Cha et al. (2018) proposed in their work detection of damage on concrete and steel structure. In particular, five types of damages have been labeled: (i) concrete crack, (ii) steel corrosion medium, (iii) steel corrosion high, (iv) bolt corrosion, and (v) steel delamination. A Faster R-CNN method has been used to detection on a dataset of 2366 images, with 4695 objects, collected from the authors. A data augmentation process has been used to expand the dataset. Training and detection phases have been performed using a Core i7-6700k @4 CPU with GeForce GTX 1080 GPU. The same method has been used for real-time detection of damage.

Li et al. (2019) presented a pixel-level damage detection method of concrete structures based on an FCN detector. This method predicts the pixel location of cracks or damages, using an FCN build on a DenseNet-121 architecture (Huang et al. 2018). Detected damages are labeled in four categories: (i) crack, (ii) spalling, (iii) efflorescence, and (v) hole. To realize this work, the authors used a dataset of 2750 images obtained after a data augmentation process. Training and detection have been performed in Linux system on a Intel Xeon CPU E5-2630 v4 @2.2 CPU with ASUS GeForce GTX 1080 Ti GPU.

Liang (2019) proposed an image-based approach for post-disaster inspection of reinforced concrete structures, divided into three detection levels. The first is the '*system-level failure analysis*', which aims to define the existence of a system of a major failure in the image. This level is a binary classification task tackled with a CNN method on a dataset of 492 images. The VGG-16 (Simonyan and Zisserman 2014) architecture has been selected for this step. The second level is the '*structural component-level detection*' with which structural components are detected in images. The Faster-RCNN method has been applied to a dataset of 236 images for this issue. The third level is the *'local-level damage localization'* and it is applied only to images with detected failure in the first level. Damage detection occurs to visual changes in the structural components, such as delamination/spall area, cracking, and exposes rebar. Semantic pixel-wise segmentation is used to identify the real area of damage, using the CNN method presented by Badrinarayanan et al. (2017). A dataset of 436 images is used. All training and detection steps are conducted on three different computers: a Core i7-6700HQ @2.6 CPU with GeForce GTX 1080 GPU, a Core i7-8700 @3.2 CPU with GeForce GTX 1080 GPU, and a Core i7-8700k @3.7 CPU with GeForce GTX 1080 GPU.

10.3.2. Detection of the existing structure

Image-vision based method can be used to improve the urban design by detection of buildings or infrastructures already existing. Proposing methods in literature have the aim to extract automatically urban objects from satellite images and to use them for landscape classification.

Inglada (2007) presented an urban detection method based on geometric features of objects. Eleven classes have been detected: isolated buildings (IB), paths and tracks (PT), crossroads (CR), bridges (BR), wide roads (WR), highways (HW), round-abouts (RA), narrow roads (NR), railways (RW), and suburbs (SB). The training dataset is made up of more than 150 images for each class; each image is 100x100 pixels with the object in the center. In this work, the geometric features used are the region boundaries and the alignments, both extracted from the *Gestalt* approach (Desolneux et al. 2000); the classification phase has been executed with an SVM algorithm applied to a features vector.

Aljumaily et al. (2017) proposed an urban detection method based on a 'light detection and ranging' (LiDAR) dataset applied to a 'digital surface model' (DSM), i.e. the dataset is composed of billion 3D coordinates of points, a timestamp, an intensity measurement, and an RGB color indicator for each point. The method is divided into three steps. The first is a MapReduce grid-based partitioning that divided the area of interest into smaller subspaces to reduce time machine during training. The second step consists of the detection of dense cubes and sparse cubes into each subspace; a dense cube is a 3D area in which points are grouped together that involved in forming part of the ground, roads, of buildings; a sparse cube is a 3D area in which points are dispersed and identify vegetation. The third step consists of a clustering process on dense cubes; sparse cubes are removed from the analysis and dense cubes are grouped into classes. For this method, a DBSCAN algorithm has been used (Xu and Tian 2015) on an Intel Core i7 @2.4 CPU.

10.3.3. Seismic field

In the seismic field, image detection has been used for the prediction of vibration frequency by Liu et al. (2019). This application is a target-tracking technique, realized using a LED light and a marker mounted on the measurement target; vibration frequencies are obtained from a feedforward neural network without any additional image or signal preprocessing. The proposed network consists of a set of 1D CONV layers, followed by batch normalization and a ReLU, connected to the number of the measurement frequency range (MFR) by two fully connected layers. The MFR is the product of the frequency range and the reciprocal of the measurement frequency precision step. Outputs of MFR is provided to a softmax layer for the vibration frequency prediction. The training process is developed on simulated data obtained from a random algorithm and adding noise to simulate real-world vibrations. The image-vision method is applied to a selected RoI of the original video to recognize every variation of brightness of pixels that correspond to a motion of the target. The method has been tested in laboratory and on the field site. The major limitation of this method is that when a different measurement range of frequency is required, a new model must be trained; moreover, only the principal frequency of the object can be detected by the approach. In this work, a computer with an NVidia GTX1060 GPU has been used.

10.3.4. Street object detection

The image-vision methods applied to street objects can be used to improves the efficiency of automation drive and system of security in the automotive field or to detect the health of asphalt and provides intervents.

Stallkamp et al. (2012) proposed a method for traffic signs detection. This work provides the application of several machine learning algorithms, to compare them and define the most accurate results. In particular, the application of four algorithms is reported: (i) linear classifier (LDA), (ii) multi-scale CNN, (iii) committee of CNNs, and (iv) Random Forest. The training dataset is made up of 10 h of video recorded on different German roads, with 144769 total labeled traffic signs, divided into 70 classes. The best algorithms, according to the authors, is the committee of CNNs, with 99.46% of correct classifications.

Zhang et al. (2017) presented a method to detect asphalt damage from 3D images of pavement surface taken from PaveVision3D system mounted in a Digital Highway Data Vehicle (DHDV). This system can take images at speed of about 100 km/h. The authors introduced in this work CrackNet, a CNN architecture made up of two FC layers, one CONV layer, one 1x1 CONV layer, and an output layer. Input is a features map of 360 features and the output is a prediction class for all pixels. The training dataset has more than 5000 3D images and it has been run on two NVidia GeForce GTX TITAN Black GPU.

11. DETECTION OF STRUCTURAL FEATURES

Computer image-vision methods, described in the previous section, can be applied to extract structure features from images; both object detection and object segmentation methods can be used. The aim of this work is to present an application of object detection to obtain information about masonry structure that influences the structure capacity and the evaluation of fragility curves of an urban area. In particular, the opening ratio of building facades is obtained from images of Google Street View, by CNN algorithms applied to 7 object detection methods, implemented in Python by two Pytorch (v1.6.0) libraries: Detectron2 (Yuxin Wu et al. 2019) and YOLOv5 (Glenn Jocher et al. 2020). Methods have been chosen by comparison reports of both libraries for both accuracy and time-machine. Each implementation required a set of parameters to optimize the accuracy of training and the time-machine; to set the best parameters and to define the more accurate method, a comparison is done using the AP value and the mAP value, evaluated on the validation set after 1000 iterations. In the next paragraphs details of the dataset, set of the algorithm, and error evaluation is reported for each method.

The training, the validation, and the detection phases have been performed on the Google Colaboratory platform using an NVidia Tesla K80s GPU or an NVidia T4s GPU.

11.1. Database

The input database of images was obtained from OpenImage Dataset, a free open-source database of Google which contains about 9 million images and labels with more than 600 different object categories. A starting dataset of 2000 images of the '*opening*' category was downloaded and a selection, a resizing, and a split in training and validation set of images are done using 'Roboflow'. A final dataset of 1000 images that contain 12786 openings, divided as 80% in the training set and 20% in the validation set, is obtained and it is exported in the required format from Python libraries. In particular, the COCO format is selected for Detectron2, and the YOLO format is selected for YOLOv5. Both formats contain images as jpeg files and labels as text files. The COCO format provides a partition of images in two folders, '*train*' and '*valid*', and a text file in JSON format for each folder to define objects and labels. The JSON file structure is divided into five sections:

- 'info': information about the creation of dataset;

```
"info": [ "year":
          "version":
          "description":
          "contributor":
          "url":
          "date_created":  ]
```

- 'licenses': information about dataset download;

```
"licenses": ["id": 1,
             "url": ,
             "name":     ]
```

- 'categories': name and number of object categories to detect;

```
"categories": ["id": 0,
               "name": ,
               "supercategory":     ]
```

- 'images': name, path to the file, and graphic features of each image; this section gives an 'id' number for each image

```
"images": ["id": ,
          "license": ,
          "file_name": ,
          "height": ,
          "width": ,
          "date_captured":  ]
```

- 'annotations': label and bounding-box of each object in the dataset, associated with the 'id' number of images; bounding-boxes are defined from 4 numbers which are the pixel coordinates of two opposite corner of the box;

```
"annotations": ["id": ,
               "image_id": ,
               "category_id": ,
               "bbox": [ , , , ]
               "area": ,
               "segmentation": [],
               "iscrowd":  ]
```

The YOLO format provides a partition of images in two folders as COCO format, and objects are defined from a YAML file and a text file for each image. The YAML file defines the path of training and validation image folders, the number of classes to detect, and the name of classes; the text file of the single image contains a row for each object in which the number of classes and the bounding-box are identified; in particular, bounding-box is defined with 4 numbers: two pixel cooridnates of one box corner, the width, and the hight of the box. Association between images and text files is done by naming corresponding files with the same name.

11.2. Two-stage frameworks

Faster R-CNN has been selected for this work among two-stage framework methods; the implementation is done by Detectron2, i.e. a state-of-art library developed by Facebook that includes several object detection methods. For each model, the library proposes several backbone layers and ResNet architectures.

In the training phase of each method, a configuration (‘*cfg*’) is required from the ‘*DefaultTrainer*’ script, in which several parameters can be set according to the used GPU and the dimension of the dataset; they are:

- *cfg.DATALOADER.NUM_WORKERS:* the number of data loading threads, set to 2;
- *cfg.SOLVER.IMS_PER_BATCH:* the number of images per batch across all machines, set to 2;
- *cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE:* the number of RoIs per training minibatch, set to 1024;
- *cfg.MODEL.ROI_HEADS.NUM_CLASSES:* the number of object classes in the dataset, set to 1;
- *cfg.SOLVER.BASE_LR:* the maximum value of the learning rate during training, set to 0.001;
- *cfg.SOLVER.MAX_ITER:* the number of iteration to do during the training process.

For the Faster R-CNN method, 4 architectures have been selected:

- Faster R-CNN with FPN ResNet 50;
- Faster R-CNN with FPN ResNet 101;
- Faster R-CNN with C4 ResNet 101;
- Faster R-CNN with FPN ResNext 101.

11.3. One-stage frameworks

YOLOv5 and RetinaNet methods have been chosen for the one-stage framework category. RetinaNet has been implemented by Detectron2 library and parameters have the same set showed in the previous section. The same '*DefaultTrainier*' script has been used for the training phase. Two different architectures have been selected for this method:

- RetinaNet with FPN ResNet 50;
- RetinaNet with FPN ResNet 101.

YOLOv5 implementation is done by the YOLO library that contains different YOLO architectures. The YOLOv5l is selected for this work, basing on a report of YOLO company about the accuracy and the time-machine of the proposed configurations.

In the training phase, the *'train'* script required a set of 3 parameters related to the dataset; they are:

- *img:* an integer that defines the pixel dimension of images, set to 640;
- *batch:* an integer that defines the batch size, i.e. the number of images to process simultaneously, set to 16;
- *epochs:* an integer that defines the number of iterations to do.

11.4. Methods comparison

To identify the best method of object detection for the treated issue, two types of comparison have been done after 1000 iterations. The former compares the error evaluation in terms of AP and mAP, as reported in Figure 23.

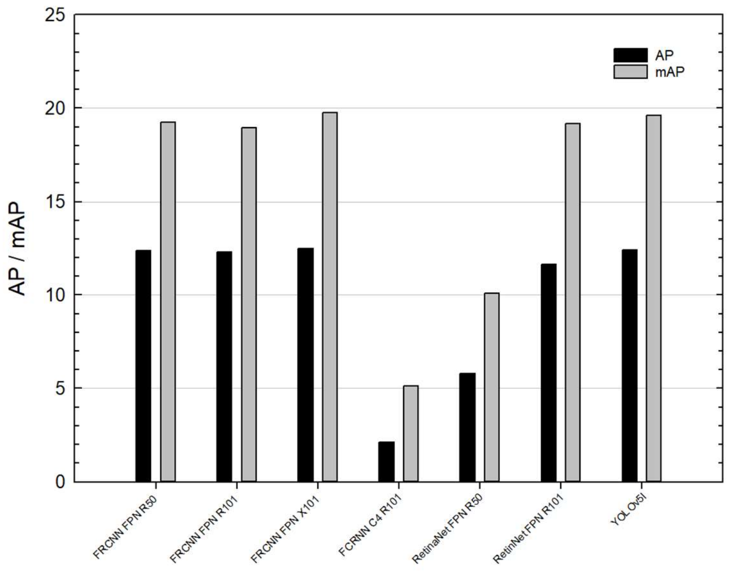


Figure 23 - AP/mAP comparison

The latter evaluates the method performance in relation to the time-machine needed, as reported in Figure 24.

