On Real-time Management of On-board Ice Protection Systems by means of Machine Learning

Bárbara Arizmendi, Tommaso Bellosta, Anabel del Val, Giulio Gori, João Reis, Mariana Prazeres

On Real-time Management of On-board Ice Protection Systems by means of Machine Learning

Bárbara Arizmendi [1] and Tommaso Bellosta[2], Anabel del Val [3], Giulio Gori[4] and João Reis[5], Mariana Prazeres[6]

1. Introduction

In-flight ice accretion represents a serious threat to passengers safety and it usually results in a sharp drop of the aircraft performances. Current regulations impose strict requirements to manufacturers which must guarantee safety against atmospheric hazards for a broad range of conditions. Crystals are typically formed when the aircraft is flying through clouds in cold weather. According to [20], in rime icing conditions water droplets are suspended in the atmosphere at a supercooled metastable equilibrium. When particles impact the aircraft surface, their unstable equilibrium is perturbed and the phase change is triggered causing the instantaneous formation of ice.

Given the very short time scale characterizing the phase change dynamics, the ice layer can reach a critical thickness within a very short amount of time, namely in a few minutes. Aircraft are equipped with on-board Ice Protection Systems (IPS) aimed either at avoiding the formation of crystals, anti-ice systems (e.g. electro-thermal system, bleed-air systems and fluid-based ice protection system), or at removing a certain amount of ice which has already formed, de-icing systems (e.g. pneumatic boots or electro-mechanical systems). Note that the purpose of anti-icing and de-icing systems are different: de-icing systems aim at removing ice from the airframe whereas anti-icing systems prevents

[1]PhD Student - Marie Skłodowska-Curie research fellow, Department of Aerospace Science and Technology, Politecnico di Milano, Via Giuseppe La Masa, 34, 20156 Milano, Italy Email: barbara.arizmendi@polimi.it

[2]Master Student - Department of Aerospace Science and Technology, Politecnico di Milano, Via Giuseppe La Masa, 34, 20156 Milano, Italy Email: tommaso.bellosta@mail.polimi.it

[3]Marie Skłodowska-Curie research fellow, Aeronautics and Aerospace Department, von Karman Institute for Fluid Dynamics, 72 Chaussée de Waterloo, 1640 Rhode-St-Genèse, Belgium, AIAA Student Member

[4]Marie Skłodowska-Curie research fellow, DeFI Team, CMAP Lab (Ecole Polytechnique, Inria Saclay - Ile de France), 1 rue Estienne d'Orves, 91120 Palaiseau, France.

[5]PhD student/Marie Skłodowska-Curie research fellow, DeFI Team, CMAP Lab (Ecole Polytechnique, Inria Saclay - Ile de France), 1 rue Estienne d'Orves, 91120 Palaiseau, France. Email: joao-miguel.felicio-dos-reis@inria.fr.

[6]PhD Student, Department of Mathematics and Statistics, McGill University, QC H3A 0B9 Montreal, Quebec, Canada. Email: mariana.oliveiraprazeres@mail.mcgill.ca

water from freezing.

Typically, protection systems are activated whenever the risk of ice formation is detected, requiring a considerable amount of power since the surface exposed to icing, and for which the presence of crystals is critical, is generally large. Moreover, the broad range of conditions included in the flight envelope, for instance the diverse aircraft attitudes and the different type of clouds the plane is flying through, force manufacturers to apply conservative strategies thus preventing an efficient management of the on-board power.

An accurate and computationally cheap way of predicting ice formation at diverse locations over the aircraft is key to improve the real-time management of the on-board power addressed to protection systems, to ultimately increase the overall aircraft efficiency. Currently, there exist computational software specifically developed to predict the formation of ice in aeronautical applications. Examples of these suits are Ansys Fensap-Ice [14], CANICE [25], PoliMIce [13] from Politecnico di Milano, NASA LEWICE [24] and LEWICE3D [4] and ICECREMO [27]. These software are able to identify portions of the domain prone to ice accretion but, given the complexity of the underlying mathematical models, predictions are computationally expensive, especially when dealing with three-dimensional geometries. This makes the available software unsuitable for real-time anti-icing applications.

This work is a proof of concept aimed at assessing the potential of Machine Learning (ML) techniques for an efficient real-time management of on-board ice protection systems. In this contribution, ML tools are applied in the aim of substituting the full computational framework with predictive models trained on a representative data base. Once trained, ML models can be used to dramatically hasten the prediction of the impingement points of water droplets, at several flying conditions, ultimately providing information fundamental to an efficient real-time management of on-board ice protection systems. In this paper, several different algorithms are selected and evaluated in terms of their performance in identify critical regions prone to ice formation.

The use of ML technique in aeronautic icing applications is a quite unexplored field. Indeed, in the past year only a limited number of works concerning the topic were published. Among them all, it is worth to recall Ref. [28] and Ref. [29] in which the authors present a preliminary attempt to apply Proper Orthogonal Decomposition techniques and clustering algorithms in the context of icing certification procedures.

Due to the serious hazard entailed by conducting experiments during actual flights, literature is in short supply of data collected in real icing scenarios. For safety reasons, experiments are carried out in expensive cryogenic wind tunnels and they are devised to investigate the icing phenomenon on only small portions of the real geometry. Although these data are very valuable to aircraft certification and to research purposes, they are often incomplete and they usually suffer from quite large uncertainties. For this reason, in this paper a synthetic data set is generated numerically, to avoid dealing with any issue related to measurements. The data set is obtained using a framework that couples a Computational Fluid Dynamics (CFD) solver and an in-house Lagrangian par-

ticle tracking solver. Nevertheless, this synthetic data set is expected to be substituted, once available, by a comprehensive experimental data set.

The article is structured as follows: Sec. 2 describes the physics involved in the ice accretion problem. In Sec. 3, a summary of the main characteristics of the ML models employed in the paper, as well as the definition of the metrics used to assess their performances, is provided. Sec. 4 briefly describes the procedure followed to generate the synthetic data, presenting the underlying hypotheses and the numerical set up to carry out simulations. In Sec. 5 the results are presented: the ML algorithms performance are investigated and reported for comparison. Eventually, Sec. 6 summarizes the findings and provides future perspectives in the field of ML for the real-time management of on-board ice protection systems.

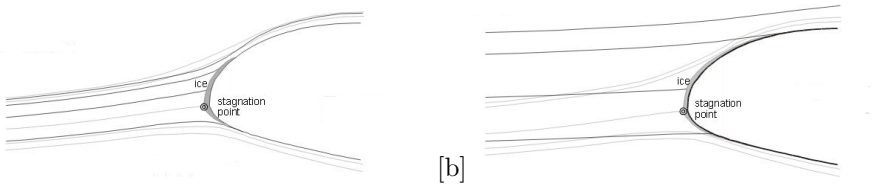
2. In-flight Icing Physical Modelling

According to [20], in the rime ice limit supercooled water droplets freeze upon impact against the surface of the aircraft. Therefore, knowing the droplets impinging points gives a way of identifying regions of the geometry subject to ice accretion. The prediction of droplets impinging points involves a complex procedure in which a demanding multi-physics problem needs to be solved.

Numerically, this information can be obtained by tracking droplets from an initial upstream position to their impact point over the frame. Practically, the trajectory of a particle dispersed in a fluid depends mainly on aerodynamic, buoyancy and inertial forces. For the considered application, additional contributions such as the Magnus, the Saffman and the Basset forces are negligible. An uncoupled kinematics is assumed, meaning that no interaction of any kind is considered among particles. Droplets are modeled as perfect rigid spheres that completely stick to the surface upon impact, meaning that no bouncing is accounted for. Aerodynamic forces depend on the relative velocity between the droplet and the fluid, on the properties of the fluid (i.e. air density and viscosity) and on the dimension of the droplet. The buoyancy force depends on the relative density difference among the particle and the fluid surrounding it whereas the inertial forces depend on the mass of the droplet. Since water is assumed to have a constant fixed density, buoyancy and inertial forces are ultimately proportional to the volume. Therefore, the motion of a droplet ultimately depends on its diameter, on the air viscosity and on the velocity and density relative of the flow field surrounding the particle. Moreover, surface portions prone to ice accretion differ significantly depending on the aircraft attitude i.e., the angle of attack.

Fig. 1 reports a comparison of the trajectories of droplets of different diameter where, Fig. 1(a) represents the trajectory of particles with a diameter of $20\mu m$ while Fig. 1(b) depicts particles with a diameter of $100\mu m$. One can see that the value of the diameter utterly dominates the droplet motion within the air stream. Small particles can follow the air streamlines more closely and, consequently, trajectories are significantly deflected in the close proximity of the

airfoil leading edge. However, droplets of larger diameters are associated with larger inertial forces that make the particles follow a straighter trajectory.



[a] [b]

Figure 1: Trajectory (dark grey lines) of water droplets of different diameter and mass. Light gray lines correspond to air streamlines. (a) $20\mu m$ diameter; (b) $100\mu m$ diameter.

Clouds are large clusters of water particles of different sizes suspended in the atmosphere. A probability density function is assigned to describe the droplets diameter distribution and to characterize the cluster. Typically, such distributions are different for different cloud types (fog, stratus, nimbostratus, etc.). In icing applications, the full probabilistic characterization of clouds is dropped in favor of the Mean Volume Diameter (MVD), a parameter representative of the whole particle size distribution spectrum. In Ref. [10] the authors reconstruct the water distribution over a two-dimensional geometry and they show that considering a collection of droplets of same size, equal to the MVD, leads to similar results as if one considers the full diameter spectrum. Though the referred work treats a cylindrical body, the latter assumption may be reasonably extended to an airfoil profile, as the region affected by ice accretion basically corresponds to the round leading edge. According to this research, a cloud can be modelled as a set of droplets of constant size equal to the MVD.