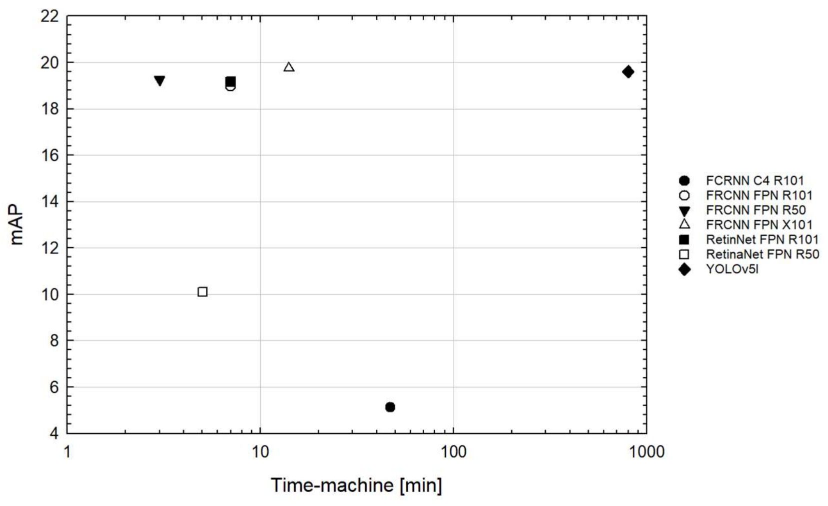


Figure 24 - time-machine / mAP comparison

Both comparisons show that only two methods (RetinaNet FPN R50 and FRCNN C4 R101) have low accuracy, while the others are close to the maximum value. The method that gives the best result is the Faster R-CNN with FPN ResNext 101 with an mAP of 19.747, executed in 14 minutes. Moreover, Figure 24 shows that the YOLOv5l model needs a higher time-machine during the training phase to obtain an mAP value close to the best one.

To increase the accuracy, more iterations have been run for the FRCNN FPN X101 method, monitoring the learning process by the '*total_loss'* value and the mAP. Generally, with the succeeding of iterations, the total loss has a global decreasing trend but it can be variable in a restricted range of iterations. Monitoring the global trend, the iteration process can be stopped when the total loss increases, i.e. when the algorithm does not converge; steps of 2000 or 3000 iterations have been considered. Figure 25 reports error evaluation (a) and total loss (b) up to 12000 iterations.

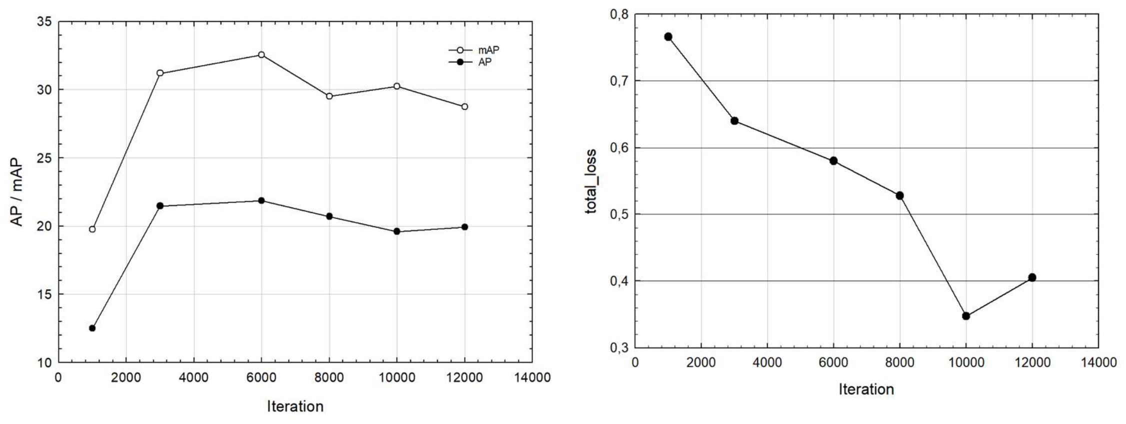


Figure 25 - Faster RCNN FPN X101 accuracy: a. error evaluation (left); b. total loss (b)

The total loss starts to increase at 10000 iterations, but the decrease of AP and mAP value at 8000 iterations suggest to stop the training process at 6000 iterations. The best score accuracy obtained is an AP value of 21.86 and an mAP value of 32.55 and the method with 6000 iterations is chosen for the detection phase.

11.5. Detection

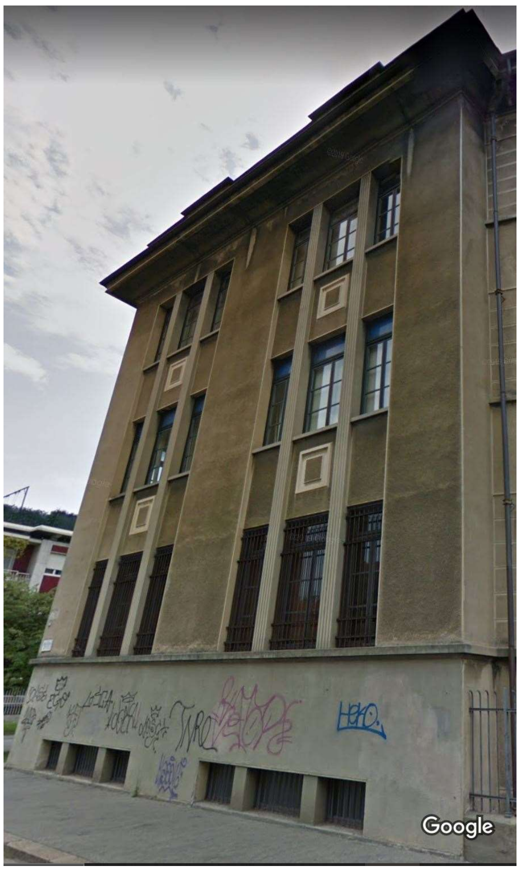
The detection process of Faster RCNN FPN X101 is done using the '*DefaultPredictor'* script of Detectron2 library and it is applied to a new set of images obtained from Google Street View. The dimension and resolution of images can influence the accuracy of detection as the perspective view.

For this reason, detection images are taken and edited in different ways to increase accuracy. In particular, three types of images are treated:

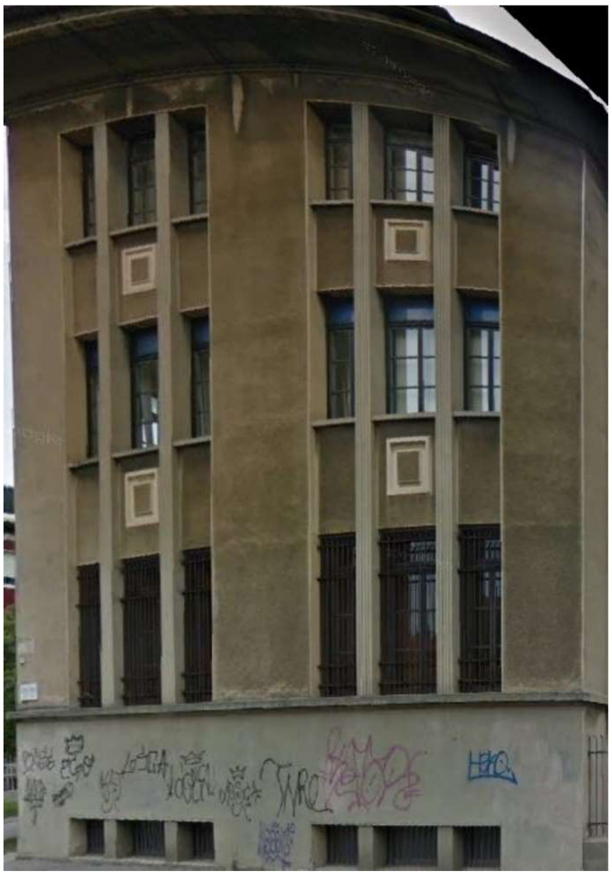
- images extracted directly from Google Street View and processed with a perspective view;
- images extracted from Google Street View and rectified by the photo editor ‘Fotus’ (AccaSoftware) ;
- panoramic images extracted from ‘Street View Downloader’.

Generally, the rectified images give the best detection, but also the panoramic view in which the building is in the center of the image has great accuracy. Perspective images are the worse ones because the far openings from the point of view are never detected.

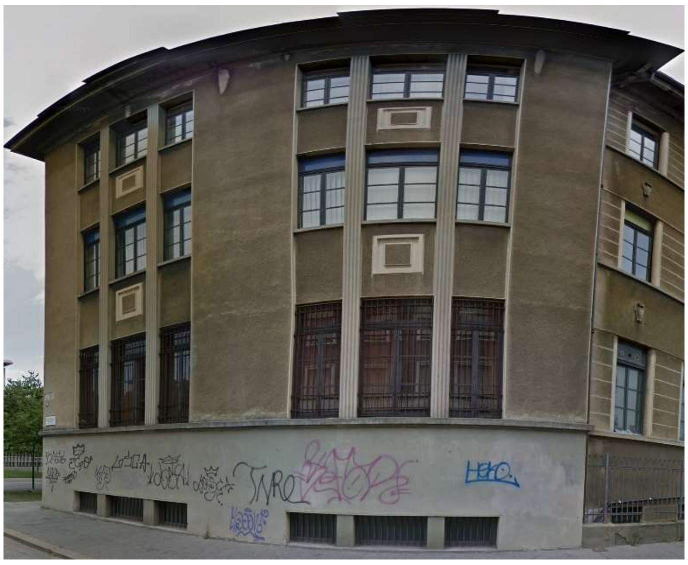
Figure 26 shows an example of each extraction.



(a)



(b)



(c)

Figure 26 – (a) perspective view; (b) rectified view; (c) panoramic view

'DefaultPredictor' results of a new set of images can be extracted as a graphic representation or numeric tensors. Figure 27 reports an example of graphic extraction, in which objects are defined by a bounding-box and an accuracy score of that prediction.

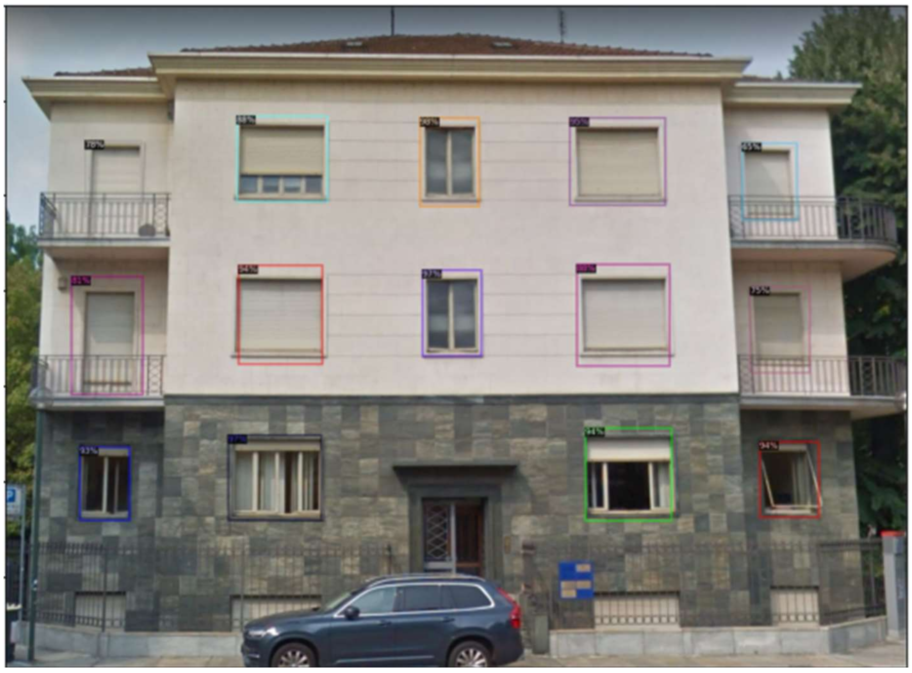


Figure 27 – Graphic result of detection

Numeric extraction is composed of three different tensors:

- the label tensor: defines the number of the classes of each prediction; in this work '0' identifies an opening;
- the box tensor: defines the pixel coordinates of bounding-box for each prediction;
- the score tensor: defines the percentage of accuracy of each prediction.