The in-flight measurement of the cloud properties, and in particular the MVD, requires special airborne probes [2], such as the Forward Scattering Spectrometer. The accuracy of these devices is rather limited, which adds uncertainty to the experimental data recorded. Due to the use of synthetic data, uncertainty is assumed negligible. However, uncertainty should be addressed when dealing with experimental data.

3. Machine Learning Algorithms

The real-time prediction of ice accretion is a challenging and very expensive task, especially when dealing with a fully three-dimensional description of the aircraft. In this context, the present work proposes to bypass this numerical barrier by taking advantage of the now-ubiquitous ML techniques.

In particular, ML algorithms are trained to predict which parts of the wing surface are prone to ice formation during the flight. The prediction is achieved by means of classification algorithms that evaluate the flight conditions to identify the different surface regions as a binary scenario where there will be or there will not be ice. In this section, the algorithms considered are outlined, as well as the reasons for their choice. Fig. 2 sketches the input/output workflow

corresponding to the conventional and the ML approaches. The ML approach trains on the input data used for the two solvers of the conventional approach and learns to predict ice formation. The input comprises an array of 5 scalars: Mach number (M [-]), angle of attack (α [deg]), density (ρ [Kg/m^3]), atmospheric temperature (T [$^\circ C$]) and median volume diameter of the impinging droplets (MVD [μm]). The output consists of a binary vector of size equal to the surface grid points where the risk formation is be evaluated. Each element of this binary vector can take a 0/1 value, depending whether ice will or will not be formed at that specific location under the input conditions.

Selecting the most appropriate ML algorithm is not straightforward due to the bias-variance trade-off [15]. The bias-variance trade-off prevents the minimization of both the error of the algorithm on its training data and the sensitivity to small changes in the input. Hence, for the ice formation problem, several algorithms are considered training the models using the generated data.

In this work, the performance of both simple and complex ML models is assessed in order to find the strengths and the weaknesses of each approach. Note that the input and output distributions might not be a faithful representation of reality. Hence, the selected algorithms should not rely on any assumption regarding input/output distribution.

First, the focus is on less sophisticated ML models which are expected to be of easier implementation and interpretation, providing valuable insights to the problem. Nevertheless, the selected algorithms should be flexible enough to allow for the capturing of non-linearities. In [11], several techniques which are deemed suitable for ice prediction applications are presented and compared. Besides the recommendations of [11], the use of decision trees is also explored [15]. Among others, in this work the scikit-learn library (v0.20.0) is exploited [23] due to its flexibility, open-access, and easy usability. Scikit-learn is a python-based library implementing various ML models and training routines.

Furthermore, more complex deep learning models are also considered. Despite being available since a long time [19], the advent of big data sets and powerful hardware lately boosted the growth of Artificial Intelligence (AI) models based on deep learning artificial neural nets. For example, in 2012, human-like accuracy was achieved in recognizing digits and traffic signs [7, 6], being able to generalize to unseen data while fitting the given data. In spite of this extraordinary ability, the field still lacks interpretability and understanding. Due to the proved capabilities of the technique, deep learning techniques are also considered. In doing so, the pytorch library is used [22] since it implements all the necessary routines and algorithms. In order to implement test as for the ML models, a pytorch wrapper compatible with scikit-learn is used, skorch [1].

A summary of the main features of all the considered ML models is found next.

3.1. Decision Trees

Decision Trees (DT) are ML models used to predict the target value or label of an item based on specific observations of certain features [15]. For

the purpose of this study, some of the advantages of decision trees over other models include the fact that they are easy to use, requiring little, if none, data preprocessing. Moreover, they can be easily interpreted and can accommodate non-linearities. On the other hand, some disadvantages have to be taken into account when dealing with decision trees. For instance, the number of samples is logarithmic in tree depth, they are unstable (meaning they can be easily perturbed with small differences in data), they are excellent overfitters and, since they only consider a single feature at a time, they have difficulty handling model additivity.

In the particular case of ice-accretion prediction, the fact that they account for non-linearities is a strong motivation to use them as well as the fact that they are easily interpretable. Overfitting issues must be taken care of by limiting the maximum depth of the trees used. Due to their unstable nature and the fact that a robust model is desirable for this application, decision tree classifiers and regressors are considered as base estimators for some ensemble methods to reduce their variance and improve the model performance. The ensemble methods considered are Random Forest, Extremely Randomized Trees, AdaBoost (from now on referred to as Boosting) [12] and Bootstrapped Aggregation or Bagging [5]. There is extensive literature about each of these ensemble methods as well as the scikit-learn library page. The performance of each of these methods is assessed in the results section of this work through a comparative study.

3.2. Support Vector Machines

Support Vector Machines (SVM) are both supervised and unsupervised learning models that fit both classification or regression tasks. In this manuscript, SVM are considered for classification problems only, therefore they are referred to also as Support Vector Classifiers (SVC). Among others, in the scikit-learn library two main algorithms are available and employed in this work namely C-SVC and ν-SVC. In this work we use C-SVC. These algorithms are rooted in the idea of separating data points using an hyperplane which maximizes the distance to all the data points, see [8]. In supervised learning, the algorithm gets a set of labeled training points $\mathbf{x}$ in a d-dimensional space and computes a hyperplane (or a set of hyperplanes) of dimension $d-1$ that separates the classes. Usually, the hyperplane is computed so that it maximizes the gap, or *margin*, between two classes i.e., the distance of the nearest points in both classes from the hyperplane. The parameter C quantifies how much this margin is. For example, large values of C yield small margins, and vice versa. Consequently, large/small values of C can lead to over/under fitting. Once the hyperplane is defined, new queries are classified according to the side in which they fall. This approach would generally limit the suitability of SVMs to linear classification problems. Nevertheless, SVMs can also deal with non-linear classification problems by means of what is commonly referred to as *kernel trick*. Kernel trick consists in mapping points into a space of higher dimension in the hope that the augmented space could possibly lead to an easier, but computationally more expensive, training process. Different kernel $k(\mathbf{x}_1, \mathbf{x}_j)$ functions may be

employed, the best one depending on the considered training data set. Namely, in this paper we considered

Polynomial	$(\gamma \mathbf{x}_1 \cdot \mathbf{x}_j)^p$	$p > 0$
Radial Basis Functions (RBF)	$\exp(-\gamma \|\mathbf{x}_1 - \mathbf{x}_j\|^2)$	$\gamma > 0$

The p parameter in C-SVC can be set to one (or higher values) to obtain a linear (or a non-linear) kernel. Note that the training of the ML model becomes generally more and more expensive as the degree of the polynomial increases. Indeed, for $p = 3$ the C-SVC training w.r.t. the synthetic data is found to be computationally too demanding. Therefore, in this work non-linear C-SVC are limited to $p = 2$. γ is instead an additional parameter that allows tuning for better data fitting. Specifically, the parameter γ for the quadratic kernel ($p = 2$) determines the shape of the parabola fitting the data. The higher the γ, the sharper is the parabola. Therefore, finding the right gamma obviously depends on the structure of the data.

On the other hand, the RBF kernel depends on a single parameter γ which is usually defined as $\gamma := 1/(2\sigma^2)$ whereas σ^2 is the variance of the data. The parameter γ therefore measures the influence of a training point on the remaining ones. A large γ means small variation of the data, meaning that we expect the nearby points to give similar classifications. This may lead to underfitting. On the other hand, small values of γ imply large variations, which may lead to overfitting. In a very natural way, the γ parameter must be tuned to avoid either overfitting or underfitting. The difference between C and γ is that the former influences on the margins of the hyperplane, whereas the latter governs the the importance of each point.

For the sake of completeness, SVMs may also be used as unsupervised learning models in clustering problems. Indeed, SVMs can separate data into clusters by using, for instance, support vectors statistics. Nevertheless, in this work SVMs are considered for classification purposes only.

3.3. k-Nearest Neighbors

K-nearest neighbors (k-NN) is an instance-based algorithm typically employed for classification and regression. It belongs to the lazy learning tool family, therefore, no explicit training step is required. The algorithm takes an integer parameter k as input and searches the training data set on the fly, to find the k elements closest to the query point. Usually, distance is intended in the Euclidean sense, but different metrics may apply. This results into a high-sensitivity of the k-NN w.r.t. to the local structure of the data.

In classification problems, the algorithm outputs a label corresponding to the most recurring one among the k samples nearest the query point. In regression, the output is given by an average of the value of the k closest elements. In both cases, one could assign a weight to each of the neighbors, usually the distance itself, so that nearer neighbors contribute more than the distant ones. k is a user-defined parameter and, in binary classification problems as the one investigated, it must be an odd number so to avoid tied votes. Usually, k is selected via an

heuristic procedure aimed at optimizing the parameter w.r.t. the considered data set. On one hand, large k values result in blurred boundaries, which lead to the misclassification of a query point located in the close proximity of class borders. On the other hand, a small k yields a classification based on a small number of elements which may be poorly representative of the local structure of the data.

3.4. Deep Neural Networks

Deep Neural Networks (DNNs) are a supervised ML tool. A DNN is a parameterized function that can be represented as a graph of connected neurons. When taking a set of N labeled training data points of the form (x_i, y_i), the DNN should be a function f_w such that $f_w(x) = y$ for all x in the data-space. The function f_w is defined a priori, but the weights w must be found via the following minimization problem,

$$\min_w \frac{1}{N} \sum_i loss_i(f(x_i; w), y_i) =: \min_w loss(f(x; w), y).$$

The loss function much be chosen in such a way that prevents the memorization of the training set, learns the underling distributions and leads to good generalization on testing data sets. For classification problems, a popular choice also employed in this paper is the Cross Entropy Loss. To solve the above minimization problems, gradient-based stochastic optimizers are used. The most common choice is stochastic gradient descent with momentum, but here the Adam [17] optimization algorithm is deemed a better option due to its adaptive nature (simplifies the tuning). Here, the word stochastic stands for stochastic gradients, usually referred to as mini-batch gradient. A mini-batch consists of evaluating the loss or its gradient on only a subset of the training data. Usually, the minimization problem is trained over epochs, as opposed to iteration steps. In one epoch, the training data is shuffled, divided into similar sized batches (default=128) and the optimizer takes each approximated gradient one at a time. Hence, one epoch is in fact $data\_size/batch\_size$ iterations.

As for the structure of function f_w, it includes multiple hidden layers between the input x and the output y. DNN layers can be connected in multiple ways (linear, convolutions, pooling, etc) and each neuron has an associated activation function (sigmoid, rectified linear unit, tanh, etc). A fully connected DNN (FNN) is composed of only linear layers. Convolutional neural networks (CNNs) are composed of convolutional, pooling and linear layers. Generally, CNNs require relatively little pre-processing compared to other algorithms but, due to their particular structure, these networks suffer from data over-fitting.

The main advantage of DNNs is detecting unusual and difficult behaviours of the data. However, it suffers from long training times, difficult implementation and low interpretability as opposed to simpler, less informative models. Moreover, when faced with simple data, it can easily over-fit the training set.

3.5. Error Metrics

The performance assessment process of ML algorithms requires providing an additional validation data set, different than the one exploited in the learning stage. Commonly, predictions from the trained ML model are compared against the validation data, to quantify the performance of the resulting model and to evaluate the effectiveness of the training strategy. To achieve these goals, one needs first to define a suitable metric to compare the model against.

The ice accretion problem investigated in this paper consists of a binary scenario already described in previous sections. Given that the superscript $*$ indicates the prediction from the ML algorithm, the performance of diverse ML models in retrieving the positive labels $+$ (i.e., if there is ice at a selected location) is assessed within our synthetic validation data set.

In binary classification problems, it is common to define *precision* as the fraction of true positives over the total number of positives (i.e, the sum of true positives and false positives) returned by the trained algorithm. That is, the percentage of correct predictions among the positives ones returned

$$\mathcal{P} = \frac{\{+^*\} \cap \{+\}}{\{+^*\}}. \tag{1}$$

In other words, a precision score of 100% means that every item labeled as positive effectively belongs to the positive class.

Similarly, *recall* is defined as the number of true positives divided by the total number of positives effectively included in the validation data set. Namely, the recall metrics provides information about the amount of items that should have been labeled as positive but were instead misclassified as negative and it reads

$$\mathcal{R} = \frac{\{+^*\} \cap \{+\}}{\{+\}}. \tag{2}$$

A recall equal to 100% points out that every positive item included in the validation set was identified by the ML model.

Each metric alone is only providing partial information. Precision alone does not allow to quantify the number of positive elements in the validation set that were not identified whereas recall misses indications about the false positives. Sometimes, precision and recall scores are merged into a single metric to provide a combined perspective on the ML algorithm performances. Here, precision and recall are assessed separately, in order to highlight different aspects of each of the considered ML models.

To compare the performance of the regression models against that of the classification ones, the continuous output from regressors is converted to a binary condition (1/0) through a round up operation. Namely, any output larger than zero is rounded up to unity. This procedure allows the application of the very same $\mathcal{P}/\mathcal{R}$ metrics adopted in binary classification problems.

For both the $\mathcal{P}$ and the $\mathcal{R}$, it is also defined an integral metric $\mathcal{F}$ to assess the gap of each algorithm w.r.t. the optimal behavior and to provide a quantitative

comparison between algorithm performances. Given that G stands for either $\mathcal{P}$ and $\mathcal{R}$, the integral metric reads

$$\mathcal{F} = \frac{\int_0^1 f'(s)ds}{\int_0^1 f''(s)ds}, \qquad f'(s) = \begin{cases} G(s) & \text{if } \; G(s) > 0, \\ 0 & \text{otherwise}, \end{cases}, \qquad f''(s) = \begin{cases} 1 & \text{if } \; G(s) > 0, \\ 0 & \text{otherwise}, \end{cases} \tag{3}$$

or, in the discrete form,

$$\mathcal{F} \approx \sum_{i=1}^{N_e} \frac{G_i}{\Delta s_i}, \qquad \text{for } \; G_i > 0, \tag{4}$$

being s the curvilinear abscissa, N_e the number of panel included in the surface grid meshing the airfoil and Δs_i the length of the single panel. Clearly, a very low integral metric $\mathcal{F}$ is an indication that the ML model is scoring very low $\mathcal{P}$, or $\mathcal{R}$, in a vast portion of the domain, at worse in the whole domain if the integral is equal to 0. On the other hand, values of $\mathcal{F}$ close, and ideally equal, to 1 indicate a good performance. Note that the metric is not defined if no ice is predicted over the entire profile. Indeed, this would lead to the undefined case 0/0 and hence the performance $\mathcal{F}$ can not be estimated.

Additionally, a statistical error metric is defined to assess how the trained ML models generalize to an independent sampling of the observation data set. The so-called *k-fold* cross-validation procedure is exploited to estimate the average performance of each ML model we train. In k-fold cross validation, the data set is randomly partitioned in complementary subsets of equal size k, with $k \leq n$ being n the total number of observation available. Note that, if $k = n$, we refer to the procedure as the leave-one-out cross-validation. Once the random partitioning is carried out, it is possible to consider k diverse combinations of training/testing data sets from the same set of observations i.e., $k-1$ subsets are used to train the model whereas the remaining data serve for validation purposes. Therefore, the selected ML model is trained k times, considering all the diverse training/testing sets combinations at separate times.

The overall performance of the selected ML model is then obtained by combining results from each of the k training processes, for instance through simply averaging the $\mathcal{P}$ and $\mathcal{R}$ metrics. In this paper, the average ($\bar{G}$) and the standard deviation (σ_G), where G stands again for either $\mathcal{P}$ and $\mathcal{R}$, are computed according to

$$\bar{G} = \frac{1}{k}\sum_{i=1}^{k} G_i, \qquad \sigma_G^2 = \frac{1}{k-1}\sum_{i=1}^{k}(G_i - \bar{G})^2. \tag{5}$$

The strength of this approach lies in the fact that all the observations (or the vast majority of them, depending on the value of k) are employed both in the training and in the validation process, so the final result is independent from the data partitioning. For this particular application, the cross-validation is done with $k = 10$ with equally-sized groups.

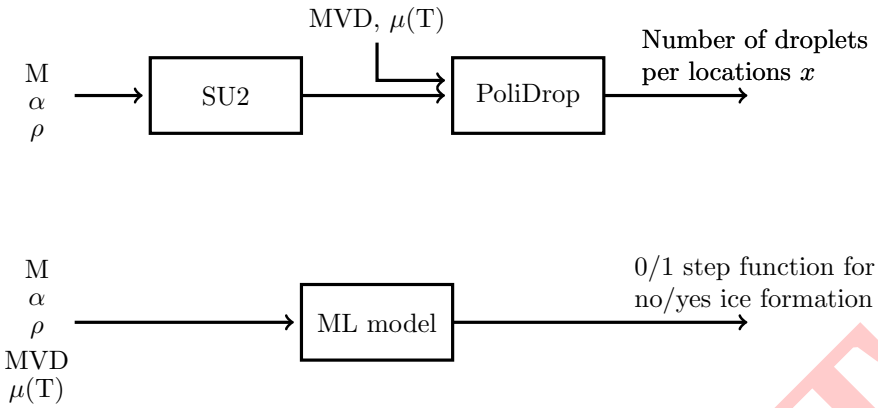


Figure 2: Comparison between the conventional and the ML learning approach. The ML approach takes significantly less time.

4. The Synthetic Data Set Generation

In this work, the classic NACA0012 airfoil is considered. The observations data set is generated using a computational framework that couples two different pieces of software. The upper part of Fig. 2 reports the flowchart of the computational framework with the inputs required by each block. First, the flow field around the airfoil is reconstructed using the open-source CFD solver SU2 [21]. The aerodynamic field is reconstructed under the inviscid and adiabatic flow assumptions, to reduce the computational cost of simulations. The inviscid flow assumption is justified by the large Reynolds number associated to tests. Viscous effects are limited to the thin boundary layer surrounding the airfoil and therefore play a minor role. Once the CFD step is carried out, the numerical solution of the flow field is passed to the PoliDrop, an in-house Lagrangian particle tracking software developed at Politecnico di Milano [3]. Differently than the CFD model, which does not account for viscosity, Polidrop accounts for viscous effects acting on droplets. Namely, the air viscosity is computed locally based on the temperature field available from the CFD solution and by taking advantage of the Sutherland's law.