Numeric results of Figure 27 are reported below.

- Labels:
```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```
- Boxes:
```
[ [ 873.2425   239.7522    996.8872    424.2485 ]
  [ 878.0138   558.2391   1002.1027    738.8287 ]
  [ 468.5426   904.5457    665.0294   1082.7137 ]
  [ 1188.3962  239.7650   1389.0076    422.7645 ]
  [ 1220.9971  889.3556   1402.0692   1082.7384 ]
  [ 1587.6997  917.9448   1711.7898   1075.7483 ]
  [ 490.7195   549.0974    667.0352    755.8149 ]
  [ 156.8676   929.7191    261.0096   1081.9192 ]
  [ 1203.1855  546.8586   1398.3090    759.5513 ]
  [ 486.6751   235.4820    677.8959    414.4551 ]
  [ 139.3964   572.4117    289.3586    819.4720 ]
  [ 168.0329   287.8788    285.8762    446.1559 ]
  [ 1568.1112  593.8766   1693.2843    773.1409 ]
  [ 1551.1387  292.4910   1666.0417    453.9851 ] ]
```

- Scores:
```
[ 0.975819 , 0.973603 , 0.968940 , 0.952717 , 0.941174
0.940078 , 0.937691 , 0.932478 , 0.881045 , 0.878877 ,
0.813660 , 0.777529 , 0.749650 , 0.651123 ]
```

Appendix B reports other graphic and numeric extractions of the detection set.

In the reported images, it can be observed that the presence of objects (e.g. traffic lights, cars, people, lampposts, etc.) near or before an opening reduce the probability of detection; this limitation can be avoided by changing the perspective or panoramic point of view if possible. The manual photoshoot can solve the problem but is less fast and not always possible for the user.

11.5.1. Post-processing functions

If graphic results are used for user evaluation of the accuracy of detection, numeric results are used for the extraction of features. Four post-processing functions are implemented on Python for this work:

- *num_open*: returns the number of detected opening in the image;
- *coord_box*: return the bounding-box coordinates for each opening in the image;
- *eval_ratio*: return the opening ratio for the building in the image;
- *coord_open*: returns the detected opening location.

To evaluate the opening ratio, the *eval_ratio* function needs as inputs the real dimension of the building (*L* and *H*) and the pixel coordinates of the L dimension. The total opening area is evaluated by a proportion of the real dimension of the building and pixel coordinates, and then it is compared with the total area of the building to evaluate the opening ratio. *L* and *H* data can be obtained automatically from the 'Open Street Map' dataset

The *coord_open* function starts from *eval_ratio* results and estimates the real location of the openings in the building facade. This function is used to define geometric parameters of the frame model of the structure, as described in paragraph 12.2.

12. EVALUATION OF FRAGILITY CURVES

The prevention and design of post-disaster rescue operation is a relevant issue of modern civil engineering. In particular earthquakes, explosions, or fire are dangerous events for buildings or infrastructures that can suffer significant damages or collapse. Both conditions need rescue operations to save human lives and to reduce the damage of other nearby structures.

In the seismic field, the concept of *seismic risk* gives a numeric measurement of the probability of a structure to suffer damages that induce relevant losses for a community in a fixed period; its mathematic formulation is given by Equation (32):

$$R = H \times D \times L \tag{32}$$

where H is the seismic hazard (i.e. the probability of being an earthquake in a geographic area and a fixed period); D is the vulnerability (i.e. the possibility that structures suffer damages), and L is the seismic exposition (i.e. the measure of the structure importance in relation to its urban location). These parameters, and in particular D, defines the *fragility curves*, that explains the relation between the probability of damage (P_D) and intensity of the seismic event (I) for different damage states.

In the seismic field, fragility curves are used for large-scale analysis of urban areas when a FEM investigation of each building is not possible. In literature, several evaluation approaches of fragility curves have been presented, and information about the location area and the features of the structures are required.

The probability of failure of a structure is evaluated as a lognormal function referred to the selected test parameters of damage; data can be obtained from a computer simulation of seismic events or laboratory tests on a scaled sample of the structure. In particular, computer simulations are conducted applying static or dynamic non-linear analysis.

This work aims to propose a method to define fragility curves of masonry building of a district or a city, collecting structure information from images (object detection) and online datasets. The probability of damage is evaluated according to the most common methods in literature and the structure is analyzed by a simplified model (Marasco 2018). In the next sections, a literature review of P_D evaluation and a description of the proposed method is reported.

12.1. Related Works

The evaluation models of the fragility curves reported in literature are based on a lognormal distribution of the probability that the evaluation parameter (IM) of damage exceeds the limit damage state (DS) considered.

Erberik and Elnashai (2003), (2006) presented a simulation of 3300 time-histories with a non-linear dynamic analysis of a flat-slab structure; the inter-story drift ratio (IDR) is adopted as the parameter of damage, depending on the spectral displacement of the input shake. Four fragility curves related to four limit states (LS1, LS2, LS3, LS4) are evaluated by calculating the area of the lognormal distribution over the horizontal line of the considered LS. The lognormal function is reported in Equation (33):

$$P(DS > ds_i \mid d = IDR) = \Phi\left[\frac{1}{\beta_i}\cdot\ln\left(\frac{d}{\theta_i}\right)\right] \tag{33}$$

where θ is the median value of damage displacement threshold, β is the standard deviation, Φ is the standard normal cumulative distribution function, and d is the displacement of the structure. For each LS different values of θ and β are evaluated from a pushover analysis.

Masi et al. (2009) reported an evaluation approach based on the Housner intensity (I_H) parameter of damage, i.e. the integral of the pseudo-velocity response spectrum value between a period of 0.1 to 2.5. Non-linear dynamic analysis is performed on different structure samples with a different number of stories; the lognormal function used has the same structure of Equation (33) and evaluates the probability that damage exceeds the limit value of I_H for the considered LS. According to FEMA, five LSs have been considered.

Vazurkar U. Y. (2016) presented a work on the application of Hazus methodology (FEMA 2011) for the evaluation of fragility curves. The '*Federal Emergency Management Agency*' (FEMA) developed the Hazus methodology in which fragility curves are estimated using a lognormal function, similar to the previous, reported in Equation (34):

$$P(ds \mid S_d) = \Phi\left[\frac{1}{\beta_{ds}} \cdot \ln\left(\frac{S_d}{\overline{S}_{d,ds}}\right)\right] \tag{34}$$

where the median value $\overline{S}_{d,ds}$ is obtained from the performance point of a pushover analysis and the standard deviation β_{ds} is obtained from Equation (35) :

$$\beta_{ds} = \sqrt{(CONV\left[\beta_C, \beta_D\right])^2 + {\beta_T}^2} \tag{35}$$

where β_C is the lognormal standard deviation that describes the variability of the capacity curve, β_D is the lognormal standard deviation that describes the variability of the demand spectrum, and β_T is the lognormal standard deviation that describes the variability of the threshold of damage state. The lognormal standard deviation β_{ds} can be considered as a measurement of the total variability of the fragility curve and can be also evaluated by using tables reported in Hazus – MH 2.1, depending on the level of design and the degradation factor (κ). This method can be applied both to structural and non-structural components at four different damage states.

Cardone and Perrone (2015) presented a method for the evaluation of fragility curves of non-structural components, in particular masonry infill walls. Five groups of masonry are identified in which the presence of opening is considered at three different levels: (i) no opening, (ii) windows, and (iii) French windows. Since there are two possible conditions of damage, the IDR is adopted for the in-plane damage evaluation and the Peak floor acceleration (PFA) is adopted for the out-of-plane collapse. This work shows as fragility curve fits depend on the opening ratio (OP) but a relation between the probability of damage and the presence of windows is not defined; the contribution of OP is considered only in the standard deviation β, i.e. the uncertainty on input parameters.

12.2. Component modeling