The sequential evaluation of the CFD and the particle tracking models allows to retrieve an estimation of the distribution of water collected over the airfoil. In this work, a synthetic data set is generated including 1000 observations. Each observation corresponds to a deterministic evaluation of the full computational model obtained considering a random combination, sampled uniformly and independently from the flight envelope, of the five input values (M, α, ρ, $\mu(T)$, MVD).

4.1. Numerical set up

The CFD solution of the flow field around the airfoil is obtained using an Approximate Riemann Solver (ARS) of Roe type employing a Monotonic Upwind

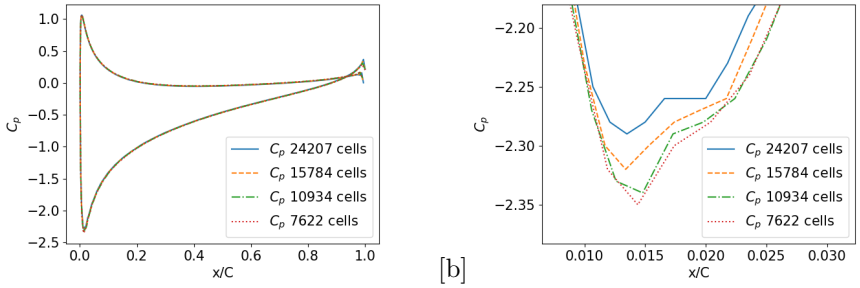


Figure 3: Sensitivity analysis of the aerodynamic pressure coefficient C_p w.r.t. selected grid. (a) C_p on the pressure and suction sides; (b) Enlargement of the C_p distribution in the close proximity of the leading edge, on the suction side.

Scheme for Conservation Laws (MUSCL) [26] coupled to the Venkatakhrishnan flux limiter. The solution convergence criteria was the reduction of the residuals by 5 orders of magnitude from their initial value.

Unstructured grids of triangular elements are built using an in-house mesh generator tool implementing an advancing-front/Delaunay algorithm [9]. A sensitivity analysis, based on four meshes of increasing spatial resolution, led to the definition of the baseline grid employed to run all the simulations needed to generate the synthetic data set. The results of the sensitivity study are reported in Fig. 3(a-b).

According to the aforementioned analysis, the coarsest mesh was selected, counting 7.6k points. Though the CFD solutions relative to meshes of higher resolution are slightly different in some regions, discrepancies are minimal and they do not justify the significant higher computational cost associated to finer grids. Moreover, this work is a proof of concept aimed at exploring the potential of ML techniques for ice-accretion applications. Since the goal is not to reproduce the physics of the problem with a high level of fidelity, but rather to generate a synthetic data set to be exploited for the training of ML model, a certain loss of accuracy is accepted as long as the general behavior is fairly reproduced. At the same time, a rough approximation of the actual physics introduces errors and uncertainties that can mimic the role of noise in real experimental measurements.

Sensitivity studies were also carried out to define the appropriate integration time-step for the reconstruction of particle trajectories, using the PoliDrop solver, see Fig. 4(a). According to the analysis, a time-step corresponding to 10^{-5} [s] was selected. Furthermore, studies were performed to assess the minimum number of droplets needed to obtain a converged solution. Fig. 4(b) reports the latter analysis obtained considering a 10^{-5} [s] time-step. Here, the ratio between the local density of water (over the surface elements) and the density of the cloud, in other words the collection efficiency, is reported. That is, the number of droplets impinging on the surface element divided by the number of droplets per unit length characterizing the cloud. According to the study, a

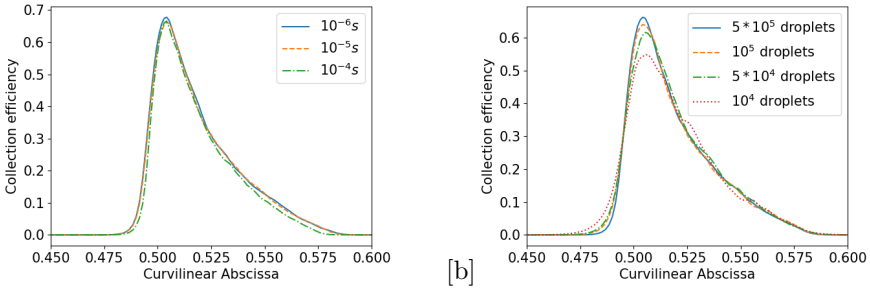


Figure 4: Sensitivity analysis of the airfoil collection efficiency w.r.t selected parameters. (a) Integration time step for the particle trajectory reconstruction. (b) Number of particles injected in the flow field.

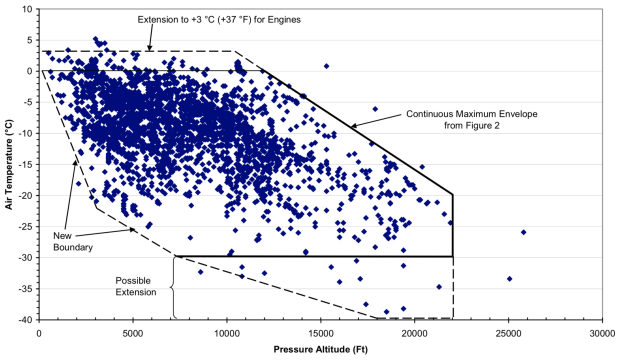


Figure 5: Temperature vs altitude flight envelope from [16]. Each point represents an observed icing event occurring when flying in layer clouds at the corresponding temperature and altitude.

total number of 50k droplets was chosen.

Note that the curves reported in Fig. 4(a-b) were regularized by means of the Kernel Density estimation algorithm, for smoothing purposes. This also applies to the whole synthetic data set. Nevertheless, the same conclusions regarding the sensitivity of the collection efficiency curve to the selected time-step and to the number of droplets hold for the non-smoothed data.

4.2. Envelope Characterization

Over the past 40 years, the icing envelopes reported in [18] have been widely exploited to define the meteorological scenarios considered in the design of in-flight protection equipment for aircraft. In this work, it is considered the updated icing envelopes reported in [16]. The envelopes help setting the boundaries of our investigation and delimit the conditions faced by the aircraft during the typical mission. Based on these boundaries, see Fig. 5, the air temperature and density ranges can be set.

	T [C]	AoA []	M [-]	MVD [μm]	ρ [Kg/m^3]
Upper bound	0.0	0.8	10	50^{-6}	0.400
Lower bound	-40	0.3	0	10^{-6}	1.225

Table 1: Table reports the upper and lower bounds of the uniform probability distributions employed to sample the random inputs for the generation of the synthetic data set.

As described in Sec. 2, each simulation considers a droplet distribution with null standard deviation, meaning that the droplet size is uniform in the domain. According to the considered envelopes, the minimum MVD value is 10 [μm] whereas, to define the upper limit, a worse case scenario approach is adopted by considering 50 [μm], the maximum value reported in the prevailing icing regulation. As reported in [16], droplets of larger diameter usually precipitate. However, the methodology presented in this paper can be applied considering any values for the MVD. The Mach air speed value spans from the approaching to the cruising flight segments of a typical commercial aircraft. In the same manner, also the angle of attack of the airfoil is considered in between the neutral position and a high lift configuration typical of the runway approach phase. Summarizing, the synthetic data are generated by randomly sampling the 5 inputs from 5 independent uniform probability distribution, see Tab. 1.

In this scenario, a data point consists in the collection efficiency corresponding to a random combination of five random parameters i.e., the inputs of our computational framework previously described. Note that the prediction of an accurate value of the collection efficiency is not relevant. Indeed, the sought information is the true/false condition that points out the presence (or the absence) of water at a selected element. Given the underlying rime ice hypothesis, the presence of water necessarily implies the formation of ice crystals. Nevertheless, we still consider the actual value of the collection efficiency in regression problems, in the hope of improving model predictions performance. Indeed, in this way we train the algorithm on a continuous problem rather that on a discrete set of data. In regression, a discrete sets of training data may lead to inaccuracies because values very close to, but larger than, zero must still be classified as belonging to the true ice condition.

5. Results

5.1. Decision Trees and Ensemble Methods

In the quest for finding a suitable machine learning algorithm that fits the purposes of this paper, the first choice was to start by studying the application of decision trees and ensemble methods. From Sec. 3, decision trees can be a powerful way to generate cheap and efficient models for classification and regression.

In this case, the binary classification problem was carried out as well as the regression of the collection efficiency with the five input parameters defining the local flow conditions. Regression results are then converted to binary values of

1 and 0 depending whether the predicted efficiency is other than 0 or if it is 0, respectively. The binary values obtained in regression are the ones compared against the ground truth of the synthetic data to compute the different metrics considered. Regression is expected to provide improved performances since it approximates a smooth curve in place of just binary values (i.e., a step function). All the ensemble methods have been trained with a sufficiently large number of decision tree estimators (100) which gives the best possible training.

The left half side of Fig. 6 reports a comparison of the $\mathcal{P}$ metric performance of decision trees and ensemble methods considering different depths. In all plots, the shady curve denotes the probability of having ice at every location over the airfoil computed w.r.t. the whole conditions included in the synthetic data set. In other words, the curve plots the number of times ice is detected on a panel over the total number of data points sampled from the envelope. This information helps assessing the performance of the algorithms since the precision metric must drop to zero at locations in which no ice formation is ever predicted for the whole envelope. Indeed, whenever the panel never collects water, the corresponding location over the airfoil is undoubtedly free of risk for ice formation. In such locations, $\mathcal{P}$ is always zero since there are no real positive items to retrieve and $\{+^*\} \cap \{+\} = \emptyset$. Therefore, an acceptable estimator is the one that gives a jump from 0 to 100 % as soon as the amount of collected water differs from zero. That is, the algorithm is supposed to classify a location as prone to ice formation as long as it collects any water, at any point of the whole envelope.

One can then see from the pictures on the left half side of Fig. 6 that the different algorithms manage to get closer or further from this ideal scenario. Particularly, classifiers seem to improve their precision as the depth of the trees is increased. Here, the maximum considered depth is equal to 9, increasing this parameter may lead to overfitting issues. Indeed, a depth of 10 layers would result in a number of classification groups larger than the actual data points, therefore making the models lose generality and significantly dropping the algorithm performance for unseen data classification.

On the right half side of Fig. 6, the $\mathcal{P}$ performance of decision trees in the regression problem is presented. The trend of improving predictions along with tree depth is observed also for regression and the very same considerations apply.

The comparison between the precision score obtained in the context of a binary classification problem and the one from regression prior to metric evaluation is evident from Fig. 6. One can conclude that, in most cases, binary classifiers based on decision trees perform better than regressors in terms of precision. The recall metric $\mathcal{R}$ for both the classification and the regression problems is reported in Fig. 7, respectively on the left and on the right hald sides. In binary classification, recall improves as the depth of the trees increases whereas an inverted trend for regression is observed. Note that, in any case, the bagging ensemble performance appears to change fairly little despite the number of levels employed. This can be due to the fact that for single trees the variation with depth from 3 to 9 is also very small. One can say that the performance of the trained models can already achieve good scores with small

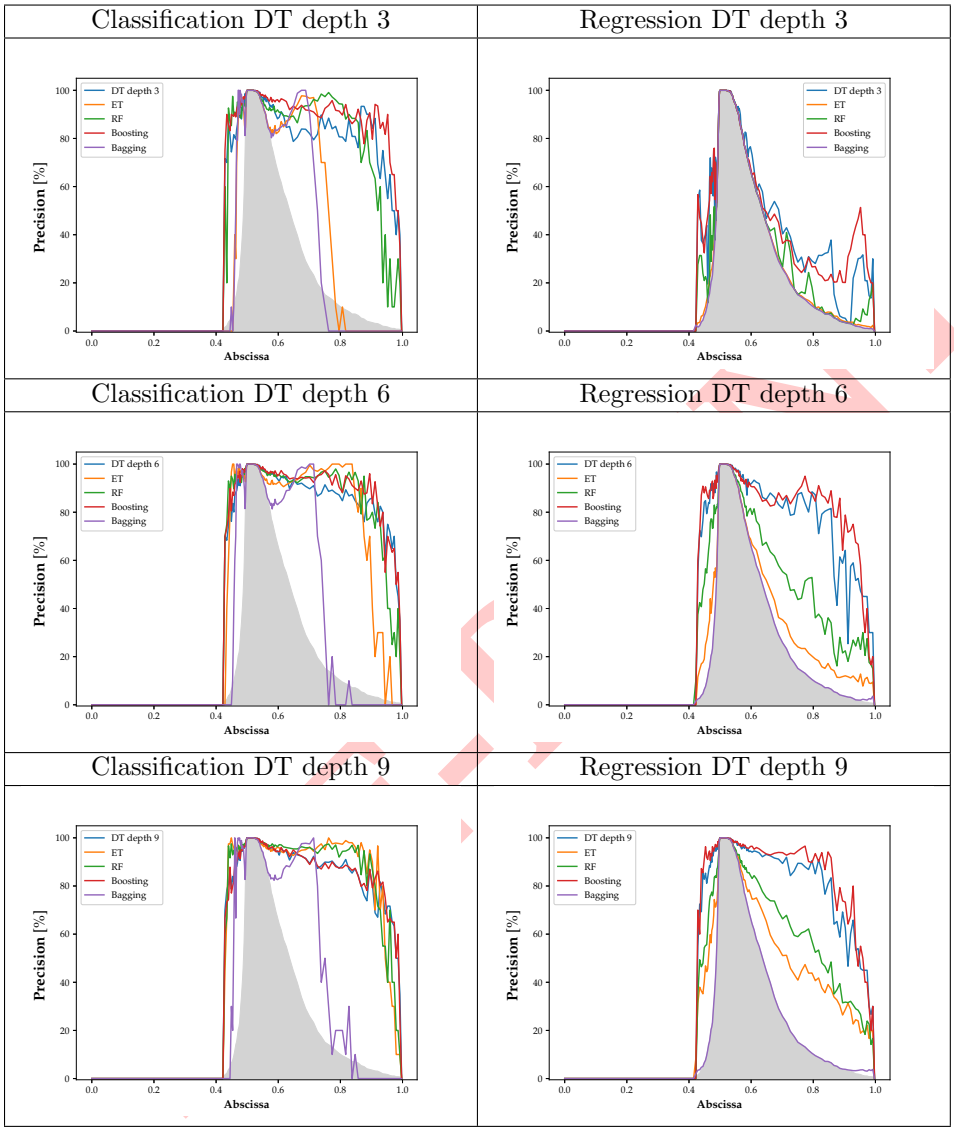


Figure 6: Decision Trees and Ensemble Methods precision comparison for binary classification and regression prior to error metric evaluation

Table 2: Comparison of the precision and recall metric integrals for decision trees and ensemble methods in binary classification

Metric	Decision Tree	Extremely Randomized Trees	Random Forest	Boosting	Baggin
Precision	0.86	0.88	0.88	0.89	0.49
Recall	0.82	0.72	0.75	0.84	0.37

depths of 3. The ensemble of these trees in a bagging algorithm presents the same trend as single trees with even more reduced variability as it is the average performance what it is evaluated.

Generally, $\mathcal{R}$ is larger for regression than for classification confirming that the former approach helps improving predictions. Nevertheless, regressors scores very poorly in the precision metric, indicating that they are over predicting the amount of positives in a given location on the airfoil. In other words, this means that regression models tend to be more pessimistic in their predictions thus resulting into a very conservative approach for icing applications. Though this over protection would make regressors very safe ML models to fly with, it would defeat the purpose of designing an efficient control strategy for the anti-ice system.

To better assess the performance of the different algorithms in terms of the $\mathcal{P}$ and the $\mathcal{R}$ metrics, the normalized integrals of the metric curves are compared along the curvilinear abscissa, see Tab. 2. All the integrals are evaluated for decision tree models with depth of 9 except for boosting, which shows a better performance for depth 6. In Tab. 2, one can see that boosting with maximum tree depth of 6 is the best performing algorithm, followed closely by decision tree of depth 9. On the other hand, bagging is by far the worst performing algorithm.

In Fig. 8, the performance of boosting is assessed in terms of the statistical error metric resulting from the application of the k-fold cross-validation with $k = 10$. Overall, this is the best that can be done by means of the considered decision trees and ensemble methods. On average, the precision and recall mean performance scores are quite high, but both metrics show a large variability w.r.t. the training data. In particular, this occurs for values of s larger than 0.7.

In general, results show that all the considered performance decay in the region included in between $s = 0.7$ and $s = 1$. In this area, corresponding to the aft section of the airfoil pressure side, the presence (or absence) of ice is utterly dominated by the value of the Angle of Attack (AoA) α and by the Mach number. Indeed, for neutral AoA the formation of ice is practically limited to the airfoil leading edge. This causes the data relative to the mentioned locations to be quite scattered for certain regions of the flight envelope. Possibly, the inclusion of additional points might help improving the DT performance.

A more advanced approach combining boosting technique to other ML models in regions of the airfoil characterized by large standard deviations might also help increasing our confidence in predictions, but this is left for future investi-