The seismic investigation on masonry buildings results in a complicated process of modeling with possible overlooking of structural features. If detailed FEM models may be adopted for non-linear dynamic analysis, simplified models result faster to generate but need a validation phase to be used. The structure modeling adopted in this work is an Equivalent Frame Model (EFM) approximation based on the Simplified Analysis Method (SAM) presented by Magenes (2000). Similar to the POR method (Tomazevic 1978), the SAM provides modeling of the masonry wall as an equivalent 2D frame made by horizontal beams, vertical pier elements, and joints. Only the in-plane mechanisms of damage are considered. According to the authors, beams and piers are deformable and connecting joints have a rigid offset. Elastic-plastic behavior of components is considered and three failure criteria are adopted: (i) diagonal shear, (ii) sliding shear, and (iii) flexural/rocking. Dimensions of frame components depend on the location of openings in the facade as shown in Figure 28.

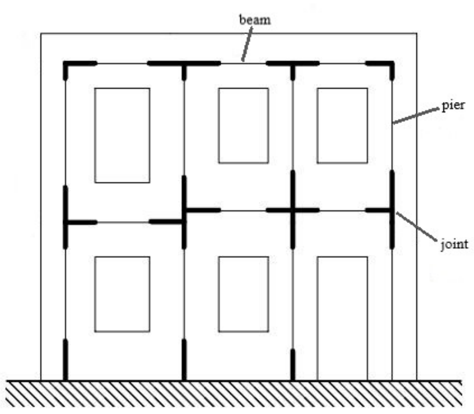


Figure 28 - SAM method

The adopted approximation, presented by Marasco, provides the same assumption of SAM, adding the conditions of the inextensibility of frames with concentrated nonlinearity properties in the most stressed sections (plastic hinges), lumped mass, and tri-linear backbone approximation of seismic response (Figure 29).

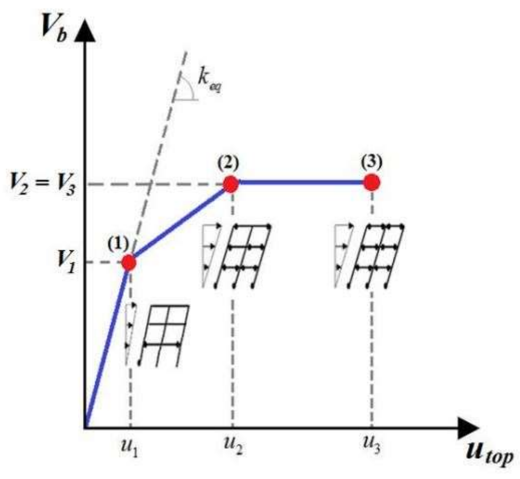


Figure 29 - Tri-linear backbone seismic response

The first point (1) refers to the generation of the first flexural or shear plastic hinge, the second point (2) refers to the maximum structure capacity, and the third point (3) is the collapse condition. To define the stiffness matrix, a bending type frame with a shear-flexure behavior of elements is considered and Equation (36) is used:

$$k_{ih} = \frac{1}{h_i^3 / E_i \cdot I_i + 1.2 \cdot h_i / G_i \cdot A_i} \tag{36}$$

where E_i and G_i are the elastic module and the shear elastic module of the *i-th* structural component. Non-linear parameters are defined by simplified procedures instead of pushover analysis, to reduce computational effort and time required. The procedures are resumed below:

- the maximum shear capacity ($V_2 = V_3$) is evaluated with the kinematic theorem of the limit analysis, calculating λ multipliers for both global and local collapse mechanism;
- the top displacement of collapse (u_3) is evaluated as the sum of three contributes: (i) displacement Δ_1 due to plastic hinges at the base of the column, (ii) displacement Δ_2 due to rotation on the top of the column, and (iii) displacement Δ_3 due to elastic deformation at each story;
- the displacement of point 2 (u_2) is evaluated with the equal energy rule applied with an iteration process.

The result of the approximation is an SDOF equivalent model of the structure reported in Figure 30.

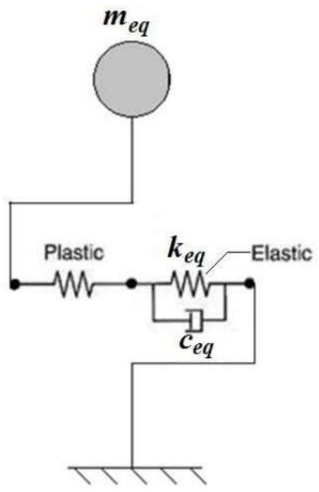


Figure 30 - SDOF model

12.3. Fragility curve

The evaluation of fragility curves is based on the lognormal distribution of P_D in relation to the selected IM parameter of the DSs, as reported in the previous section. The median value (θ) of IM and the standard deviation (β) of the capacity curve are required for the evaluation.

The adopted approach (Marasco 2018) defines the PGA as the IM parameter and considers four DSs: slight, moderate, extensive, and complete damage.

The first step of evaluation is the time-history analysis; the seismic scenario is defined by the GMSM method (Marasco and Cimellaro 2018) and 7 ground motions are individuated as representative of the seismic intensity of the site. Results of non-linear dynamic analysis conducted on the EFM by FEM software (SAP2000) are extracted as maximum absolute top displacement (i.e. refired to an SDOF system). The second step is the conversion of results in the maximum inter-story drift (i.e. refired to an MDOF system), according to the lateral displacement distribution. Finally, the PGA damage state limits are derived based on the threshold proposed by Ghobarah (2004) and the median IM values (θ) and standard deviation (β) are defined.

Application of object detection and feature extraction model, presented in this work, improve the speed and accuracy of the evaluation of fragility curves for a district or a city. The presented functions in section 11.5.1 are used to define the EFM and the linear/non-linear features of the corresponding SDOF model; in particular, *eval_ratio* and *coord_open* functions are used to the stiffness matrix and the tri-linear fit of the capacity curve. Time-history analysis conducted by SAP2000 on the SDOF model gives input data for the fragility curve definition.

12.4. Validation of the simplified model

The proposed model is applied to the building in Figure 27 and the results are compared with a FEM investigation on the same frame. The masonry structure has been modeled with Poroton P800, which features are reported in Table 12.

Table 12 - Material features

Medium strength [N/mm^2] (non-linear analysis)	
Compressive Strength (f_m)	4.75
Compressive Horizontal Strength (f_{hm})	0.76
Shear Strength (no vertical load) (f_{vm0})	0.26
Shear Strength (f_{vm})	$f_{vm0}+0.4\sigma_n$
Max Shear Strength ($f_{vm,lim}$)	0,98

Deformation Parameters [N/mm^2] (non-cracked masonry)	
Elasticity module (E)	3798
Shear module (G)	1519
Poisson module (ν) [adim.]	0.25

Numeric results of opening detection are reported in section 11.5. The EFM geometric representation in Figure 31 has been used for both analyses.

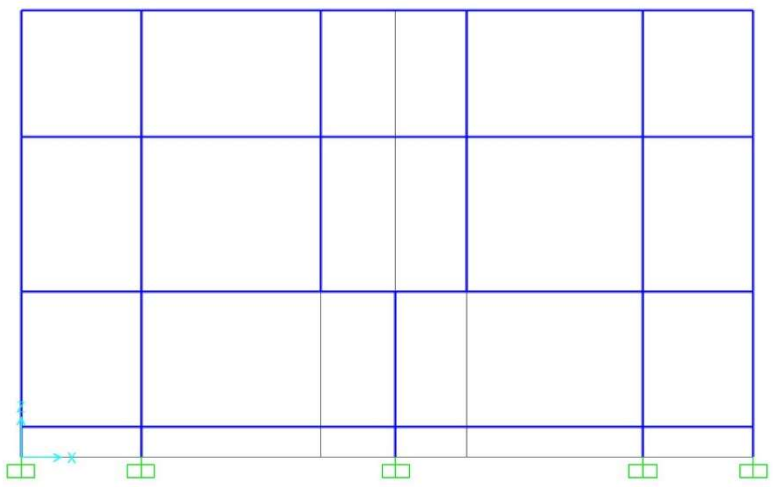


Figure 31 - FEM model

The FEM analysis performed in SAP2000 is based on the SAM model of the structure; the capacity curve obtained from a pushover analysis has been compared with that of the simplified model.

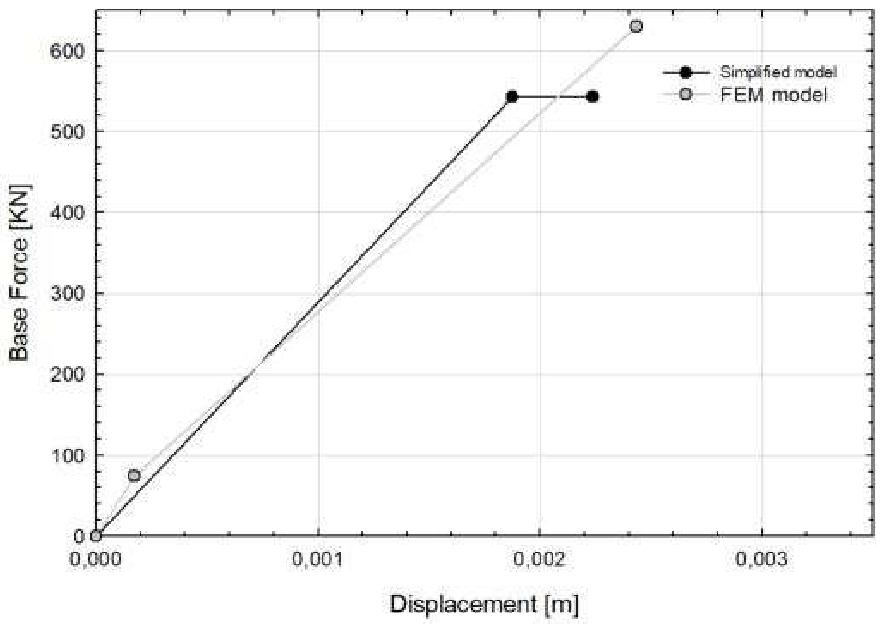


Figure 32 - Capacity curves comparison

Figure 32 shows that the simplified model accurately simulates the global capacity of the structure according to the equal energy rule; in particular, point (1) and (2) are aligned and point (3) define a plastic behavior that underestimates the top strength/displacement of the FEM analysis.

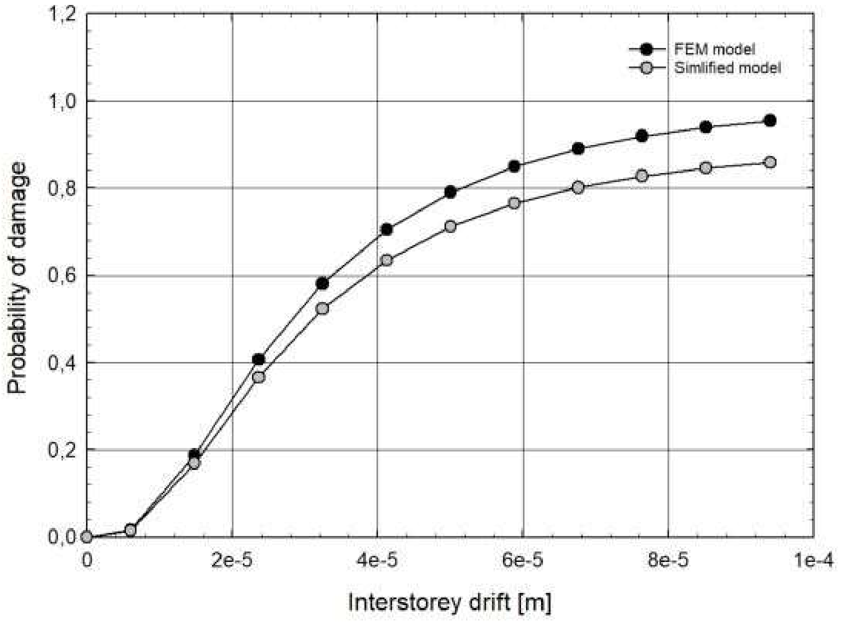


Figure 33 - Fragility curves comparison

In Figure 33 the comparison between the fragility curve of the FEM model and the simplified model is reported. The P_D evaluation of the SDOF model has 90% accuracy, so, obtained results can be considered as a valid approximation of the real behavior of the structure and can be used instead of the FEM model evaluation.

13. CONCLUDING REMARKS

Evaluation of the fragility curves of a district or a city is a relevant step for the prevention of seismic post-disaster scenarios. This work presented a Machine Learning (ML) application for the individuation of relevant features that influence the probability of damage of a masonry structure. In particular, an Object Detection (OD) model has been applied to pictures of symbolic structures in the considered urban area, to evaluate several fragility curves that identify the global behavior of the zone. Images have been extracted from Google Street View and processed by several OD models. An error evaluation on a test dataset identifies the Faster R-CNN ResNext101 FPN as the more accurate model for the individuation of openings in a building façade. From the numeric evaluation of location and dimensions of openings, a simplified Equivalent Frame Model (EFM) can be defined; the application of Marasco (2018) simplified analysis defines an equivalent SDOF model of the structure with a characteristic tri-linear approximation of the elastic/plastic behavior. After a non-linear dynamic analysis of the SDOF model, fragility curves have been obtained as the lognormal distribution of the mean values of the interstorey drift (ID). Obtained results have been compared with a FEM analysis and the simplified model shows a 90% accuracy. The application of the presented process substantially reduces the needed time for the FEM definition and analysis of the structure: object detection model identifies the real location and the opening ratio of the façade in a few seconds per image, and the simplified SDOF model reduces the time-machine for the time-history analysis.

Application of the presented work can improve the rescue operations in a post-disaster scenario, enabling a prediction of the most dangerous structures.

Future works can improve object detection accuracy, implementing the used training dataset to identify openings in a different perspective point of view; moreover, a correlation between the probability of damage of buildings and the debris extent evaluation, presented in chapter 1 of this work, can be defined, applying object detection model to several damaged structure.

APPENDIX A –
MACHINE LEARNING ARTICLES EXTRACTED FROM *STEP 1*

Authors	Article	Publication Year	Journal	Times Cited
Dibike, YB et al.	Model induction with support vector machines: Introduction and applications	2001	Journal of computing in civil engineering	296