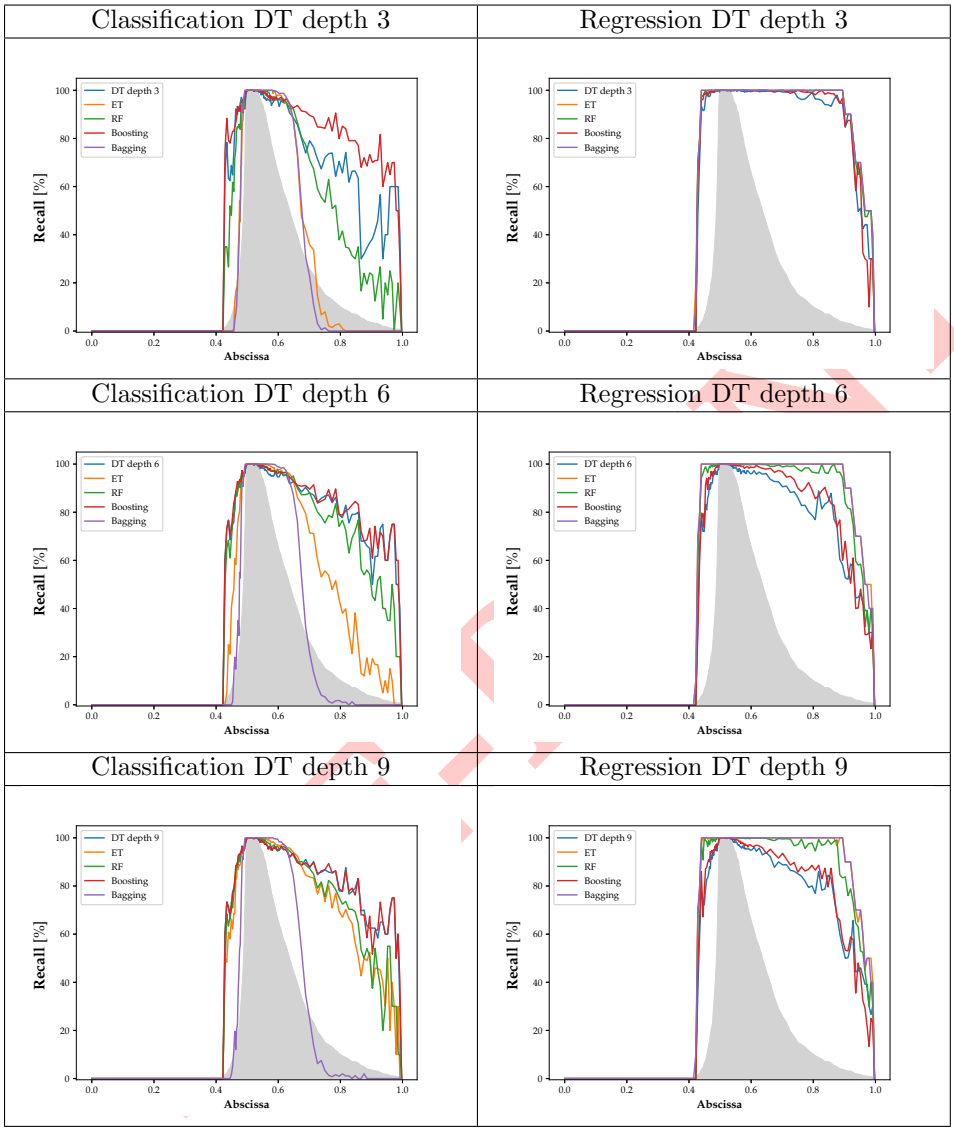


Figure 7: Decision Trees and Ensemble Methods recall comparison for binary classification and regression prior to error metric evaluation

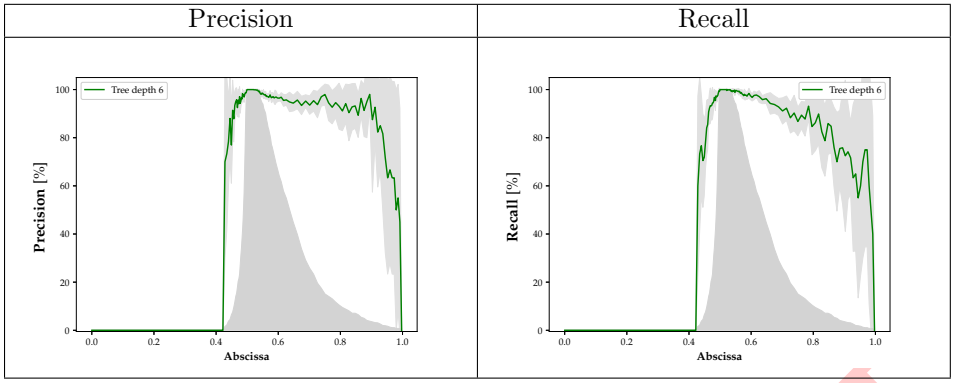


Figure 8: Statistical error metrics, score average and standard deviation, related to the boosting method implementing decision trees of depth 3 for binary classification problems.

gations.

5.2. Support Vector Machines

In this section the performance of SVMs for the classification problem is analysed. A SVC is trained for each airfoil panel i.e., for a fixed location along the curvilinear abscissa, to predict the presence of ice according to the inputs provided.

First, the performance of SVCs is assessed considering a linear kernel ($p = 1$). Fig. 9 reports, respectively on the left and on the right half sides, the $\mathcal{P}$ and the $\mathcal{R}$ metrics. Results show that linear SVCs score poorly, especially when compared to ML models considered in previous sections, in the region corresponding to the aft part of the airfoil pressure side.

Then, the performance of non-linear SVCs i.e., SVCs employing a quadratic kernel ($p = 2$), are assessed. Fig. 10 reports the analysis of the precision and recall metrics for diverse values of γ and C. In particular, non-linear SVCs are found to perform rather similarly w.r.t. linear ones. The linear kernel gives good performances for $C = 10$, see Fig. 9. Although integrals are similar for $C = 20$, there is more variation, which decreases a potential generalisation of this algorithm to other testing sets. Eventually, SVCs models are trained based on the RBF kernel. The analysis of the $\mathcal{P}$ and the $\mathcal{R}$ scores for diverse kernel parameters is reported in Fig. 11. Clearly, the RBF SVCs deliver very poor performances: despite a significant variation of the C and γ, the models return quite low scores, especially in the critical area in the proximity of the airfoil trailing edge. According to the results shown so far, a good performing algorithm is, for examples, SVC based on a polynomyal kernel of order 2, with $C = 0.1$ and $\gamma = 1$. In Fig. 12, the average metrics $\bar{\mathcal{P}}$ and $\bar{\mathcal{R}}$, computed via the k-fold approach, are reported respectively on the left and on the right hand side. The two quantities are characterized by a very limited variability indicating a loose dependency on the training data set of choice. Eventually, the integrals of the precision and recall metrics are computed for the linear and quadratic

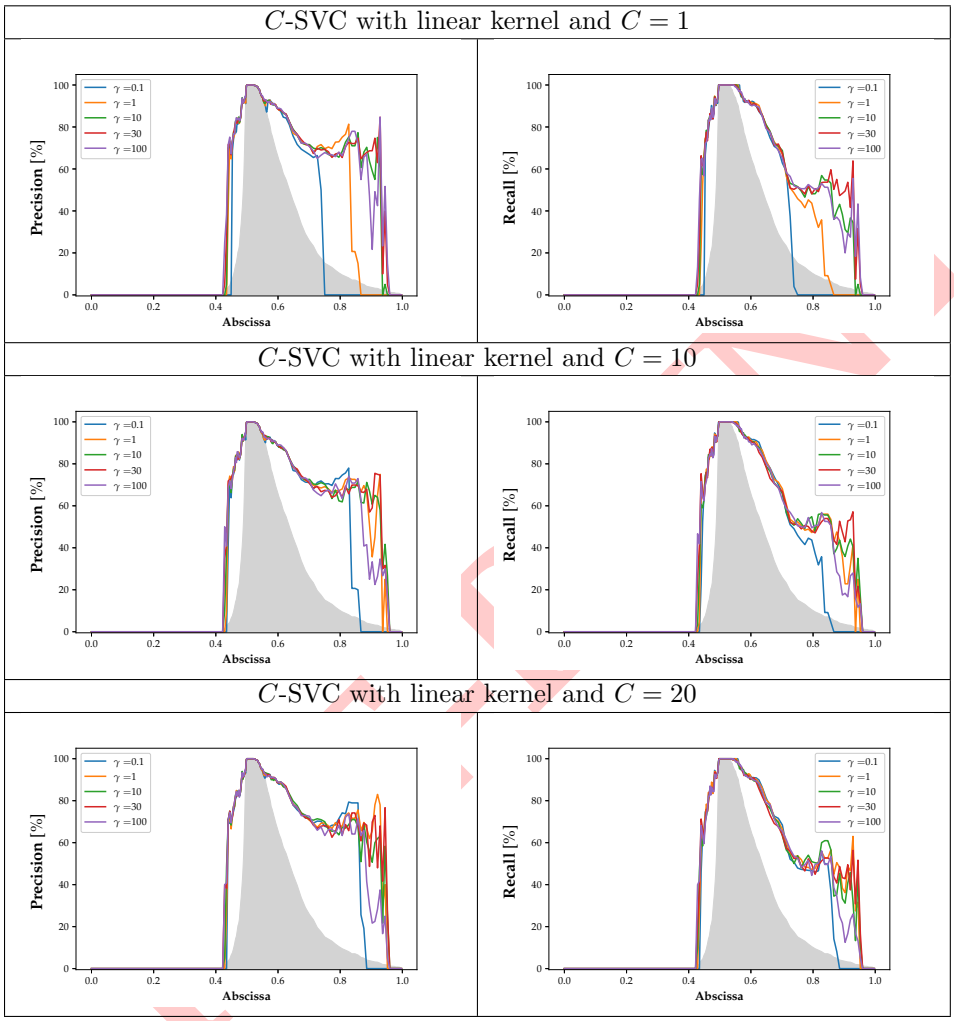


Figure 9: Performance analysis of C-Support Vector Classifiers with linear kernel ($p = 1$) for binary classification.