Chou, JS et al.	Optimizing the Prediction Accuracy of Concrete Compressive Strength Based on a Comparison of Data-Mining Techniques	2011	Journal of computing in civil engineering	87
Golparvar-Fard, M et al.	Automated Progress Monitoring Using Unordered Daily Construction Photographs and IFC-Based Building Information Models	2015	Journal of computing in civil engineering	68
Zhou, J et al.	Classification of Rockburst in Underground Projects: Comparison of Ten Supervised Learning Methods	2016	Journal of computing in civil engineering	66
Moradkhani, H et al.	Statistical Downscaling of Precipitation Using Machine Learning with Optimal Predictor Selection	2011	Journal of hydrologic engineering	59
Huang, RQ et al.	Network-Scale Traffic Modeling and Forecasting with Graphical Lasso and Neural Networks	2012	Journal of transportation engineering	51
Chou, JS et al.	Machine learning in concrete strength simulations: Multi-nation data analytics	2014	Construction and building materials	40
Kim, C et al.	Automated Color Model-Based Concrete Detection in Construction-Site Images by Using Machine Learning Algorithms	2012	Journal of computing in civil engineering	39
Yan, SQ et al.	Applying Dynamic Surrogate Models in Noisy Genetic Algorithms to Optimize Groundwater Remediation Designs	2011	Journal of water resources planning and management-asce	37
Adeli, H et al.	A Novel Machine Learning Model for Estimation of Sale Prices of Real Estate Units	2016	Journal of construction engineering and management	35
Ying, YJ et al.	Toward Data-Driven Structural Health Monitoring: Application of Machine Learning and Signal Processing to Damage Detection	2013	Journal of computing in civil engineering	34
Lin, ZB et al.	Data-Driven Support Vector Machine with Optimization Techniques for Structural Health Monitoring and Damage Detection	2017	Ksce journal of civil engineering	33
Feng, ZK et al.	Forecasting Daily Runoff by Extreme Learning Machine Based on Quantum-Behaved Particle Swarm Optimization	2018	Journal of hydrologic engineering	29
Qi, CC et al.	Comparative Study of Hybrid Artificial Intelligence Approaches for Predicting Hangingwall Stability	2018	Journal of computing in civil engineering	23
Lee, S et al.	Comparative Study of Motion Features for Similarity-Based Modeling and Classification of Unsafe Actions in Construction	2014	Journal of computing in civil engineering	22
Li, L et al.	Structural damage identification based on autoencoder neural networks and deep learning	2018	Engineering structures	21
Hoang, ND et al.	Predicting Compressive Strength of High-Performance Concrete Using Metaheuristic-Optimized Least Squares Support Vector Regression	2016	Journal of computing in civil engineering	21
Hoang, ND et al.	A Novel Relevance Vector Machine Classifier with Cuckoo Search	2016	Journal of computing in civil engineering	20

	Optimization for Spatial Prediction of Landslides			
Deo, RC et al.	Forecasting Evaporative Loss by Least-Square Support-Vector Regression and Evaluation with Genetic Programming, Gaussian Process, and Minimax Probability Machine Regression: Case Study of Brisbane City	2017	Journal of hydrologic engineering	19
Cevik, A et al.	Support vector machines in structural engineering: a review	2015	Journal of civil engineering and management	19
Adamowski, JF et al.	Medium-Term Urban Water Demand Forecasting with Limited Data Using an Ensemble Wavelet-Bootstrap Machine-Learning Approach	2015	Journal of water resources planning and management	19
Hoang, ND et al.	Risk Score Inference for Bridge Maintenance Project Using Evolutionary Fuzzy Least Squares Support Vector Machine	2014	Journal of computing in civil engineering	18
Neves, AC et al.	Structural health monitoring of bridges: a model-free ANN-based approach to damage detection	2017	Journal of civil structural health monitoring	17
Dawood, T et al.	Machine vision-based model for spalling detection and quantification in subway networks	2017	Automation in construction	17
Jiang, LS et al.	Feasibility of Stochastic Gradient Boosting Approach for Evaluating Seismic Liquefaction Potential Based on SPT and CPT Case Histories	2019	Journal of performance of constructed facilities	16
El-Gohary, NM et al.	Semantic Text Classification for Supporting Automated Compliance Checking in Construction	2016	Journal of computing in civil engineering	15
Lin, ZB et al.	Time-Frequency-Based Data-Driven Structural Diagnosis and Damage Detection for Cable-Stayed Bridges	2018	Journal of bridge engineering	12
Cho, YK et al.	Principal Axes Descriptor for Automated Construction-Equipment Classification from Point Clouds	2017	Journal of computing in civil engineering	12
Yazdi, HS et al.	Prediction of Elastic Modulus of Concrete Using Support Vector Committee Method	2013	Journal of materials in civil engineering	12
Saavedra, A et al.	Bayesian Decision Tool for the Analysis of Occupational Accidents in the Construction of Embankments	2017	Journal of construction engineering and management	11
Okkan, U et al.	Bayesian Learning and Relevance Vector Machines Approach for Downscaling of Monthly Precipitation	2015	Journal of hydrologic engineering	11
Glinicki, MA et al.	Assessment of Scaling Durability of Concrete with CFBC Ash by Automatic Classification Rules	2012	Journal of materials in civil engineering	11
Kargah-Ostadi, N et al.	Framework for Development and Comprehensive Comparison of Empirical Pavement Performance Models	2015	Journal of transportation engineering	10
Bachour, R et al.	Estimation of Spatially Distributed Evapotranspiration Using Remote Sensing and a Relevance Vector Machine	2014	Journal of irrigation and drainage engineering	10

Pai, PF et al.	Using ADABOOST and Rough Set Theory for Predicting Debris Flow Disaster	2014	Water resources management	10
Smyth, AW et al.	Framework of Data Acquisition and Integration for the Detection of Pavement Distress via Multiple Vehicles	2017	Journal of computing in civil engineering	9
Chou, JS et al.	Nature-Inspired Metaheuristic Regression System: Programming and Implementation for Civil Engineering Applications	2016	Journal of computing in civil engineering	9
El-Gohary, NM et al.	Extending Building Information Models Semiautomatically Using Semantic Natural Language Processing Techniques	2016	Journal of computing in civil engineering	9
Tabatabaee, N et al.	Two-Stage Support Vector Classifier and Recurrent Neural Network Predictor for Pavement Performance Modeling	2013	Journal of infrastructure systems	9
Ying, YJ et al.	Damage Detection in Pipes under Changing Environmental Conditions Using Embedded Piezoelectric Transducers and Pattern Recognition Techniques	2013	Journal of pipeline systems engineering and practice	9
Zhang, J et al.	Zernike-moment measurement of thin-crack width in images enabled by dual-scale deep learning	2019	Computer-aided civil and infrastructure engineering	8
Goulet, JA et al.	Empirical Validation of Bayesian Dynamic Linear Models in the Context of Structural Health Monitoring	2018	Journal of bridge engineering	8
Alipour, M et al.	Load-Capacity Rating of Bridge Populations through Machine Learning: Application of Decision Trees and Random Forests	2017	Journal of bridge engineering	8
Bonakdari, H et al.	New Approach to Estimate Velocity at Limit of Deposition in Storm Sewers Using Vector Machine Coupled with Firefly Algorithm	2017	Journal of pipeline systems engineering and practice	8
Schmelter, ML et al.	Traditional and Bayesian Statistical Models in Fluvial Sediment Transport	2013	Journal of hydraulic engineering	8
Kandil, A et al.	Litigation Outcome Prediction of Differing Site Condition Disputes through Machine Learning Models	2012	Journal of computing in civil engineering	8
Rafiei, MH et al.	Novel Machine-Learning Model for Estimating Construction Costs Considering Economic Variables and Indexes	2018	Journal of construction engineering and management	7
Zhao, X et al.	Automated Traffic Surveillance System with Aerial Camera Arrays Imagery: Macroscopic Data Collection with Vehicle Tracking	2017	Journal of computing in civil engineering	7
Chen, FY et al.	Robust Adaptive Fault-Tolerant Control for Hypersonic Flight Vehicles with Multiple Faults	2015	Journal of aerospace engineering	7
Sun, C et al.	Application of a Machine Learning Method to Evaluate Road Roughness from Connected Vehicles	2018	Journal of transportation engineering part b-pavements	6
Yuen, KV et al.	Novel Sparse Bayesian Learning and Its Application to Ground Motion Pattern Recognition	2017	Journal of computing in civil engineering	6
Tseranidis, S et al.	Data-driven approximation algorithms for rapid performance	2016	Automation in construction	6

	evaluation and optimization of civil structures			
Meruane, V et al.	Online Sequential Extreme Learning Machine for Vibration-Based Damage Assessment Using Transmissibility Data	2016	Journal of computing in civil engineering	6
Nourzad, SHH et al.	Ensemble Methods for Binary Classifications of Airborne LIDAR Data	2014	Journal of computing in civil engineering	6