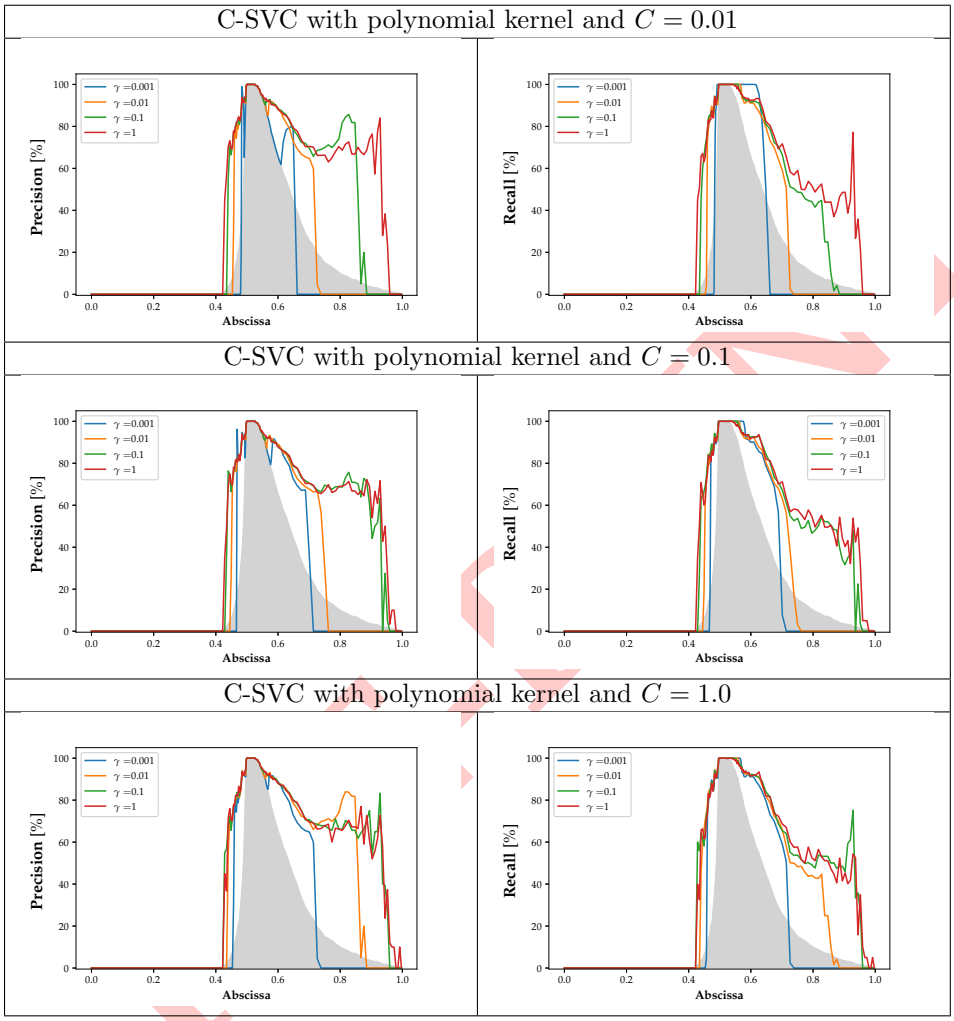


Figure 10: Performance analysis of C-Support Vector Classifiers with polynomial kernel ($p = 2$) for binary classification.

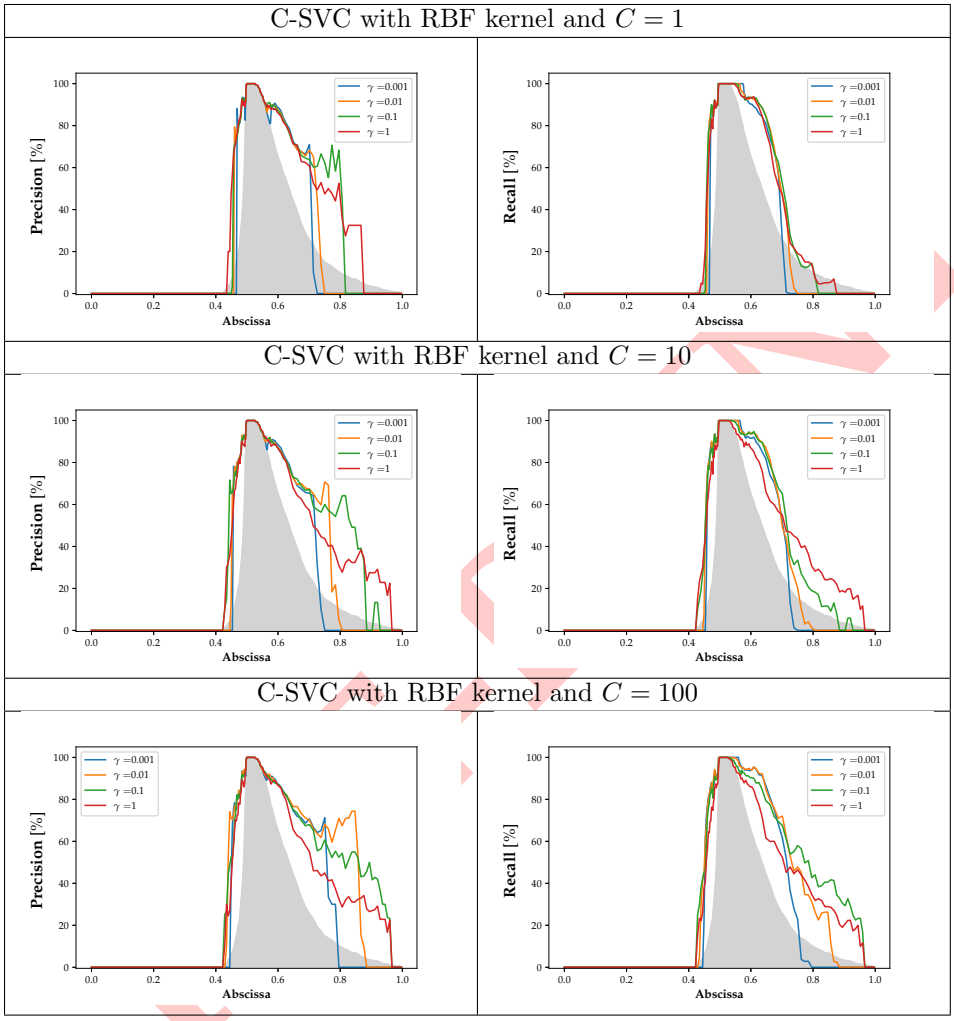


Figure 11: Performance analysis of C-Support Vector Classifiers with RBF kernel for binary classification.

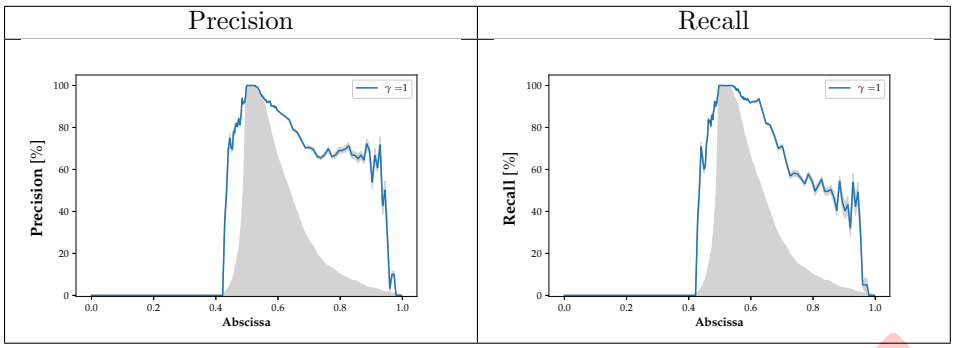


Figure 12: Statistical error metrics, score average and standard deviation, related to the quadratic C-SVC with $C = 0.1$ and $\gamma = 1$.

Table 3: Comparison of the precision and recall metric integrals for C-SVC. The right handside table gives the results for linear kernel with $C = 10$. The left handside one gives results for quadratic kernel with $C = 0.1$.

Metric	$\gamma = 10$	$\gamma = 30$
Precision	0.68	0.69
Recall	0.61	0.62

Metric	$\gamma = 10$	$\gamma = 30$
Precision	0.66	0.7
Recall	0.6	0.63

C-SVCs, and they are reported in Tab. 3 for comparison. Unsurprisingly, the integral scores increase, once again asymptotically, up to a limiting value.

5.3. k-Nearest Neighbors

The performance of the k-NN algorithm for the classification and the regression problems are here assessed. The ML model is trained for different values of the user-defined parameter k, see Sec. 3.3.

Fig. 13 reports parametric studies, w.r.t. k, concerning all the considered metrics, for the binary classification problem. Once again, results show that both the precision and the recall scores decrease significantly with k, in particular in the region included in between $s = 0.7$ and $s = 1$. Again, that is possibly due to the fact that the presence of ice in panels located in such position, corresponding to the aft region of the airfoil pressure side, suffers from a strong dependency on the angle of attack. In such region, the presence of ice is quite rare, occurring only at very large α, and hence the related data will include a large number of negative labels. Since the algorithm assigns a label to a query according to the majority of the neighboring elements, in regions not prone to ice accretion the model will likely return a negative label even though it is supposed to make a positive prediction. In other words, a single element belonging to class A surrounded by a multitude of B items will labeled incorrectly. Large values of k will make the model account for more neighbours, thus worsening the issue in case of highly scattered data. Therefore, the ML model assigns a negative value also to a positive element compromising the overall model performance to the point that a very small k provides better scores. Possibly, a

data set enriched with additional points sampled in this portion of the envelope might lead to notable improvements.

Consistently with the parametric studies presented earlier, the integral metric $\mathcal{F}$, reported in Fig. 13, confirm a dramatic general loss of performance with increasing k. The stochastic metrics computed via the 10-fold approach, reported for $k = 1$ only, point out poor averaged performance in a large part of the domain interested by ice formation. At the same time, the variability of both curves i.e., the standard deviation w.r.t. the diverse training sets used for cross validation, is significantly large in the critical area past $s > 0.7$.

On the other hand, better performance was attained considering the implementation of the k-NN algorithm for the regression problem. Fig. 14 reports the metric scores, which indicate a significantly better performance in terms of recall. In particular, $\mathcal{R}$ increases with k and also the related integral is shown to converge asymptotically to a steady value. On the other hand, the integrated $\mathcal{P}$ metric decreases with k, thus forcing a trade-off between precision and recall when selecting the parameter k.