Nitsche, P et al.	Comparison of Machine Learning Methods for Evaluating Pavement Roughness Based on Vehicle Response	2014	Journal of computing in civil engineering	6
Torres-Rua, AF et al.	Machine Learning Approaches for Error Correction of Hydraulic Simulation Models for Canal Flow Schemes	2012	Journal of irrigation and drainage engineering	6
Li, CX et al.	Support Vector Machines Approach to Conditional Simulation of Non-Gaussian Stochastic Process	2012	Journal of computing in civil engineering	6
Wang, WY et al.	Sequence-based prediction in conceptual design of bridges	1997	Journal of computing in civil engineering	6
Jeong, HD et al.	NLP-Based Approach to Semantic Classification of Heterogeneous Transportation Asset Data Terminology	2017	Journal of computing in civil engineering	5
Gul, M et al.	Vibration-Based Damage Detection of Bridges under Varying Temperature Effects Using Time-Series Analysis and Artificial Neural Networks	2017	Journal of bridge engineering	5
Ao, H et al.	Roller Bearing Fault Diagnosis Method Based on Chemical Reaction Optimization and Support Vector Machine	2015	Journal of computing in civil engineering	5
Tilahun, N et al.	Incorporating Weather Information into Real-Time Speed Estimates: Comparison of Alternative Models	2013	Journal of transportation engineering	5
Cao, MT et al.	Predicting Equilibrium Scour Depth at Bridge Piers Using Evolutionary Radial Basis Function Neural Network	2015	Journal of computing in civil engineering	4
Rodriguez-Perez, JR et al.	Machine Learning Techniques Applied to the Assessment of GPS Accuracy under the Forest Canopy	2011	Journal of surveying engineering	4
Biswas, P et al.	Real-Time Identification of Cyber-Physical Attacks on Water Distribution Systems via Machine Learning-Based Anomaly Detection Techniques	2019	Journal of water resources planning and management	3
Fu, Q et al.	Application of Particle Swarm Optimization and Extreme Learning Machine Forecasting Models for Regional Groundwater Depth Using Nonlinear Prediction Models as Preprocessor	2018	Journal of hydrologic engineering	3
Karanci, E et al.	Modeling Corrosion in Suspension Bridge Main Cables. I: Annual Corrosion Rate	2018	Journal of bridge engineering	3
Wu, YJ et al.	Machine Learning Approach to Decomposing Arterial Travel Time Using a Hidden Markov Model with Genetic Algorithm	2018	Journal of computing in civil engineering	3
Hadidi, LA et al.	Effect of Failure Type on Downtime Duration for a Manufacturing Facility	2017	Journal of performance of constructed facilities	3

Batt, HA et al.	How to Utilize Relevance Vectors to Collect Required Data for Modeling Water Quality Constituents and Fine Sediment in Natural Systems: Case Study on Mud Lake, Idaho	2014	Journal of environmental engineering	3
Subramanian, D et al.	Building and Validating Geographically Refined Hurricane Wind Risk Models for Residential Structures	2014	Natural hazards review	3
Travis, QB et al.	Prediction of Intake Vortex Risk by Nearest Neighbors Modeling	2011	Journal of hydraulic engineering-asce	3
Pomponi, F et al.	Machine Learning for Sustainable Structures: A Call for Data	2019	Structures	2
Lin, ZB et al.	Enabling Damage Identification of Structures Using Time Series-Based Feature Extraction Algorithms	2019	Journal of aerospace engineering	2
Liu, F et al.	Performance prediction of impact hammer using ensemble machine learning techniques	2018	Tunnelling and underground space technology	2
Tinoco, J et al.	Robust Bad Data Detection Method for Microgrid Using Improved ELM and DBSCAN Algorithm	2018	Journal of energy engineering	2
Yadav, B et al.	Stability Condition Identification of Rock and Soil Cutting Slopes Based on Soft Computing	2018	Journal of computing in civil engineering	2
Nam, BH et al.	Simulation-Optimization Approach for the Consideration of Well Clogging during Cost Estimation of In Situ Bioremediation System	2018	Journal of hydrologic engineering	2
Zhu, HP et al.	Soil-Compressibility Prediction Models Using Machine Learning	2018	Journal of computing in civil engineering	2
Vanajakshi, LD et al.	Performing Global Uncertainty and Sensitivity Analysis from Given Data in Tunnel Construction	2017	Journal of computing in civil engineering	2
Lu, M et al.	Arterial Path-Level Travel-Time Estimation Using Machine-Learning Techniques	2017	Journal of computing in civil engineering	2
Lv, GY et al.	Modified Stepwise Regression Approach to Streamlining Predictive Analytics for Construction Engineering Applications	2017	Journal of computing in civil engineering	2
Cremona, C et al.	Voltage Sag Source Location Based on Pattern Recognition	2013	Journal of energy engineering	2
Lee, D et al.	Supervised learning algorithms for damage detection and long term bridge monitoring	2012	Bridge maintenance safety management resilience and sustainability	2
Vakharia, V et al.	Predicting Residential Water Demand with Machine-Based Statistical Learning	2020	Journal of water resources planning and management	1
Ahmadi, HR et al.	Prediction of compressive strength and portland cement composition using cross-validation and feature ranking techniques	2019	Construction and building materials	1
Wu, ZY et al.	Application of power spectral density function for damage diagnosis of bridge piers	2019	Structural engineering and mechanics	1
Lee, JS et al.	Evolutionary Deep Learning with Extended Kalman Filter for Effective Prediction Modeling and Efficient Data Assimilation	2019	Journal of computing in civil engineering	1
He, Q et al.	Prediction of Track Deterioration Using Maintenance Data and Machine Learning Schemes	2018	Journal of transportation engineering part a-systems	1

Lattanzi, D et al.	Joint Prediction of Remaining Useful Life and Failure Type of Train Wheelsets: Multitask Learning Approach	2018	Journal of transportation engineering part a-systems	1
Qiu, S et al.	Bridge Type Classification: Supervised Learning on a Modified NBI Data Set	2017	Journal of computing in civil engineering	1
Batt, HA et al.	Reducing the Effect of Inaccurate Lane Identification on PP69-10-Based Rut Characterization	2016	Journal of infrastructure systems	1
Melhem, HG et al.	Can Suspended Fine-Sediment Transport in Shallow Lakes Be Predicted Using MVRVM with Limited Observations?	2016	Journal of environmental engineering	1
Basu, B et al.	Machine learning and its application to civil engineering systems	1996	Civil engineering systems	1
Basu, B et al.	New Approach to Multisite Downscaling of Precipitation by Identifying Different Set of Atmospheric Predictor Variables	2020	Journal of hydrologic engineering	0
Liu, R et al.	Mining Social Media Data for Rapid Damage Assessment during Hurricane Matthew: Feasibility Study	2020	Journal of computing in civil engineering	0
Sun, LZ et al.	Detectability of Bridge-Structural Damage Based on Fiber-Optic Sensing through Deep-Convolutional Neural Networks	2020	Journal of bridge engineering	0
Choi, H et al.	Advanced Quality Control Models for Concrete Admixtures	2020	Journal of materials in civil engineering	0
Gattulli, V et al.	Data-driven optimal predictive control of seismic induced vibrations in frame structures	2020	Structural control & health monitoring	0
Azar, ER et al.	Estimating Stripping of Asphalt Coating Using k-Means Clustering and Machine Learning-Based Classification	2020	Journal of computing in civil engineering	0
Attoh-Okine, N et al.	Machine Learning Ensembles and Rail Defects Prediction: Multilayer Stacking Methodology	2019	Asce-asme journal of risk and uncertainty in engineering systems part a-civil engineering	0
Kim, CS et al.	Prediction of bond performance of tension lap splices using artificial neural networks	2019	Engineering structures	0
Jiang, ZS et al.	Infrasound-Based Noncontact Sensing for Bridge Structural Health Monitoring	2019	Journal of bridge engineering	0
Yang, XC et al.	Automatic Biomechanical Workload Estimation for Construction Workers by Computer Vision and Smart Insoles	2019	Journal of computing in civil engineering	0
Ghasemi, MR et al.	An intelligent health monitoring method for processing data collected from the sensor network of structure	2018	Steel and composite structures	0
Wu, YC et al.	Sustainable and Cost-Effective Pavement Preservation Method: Micromilling and Thin Overlay	2018	Journal of transportation engineering part a-systems	0
Chalouhi, EK et al.	Vibration-Based SHM of Railway Bridges Using Machine Learning: The Influence of Temperature on the Health Prediction	AND	Experimental vibration analysis for civil structures: testing sensing monitoring	0
Neves, AC et al.	A New Approach to Damage Detection in Bridges Using Machine Learning	AND	Experimental vibration analysis for civil	0

			structures: testing sensing monitoring	
Seow, MXC et al.	Correcting Systematic Underprediction of Biochemical Oxygen Demand in Support Vector Regression	2017	Journal of environmental engineering	0
Almeida, G et al.	Identifying Household Water Use through Transient Signal Classification	2016	Journal of computing in civil engineering	0
Yang, KJ et al.	Estimation of the Grain-Size Distribution Using Semisupervised Affinity Propagation	2015	Journal of hydrologic engineering	0
Effati, M et al.	Prediction of Crash Severity on Two-Lane, Two-Way Roads Based on Fuzzy Classification and Regression Tree Using Geospatial Analysis	2015	Journal of computing in civil engineering	0
Park, J et al.	Analyzing the Temporal Variation of Wind Turbine Responses Using Gaussian Mixture Model and Gaussian Discriminant Analysis	2015	Journal of computing in civil engineering	0
Vines-Cavanaugh, D et al.	In Field Application of Rapid Roadway Inspection System Using Vehicle-Mounted Multi-Modal Sensing	2013	Structural health monitoring 2013 vols 1 and 2	0
Imam, IFet al.	Learning flexible concepts for the wind bracing problem	1996	Computing in civil engineering	0

APPENDIX B –
Results of detection phase of Faster R-CNN ResNext101 FPN

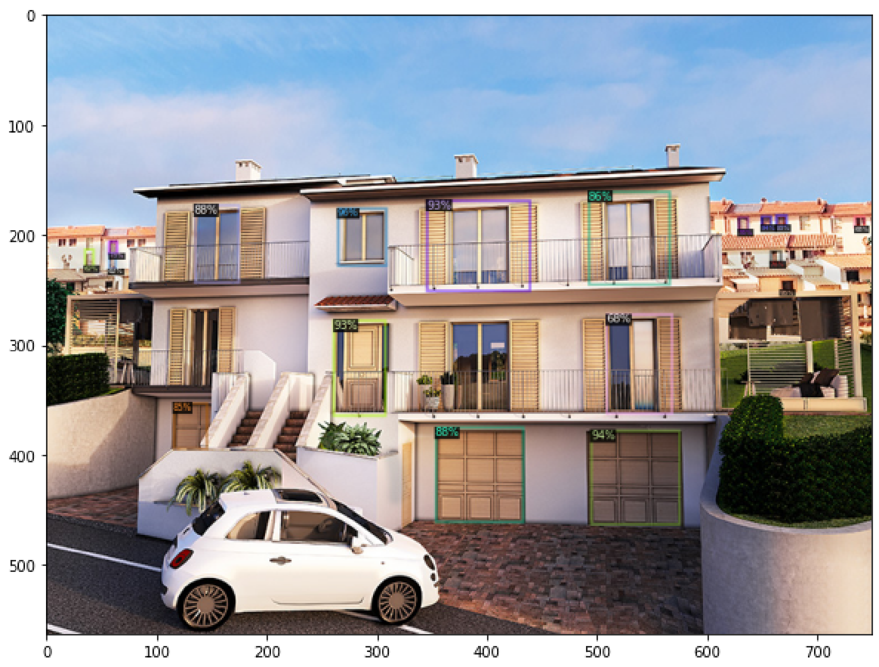


Figure 34 – Detection 1

Labels	Boxes				Scores	Opening ratio
	x_1	y_1	x_2	y_2		
0	264,9231	177,0799	308,9836	228,7264	0,9586	
0	666,9189	240,6361	680,7583	251,143	0,9483	
0	494,9333	378,5225	574,8807	463,0309	0,9438	
0	649,6607	184,165	658,3524	192,4658	0,9383	
0	663,0741	182,9074	671,899	192,6532	0,9319	0,20
0	261,2491	278,5709	307,805	364,7436	0,9309	
0	346,085	169,4243	438,7928	251,0871	0,9303	
0	353,806	375,7302	432,5551	460,7808	0,8823	
0	134,6785	173,556	175,1445	243,638	0,8799	

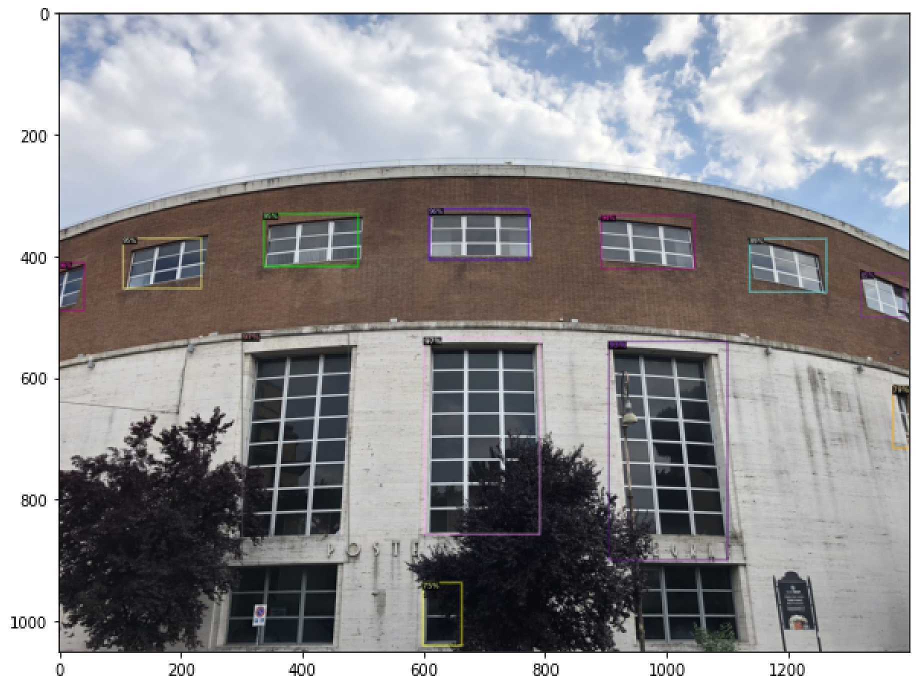


Figure 35 – Detection 2

Labels	**Boxes**				**Scores**	**Opening ratio**
	x_1	y_1	x_2	y_2		
0	601,7136	534,17	791,9277	857,0928	0,9735	0,24
0	302,2591	527,9074	477,4766	903,8524	0,9706	
0	609,5942	322,6937	773,3999	406,1789	0,9625	
0	0	411,3871	41,7848	489,8201	0,9587	
0	105,7011	370,7271	235,232	453,9614	0,9526	
0	337,6035	330,7029	492,9367	416,548	0,9458	
0	893,4379	334,0883	1046,692	420,3917	0,9418	
0	905,8902	540,6525	1100,887	898,6874	0,9302	
0	1136,83	371,481	1263,923	460,0085	0,8896	
0	1320,929	427,4391	1392,332	500,6025	0,8522	
0	1373,558	614,2265	1400	716,0573	0,7918	
0	600,1249	935,8387	663,2359	1039,604	0,7481	

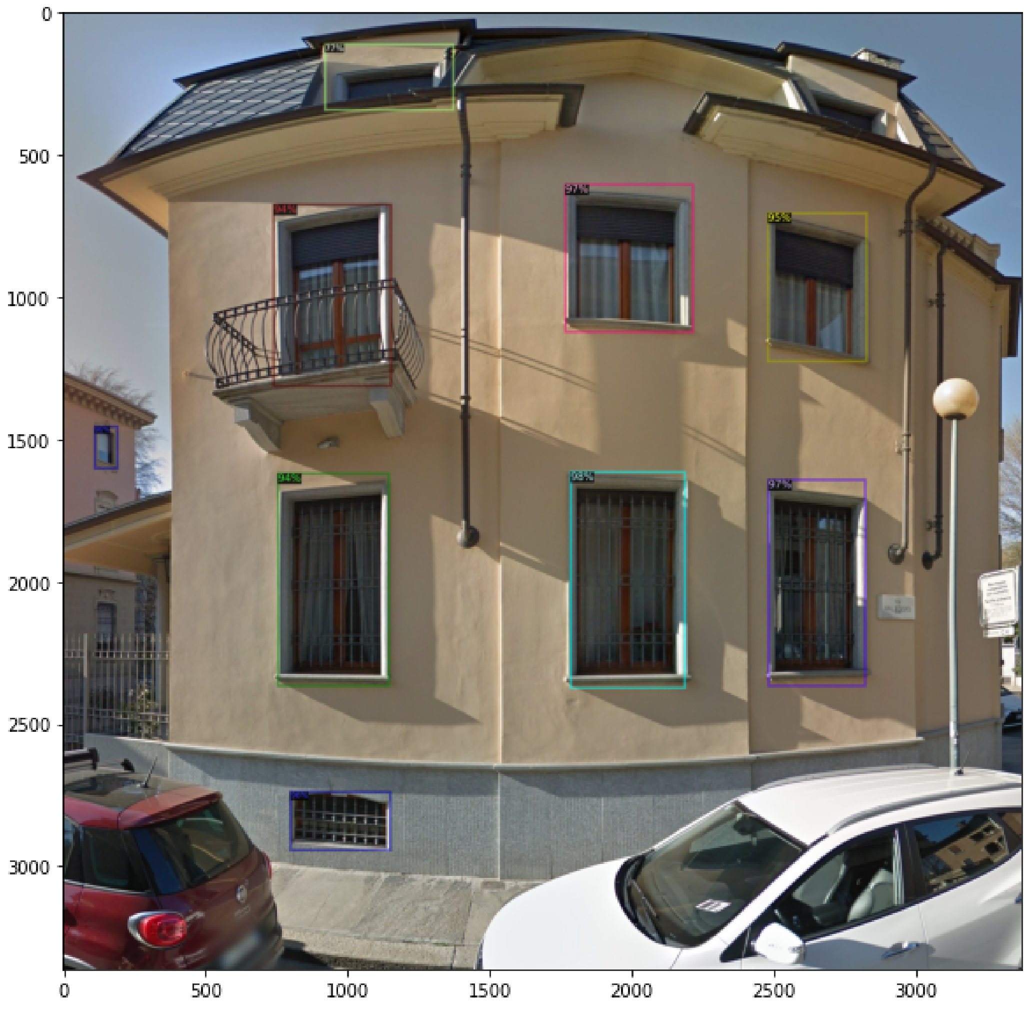


Figure 36 – Detection 3

Labels	Boxes				Scores	Opening ratio
	x_1	y_1	x_2	y_2		
0	1786,721	1615,783	2188,234	2374,762	0,9834	0,30
0	1767,858	606,485	2215,832	1123,798	0,9748	
0	2481,514	1643,323	2822,08	2366,295	0,9684	
0	801,8633	2739,573	1153,303	2943,128	0,9624	
0	113,7085	1456,73	192,5211	1602,908	0,9579	
0	2480,018	706,5493	2826,95	1227,07	0,9489	
0	756,5314	1621,03	1146,646	2366,004	0,9422	
0	742,2885	676,9799	1155,061	1313,304	0,9378	
0	923,7176	114,595	1373,576	343,9242	0,7698	

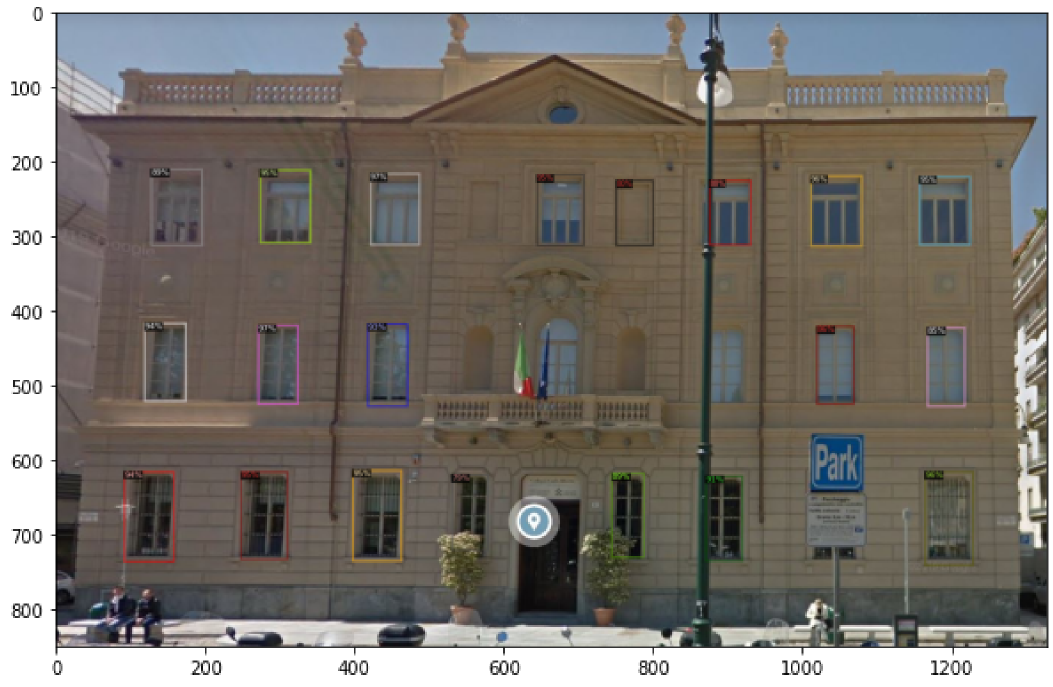


Figure 37 – Detection 4

Labels	Boxes				Scores	Opening ratio
	x1	y1	x2	y2		
0	421,7081	216,8419	486,9183	311,9136	0,971	0,16
0	271,2592	419,1763	324,051	524,0217	0,969	
0	1164,787	615,371	1228,856	740,5356	0,9632	
0	1011,806	219,4588	1079,546	313,8314	0,955	
0	1157,957	220,3064	1224,743	311,6269	0,9539	
0	248,1903	615,7675	310,3988	732,5573	0,9532	
0	397,7978	612,9668	463,3754	733,7719	0,9527	
0	644,6409	218,1649	707,4723	312,8714	0,9506	
0	273,9719	211,4928	341,3835	308,6852	0,9481	
0	92,1222	615,0067	157,6589	735,4909	0,9375	
0	119,3639	416,4813	173,6559	520,2803	0,9364	
0	418,4767	417,3091	470,6307	527,0535	0,9325	
0	871,1489	620,8339	919,6293	734,5856	0,9118	
0	745,4019	617,6537	788,4359	728,9729	0,8904	
0	128,1592	210,7031	195,8229	312,469	0,8882	
0	875,9107	225,2632	931,3513	311,18	0,8839	
0	1020,156	420,9634	1069,437	524,7141	0,8607	
0	1167,805	422,5177	1217,766	526,9594	0,8511	
0	751,2126	225,9017	800,1953	312,5845	0,7997	
0	530,8936	619,312	578,6857	719,6852	0,7918	

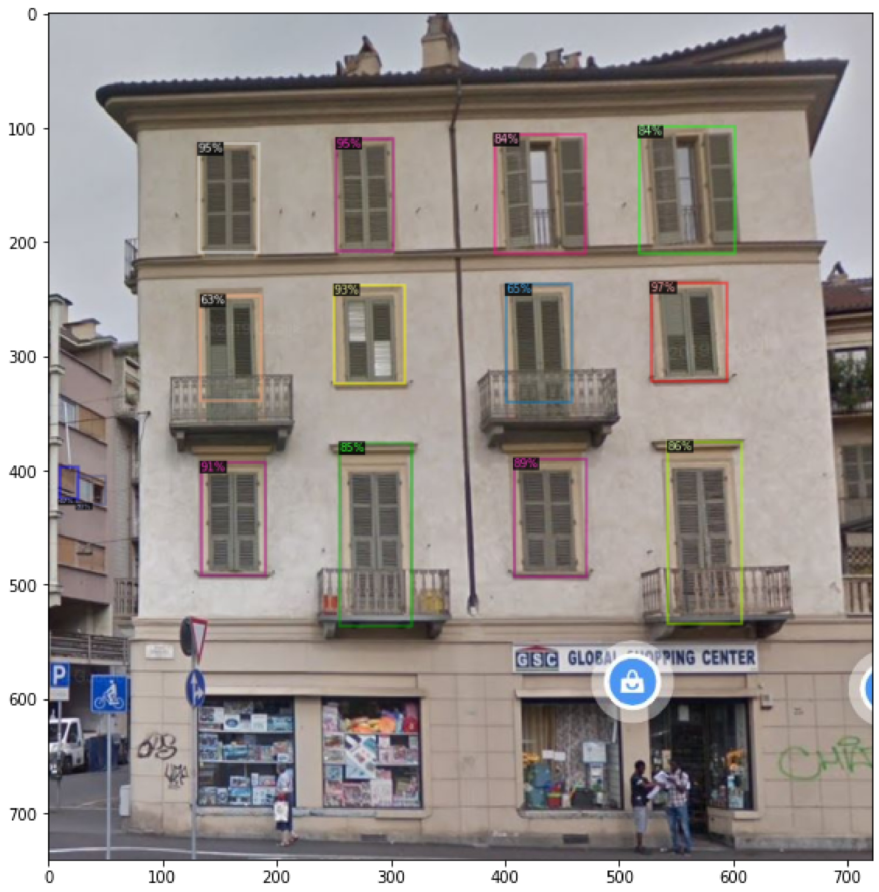


Figure 38 – Detection 5

Labels	**Boxes**				**Scores**	**Opening ratio**
	x_1	y_1	x_2	y_2		
0	527,6936	236,3896	593,7548	322,9989	0,9694	
0	131,6558	115,0468	183,8259	209,9839	0,9549	
0	252,4915	110,6332	301,8715	208,3594	0,9522	
0	250,3833	238,4113	311,9686	325,0388	0,9287	
0	133,8308	393,1595	189,8949	492,2303	0,9075	
0	407,573	390,0988	471,3427	493,0115	0,8851	
0	542,261	375,0743	606,5684	534,3165	0,8646	0,25
0	255,491	376,5842	318,0625	536,7421	0,8517	
0	390,6302	106,744	469,843	210,8477	0,8446	
0	516,5027	100,1982	601,3801	210,4067	0,8355	
0	24,7981	405,3992	49,6384	429,9466	0,7981	
0	9,5488	397,693	26,0396	425,6671	0,6672	
0	401,1602	238,1307	457,8889	340,6545	0,654	
0	133,4558	247,524	185,5396	339,3498	0,6287	

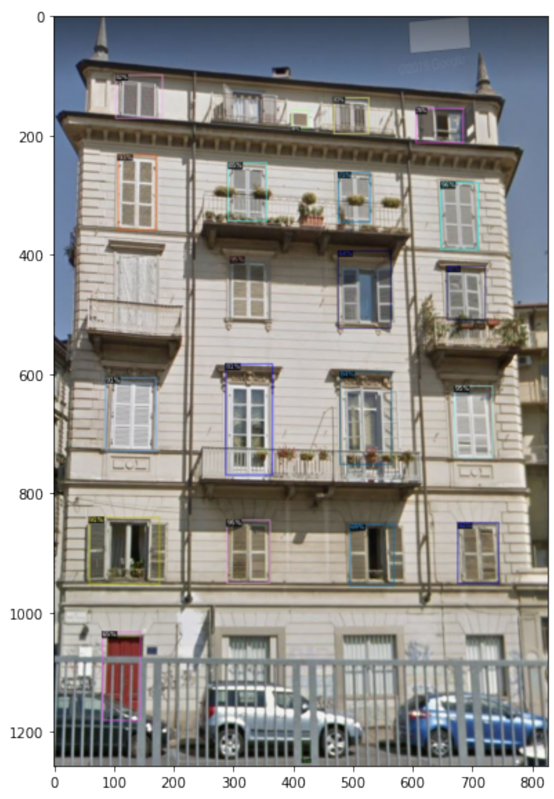


Figure 39 – Detection 6

Labels	Boxes				Scores	Opening ratio
	x_1	y_1	x_2	y_2		
0	647,9697	279,6671	709,6572	391,3613	0,9618	
0	290,0705	845,2998	362,196	949,9896	0,9508	
0	670,9411	620,9606	734,2631	740,2403	0,9508	
0	295,1574	404,8375	361,7502	510,3667	0,9495	
0	107,9881	233,5334	170,7815	358,3411	0,9321	
0	676,1906	849,7325	744,3848	950,8994	0,9204	
0	87,2241	606,2409	171,0699	729,7562	0,9125	
0	59,7456	840,3843	177,0385	952,1441	0,9101	
0	397,8985	159,3785	427,0435	187,3952	0,8855	
0	657,6363	419,8861	722,2132	509,5925	0,8813	0,20
0	292,2739	246,6863	356,2957	343,4731	0,8529	
0	479,2304	596,794	565,2313	751,4935	0,8389	
0	468,8142	137,967	526,8753	197,5303	0,8268	
0	288,0509	583,6569	365,5146	769,4528	0,8137	
0	606,395	156,0153	684,7675	213,1385	0,7801	
0	415,5479	1213,654	429,2619	1245,67	0,7524	
0	476,2226	262,3134	529,4855	345,3856	0,7149	
0	496,3573	852,6243	569,3348	955,6214	0,6923	
0	81,5697	1032,096	144,9766	1179,976	0,6458	
0	476,3241	394,4429	564,0933	523,6514	0,6435	

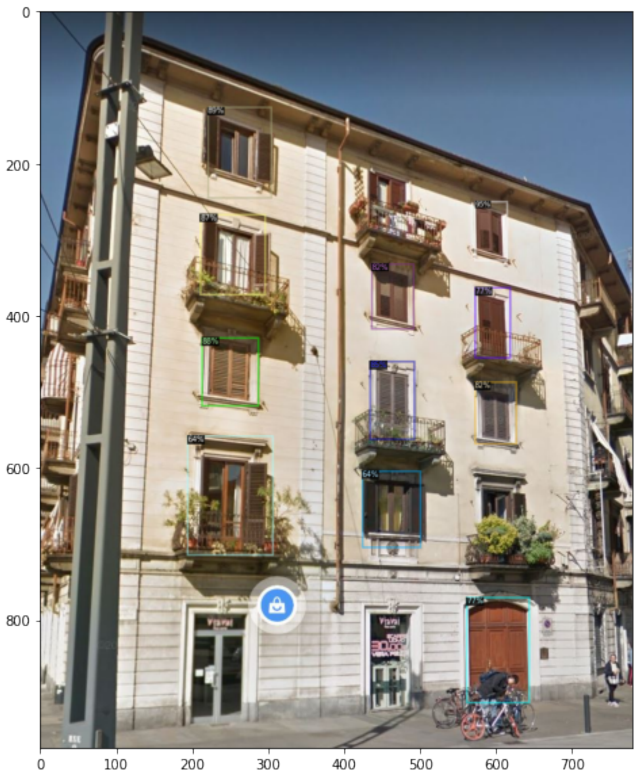


Figure 40 – Detection 7

Labels	Boxes				Scores	Opening ratio
	x_1	y_1	x_2	y_2		
0	572,2825	250,4203	614,1149	327,2379	0,9516	
0	220,5804	127,0863	304,2824	246,1804	0,8893	
0	213,3818	429,5836	287,4757	518,627	0,8829	
0	211,0656	267,5049	296,3767	372,8186	0,8695	
0	433,9421	460,5418	491,7554	562,3473	0,8479	
0	571,7697	487,7219	625,519	568,302	0,8232	0,15
0	436,2101	332,3547	490,872	417,7895	0,8168	
0	560,7852	770,5828	643,2183	908,4412	0,7735	
0	572,3752	363,7879	617,7044	455,1782	0,7652	
0	423,615	604,2324	500,0332	703,8224	0,6387	
0	194,42	558,4393	305,8589	714,4859	0,6387	

REFERENCES

Aljumaily, H., Laefer, D., and Cuadra, D. (2017). "Urban Point Cloud Mining Based on Density Clustering and MapReduce." *Journal of Computing in Civil Engineering*, 31, 04017021.

Araújo, C. A. (2006). "Bibliometria: evolução histórica e questões atuais." *Em questão*, 12(1), 11-32.

Badrinarayanan, V., Kendall, A., and Cipolla, R. (2017). "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation." *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(12), 2481-2495.

Bilal, M., Oyedele, L. O., Qadir, J., Munir, K., Ajayi, S. O., Akinade, O. O., Owolabi, H. A., Alaka, H. A., and Pasha, M. (2016). "Big Data in the construction industry: A review of present status, opportunities, and future trends." *Adv. Eng. Inform.*, 30(3), 500-521.

Bordin, A. S., Gonçalves, A. L., and Todesco, J. L. (2014). "Análise da colaboração científica departamental através de redes de coautoria." *Perspectivas em Ciência da Informação*, 19(2), 37-52.

Borgatti, S. 2002. NetDraw Software for Network Visualization, version 2.16, Analytic Technologies: Lexington, KY.

Breiman, L. (2001). "Random forests." *Mach. Learn.*, 45(1), 5-32.

Cardone, D., and Perrone, G. (2015). "Developing fragility curves and loss functions for masonry infill walls." *Earthquakes and Structures*, 9, 257-279.

Cha, Y.-J., Choi, W., Suh, G., Mahmoudkhani, S., and Büyüköztürk, O. (2018). "Autonomous Structural Visual Inspection Using Region-Based Deep Learning for Detecting Multiple Damage Types." *Computer-Aided Civil and Infrastructure Engineering*, 33(9), 731-747.

Chongchong, Q., Fourie, A., Ma, G., Tang, X., and Du, X. (2018). "Comparative Study of Hybrid Artificial Intelligence Approaches for Predicting Hangingwall Stability." *Journal of Computing in Civil Engineering*, 32(2), 04017086.

Desolneux, A., Moisan, L., and Morel, J.-M. (2000). "Meaningful Alignments." *International Journal of Computer Vision*, 40(1), 7-23.

Di Girolamo, G. D., Smarra, F., Gattulli, V., Potenza, F., Graziosi, F., and D'Innocenzo, A. (2020). "Data-driven optimal predictive control of seismic induced vibrations in frame structures." *Structural Control and Health Monitoring*, 27(4), e2514.

Dibike, Y. B., Velickov, S., Solomatine, D., and Abbott, M. B. (2001). "Model Induction with Support Vector Machines: Introduction and Applications." *Journal of Computing in Civil Engineering*, 15(3), 208-216.

Erberik, M. A., and Elnashai, A. S. (2003). "TECHNICAL REPORT

MID-AMERICA EARTHQUAKE CENTER

DS-9 PROJECT (RISK ASSESSMENT MODELING) ", University of Illinois

Erberik, M. A., and Elnashai, A. S. (2006). "Loss Estimation Analysis of Flat-Slab Structures." *Natural Hazards Review*, 7(1), 26-37.

Faber, F. A., Hutchison, L., Huang, B., Gilmer, J., Schoenholz, S. S., Dahl, G. E., Vinyals, O., Kearnes, S., Riley, P. F., and von Lilienfeld, O. A. (2017). "Prediction Errors of Molecular Machine Learning Models Lower than Hybrid DFT Error." *J. Chem. Theory Comput.*, 13(11), 5255-5264.

FEMA (2011). "Hazus FEMA's methodology for estimating potential losses from

disasters, FEMA Federal Emergency Management Agency, Washington,

D.C.".

Garcia, S., Kahhat, R., and santa cruz, S. (2016). "Methodology to characterize and quantify debris generation in residential buildings after seismic events." *Resources, Conservation and Recycling*, 117.

Ghobarah, A. (2004). "On drift limits associated with different damage levels." *International Workshop*Bled, Slovenia.

Girshick, R. "Fast r-cnn." *Proc., Proceedings of the IEEE international conference on computer vision*, 1440-1448.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. "Rich feature hierarchies for accurate object detection and semantic segmentation." *Proc., Proceedings of the IEEE conference on computer vision and pattern recognition*, 580-587.

Glenn Jocher, Alex Stoken, Jirka Borovec, and, N., and, C., and, L. C., and, L., and, A. H., and, l., and, t., and, y., and, A., and, L. D., and, M., and, w., and, m. a., and, D., and, H., and, J. P., Lijun Yu, and, c., and, P. R., and, R. F., and, T. S., and, W. X., and, Y., and, E. R. C., and, h., and, p. d., and yzchen (2020). "YOLOv5." Zenodo.

Goldberger, J., Roweis, S., Hinton, G., and Salakhutdinov, R. (2004). "Neighbourhood components analysis." *Proceedings of the 17th International Conference on Neural Information Processing Systems*, MIT Press, Vancouver, British Columbia, Canada, 513–520.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. "Mask r-cnn." *Proc., Proceedings of the IEEE international conference on computer vision*, 2961-2969.

He, K., Zhang, X., Ren, S., and Sun, J. "Deep residual learning for image recognition." *Proc., Proceedings of the IEEE conference on computer vision and pattern recognition*, 770-778.

Hirokawa, N., and Osaragi, T. (2016). "Earthquake Disaster Simulation System: Integration of Models for Building Collapse, Road Blockage, and Fire Spread." *Journal of Disaster Research*, 11(2), 175-187.

Huang, G., Zhuang, L., van der Maaten, L., and Weinberger, K. (2018). "Densely connected convolutional networks. arXiv 2018." *arXiv preprint arXiv:1608.06993*.

Hui, J. (2018). "mAP (mean Average Precision) for Object Detection." <https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173>.

Inglada, J. (2007). "Automatic recognition of man-made objects in high resolution optical remote sensing images by SVM classification of geometric image features." *Isprs Journal of Photogrammetry and Remote Sensing - ISPRS J PHOTOGRAMM*, 62, 236-248.

Jocher, G., Alex Stoken, Jirka Borovec, NanoCode012, ChristopherSTAN, Liu Changyu, Laughing, Adam Hogan, lorenzomammana, tkianai, yxNONG, AlexWang1900, Laurentiu Diaconu, Marc, wanghaoyang0106, ml5ah, Doug, Hatovix, Jake Poznanski, Lijun Yu, Prashant Rai, Russ Ferriday, Trevor Sullivan, Wang Xinyu, YuriRibeiro, Eduard Reñé Claramunt, hopesala, pritul dave, and yzchen. 2020. Ultralytics/yolov5: v.3.0, version v3.0Zenodo.

Johannesen, N. J., Kolhe, M., and Goodwin, M. (2019). "Relative evaluation of regression tools for urban area electrical energy demand forecasting." *Journal of cleaner production*, 218, 555-564.

Khadem, A., and Hossein-Zadeh, G. A. (2014). "Estimation of direct nonlinear effective connectivity using information theory and multilayer perceptron." *J. Neurosci. Methods*, 229, 53-67.

Knight, K., and Fu, W. (2000). "Asymptotics for lasso-type estimators." *Ann. Statist.*, 28(5), 1356-1378.

Krizhevsky, Sutskever, I., and Hinton, G. E. (2012). "ImageNet Classification with Deep Convolutional Neural Networks." 1097--1105.

Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., and Duerig, T. (2018). "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale." *arXiv preprint arXiv:1811.00982*.

Li, S., Zhao, X., and Zhou, G. (2019). "Automatic pixel-level multiple damage detection of concrete structure using fully convolutional network." *Computer-Aided Civil and Infrastructure Engineering*, 34(7), 616-634.

Liang, X. (2019). "Image-based post-disaster inspection of reinforced concrete bridge systems using deep learning with Bayesian optimization." *Computer-Aided Civil and Infrastructure Engineering*, 34(5), 415-430.

Lin, T., Goyal, P., Girshick, R., He, K., and Dollár, P. (2002). "Focal loss for dense object detection. arXiv 2017." *arXiv preprint arXiv:1708.02002*.

Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). "Microsoft COCO: Common objects in context. arXiv 2014." *arXiv preprint arXiv:1405.0312*.

Liu, J., Yang, X., and Zhu, M. (2019). "Neural Network with Confidence Kernel for Robust Vibration Frequency Prediction." *Journal of Sensors*, 2019, 6573513.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C., and Berg, A. (2020). "SSD: Single shot multibox detector. arXiv 2016." *arXiv preprint arXiv:1512.02325*.

Magenes, G. (2000). "A method for pushover analysis in seismic assessment of masonry buildings."

Marasco, S. (2018). "Large Scale Simulation of IDEAL

CITY under Seismic Scenario." Politecnico di Torino.

Marasco, S., and Cimellaro, G. P. (2018). "A new energy-based ground motion selection and modification method limiting the dynamic response dispersion and preserving the median demand." *Bulletin of Earthquake Engineering*, 16(2), 561-581.

Masi, A., Vona, M., and Digrisolo, A. (2009). "Costruzione di curve di fragilità di alcune tipologie strutturali rappresentative di edifici esistenti in c.a. mediante analisi dinamiche non lineari."

Meinshausen, N., and Bühlmann, P. (2006). "High-dimensional graphs and variable selection with the lasso." *The annals of statistics*, 34(3), 1436-1462.

Nguyen, K., Huynh, N. T., Nguyen, P. C., Nguyen, K. D., Vo, N. D., and Nguyen, T. V. (2020). "Detecting Objects from Space: An Evaluation of Deep-Learning Modern Approaches." *Electronics*, 9(4), 18.

Persson, O., Danell, R., and Wiborg Schneider, J. (2009). "How to use Bibexcel for various types of bibliometric analysis." *Celebrating scholarly communication studies: A Festschrift for Olle Persson at his 60th Birthday*, 9-24.

Qi, C., Fourie, A., Ma, G., Tang, X., and Du, X. (2018). "Comparative Study of Hybrid Artificial Intelligence Approaches for Predicting Hangingwall Stability." *Journal of Computing in Civil Engineering*, 32(2), 04017086.

Rafee, N., Karbassi, A. R., Nouri, J., Safari, E., and Mehrdadi, M. (2008). "Strategic Management of Municipal Debris aftermath of an earthquake." *International Journal of Environmental Research (ISSN: 1735-6865) Vol 2 Num 2*, 2.

Redmon, J., and Farhadi, A. (2018). "YOLOv3: An Incremental Improvement."

Ren, S. Q., He, K. M., Girshick, R., and Sun, J. (2017). "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(6), 1137-1149.

Scikit-learn, d. (2018). "Scikit-learn user guide."

Simonyan, K., and Zisserman, A. (2014). "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556*.

Singh, B., Najibi, M., and Davis, L. S. "Sniper: Efficient multi-scale training." *Proc., Advances in neural information processing systems*, 9310-9320.

Smola, A. (1996). "Regression estimation with support vector learning machines."

Stallkamp, J., Schlipsing, M., Salmen, J., and Igel, C. (2012). "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition." *Neural Networks*, 32, 323-332.

Tibshirani, R. (1996). "Regression Shrinkage and Selection Via the Lasso." *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1), 267-288.

Tomazevic, M. (1978). " The Computer Program POR Report ZRMK."

Van der Maaten, L., and Hinton, G. (2008). "Visualizing Data using t-SNE." *J. Mach. Learn. Res.*, 9, 2579-2605.

Vapnik, V. N. (1995). *The nature of statistical learning theory*, Springer-Verlag.

Vapnik, V. N. (1998). *Statistical Learning Theory*, Wiley-Interscience.

Vazurkar U. Y., C., D. J. (2016). "Development of Fragility Curves for RC Buildings." *International Journal of Engineering Research*.

Xu, D., and Tian, Y. (2015). "A Comprehensive Survey of Clustering Algorithms." *Annals of Data Science*, 2(2), 165-193.

Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Girshick, R. (2019). "Detectron2."

Zeiler, M. D., and Fergus, R. "Visualizing and understanding convolutional networks." *Proc., European conference on computer vision*, Springer, 818-833.

Zhang, A., Wang, K. C. P., Li, B., Yang, E., Dai, X., Peng, Y., Fei, Y., Liu, Y., Li, J. Q., and Chen, C. (2017). "Automated Pixel-Level Pavement Crack Detection on 3D Asphalt Surfaces Using a Deep-Learning Network." *Computer-Aided Civil and Infrastructure Engineering*, 32(10), 805-819.

Zhang, C. H., and Huang, J. (2008). "The sparsity and bias of the lasso selection in high-dimensional linear regression." *Ann. Stat.*, 36(4), 1567-1594.

Zhang, F., Du, B., and Zhang, L. (2016). "Scene Classification via a Gradient Boosting Random Convolutional Network Framework." *IEEE Transactions on Geoscience and Remote Sensing*, 54(3), 1793-1802.

Zhou, X., Wang, D., and Krähenbühl, P. (2019). "Objects as points." *arXiv preprint arXiv:1904.07850.*

Zou, H., and Hastie, T. (2005). "Regularization and variable selection via the elastic net." *J. R. Stat. Soc. Ser. B-Stat. Methodol.*, 67, 301-320.