Moreover, stochastic metrics related to recall are also very encouraging, as the score is the highest in terms of average and it is characterized by a very low variability. Additionally, the metrics seems to score good even in the aft portion of the airfoil. Nevertheless, the algorithm still performs poorly in terms of average precision. The stochastic metrics again return an indication of a poor average precision and a large variability in the region past $s = 0.7$. Possibly, at a large AoA the algorithm over estimates the number of positive items therefore scoring low in precision. As pointed out before, this would lead to a very conservative, but quite inefficient, strategy.

Overall both classification and regression produce similar performances for $k = 1$. Nevertheless the k-NN ML models, in particular the classifiers, are found to be poorly predictive and, therefore, they are deemed not suitable for ice accretion applications.

5.4. Neural Networks

Finally, neural nets, which are considered a more sophisticated algorithm than those presented above, are deployed. As for implementation, the Pytorch library was exploited for the elaboration of neural nets and the wrapper sktorch was included to bridge scikit-learn to Pytorch.

In this section, the metrics for DNNs generally indicate very poor performance. Moreover, DNNs training time is much longer and require time-consuming hyperparameter search. Furthermore, taking into account the relative small size of the input and output, a neural network is much likelier to overfit or train to poorly generalizing models. Two types of neural nets are considered: FNN and CNN, as defined in Sec. 3.4. The FNN has two hidden linear layers of 10 neurons and activations functions of ReLU and sigmoid type in the first and second layer, respectively. The FNN was chosen according to deep learning intuition such that it had enough neurons and connections to capture the behaviour of the data. The CNN model was deployed including

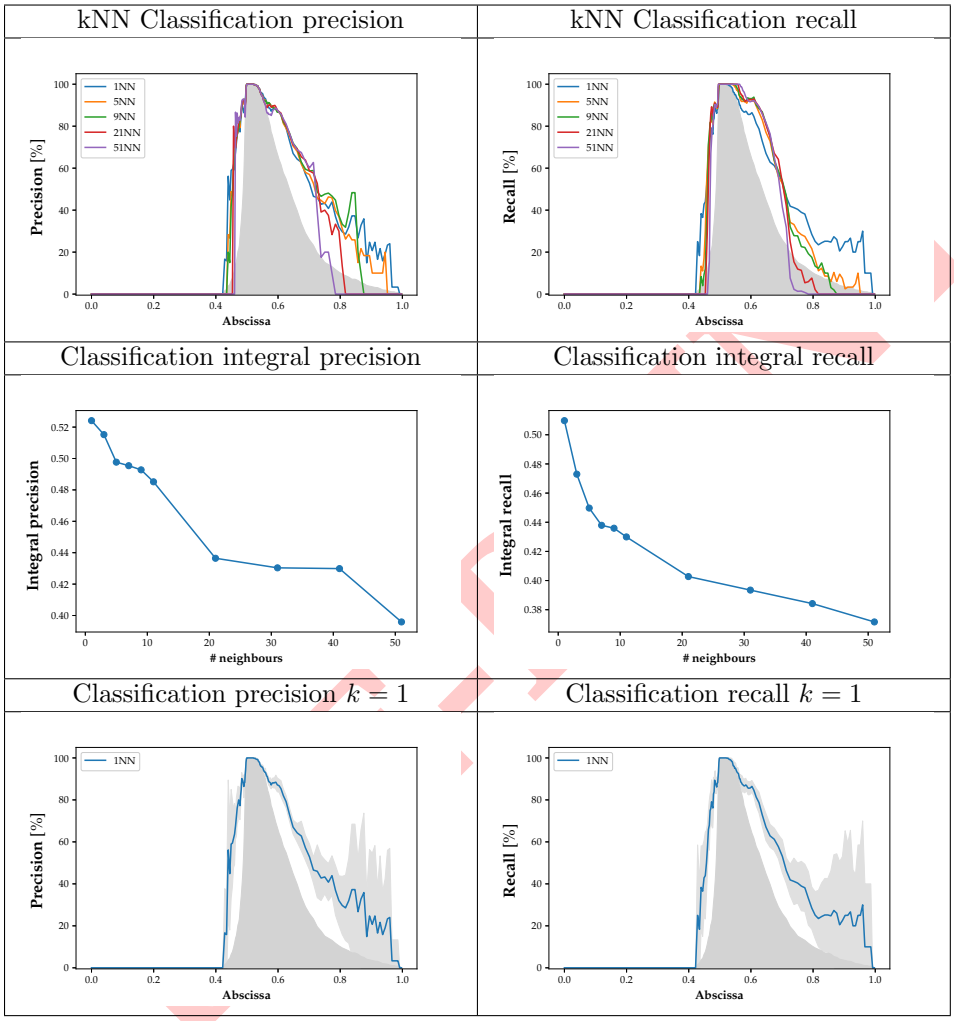


Figure 13: Performance analysis of k-NN models for binary classification.

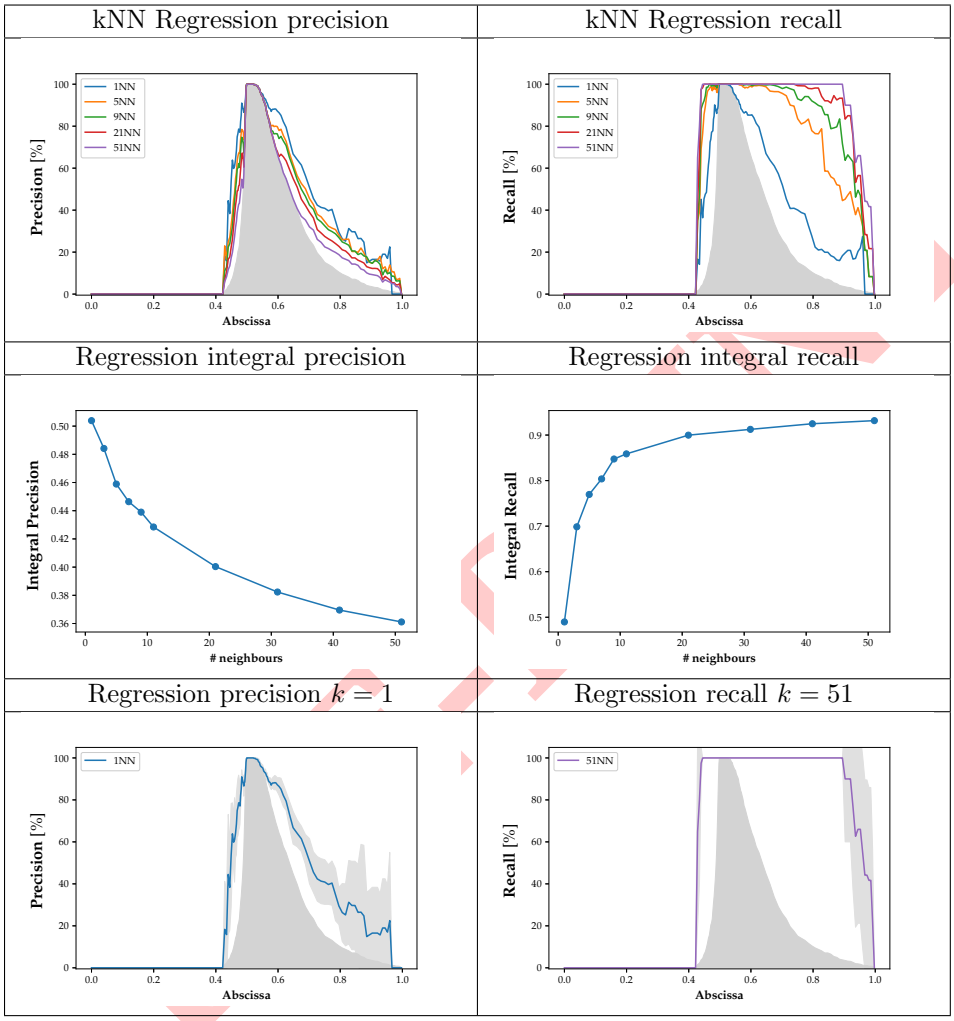


Figure 14: Performance analysis of k-NN models for regression.

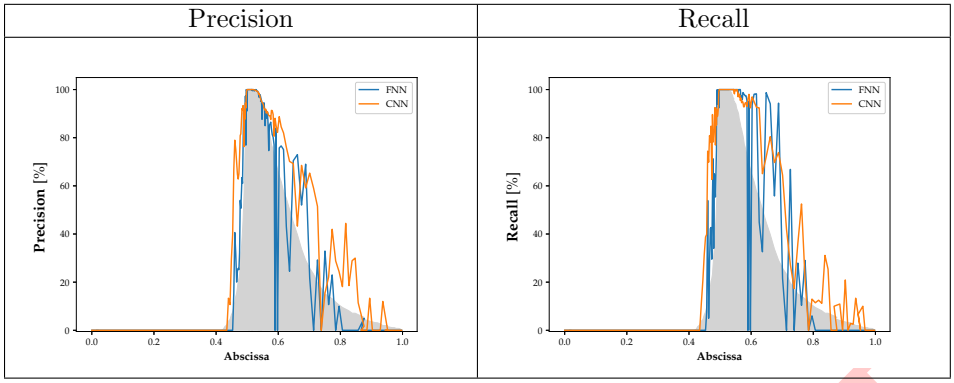


Figure 15: Discrete neural network precision and recall comparison for binary classification

two convolutions layers and two hidden linear layers with standard amounts of neurons and connections.

First of all, for each surface location, the same model is trained, obtaining several networks with different parameters. At the training stage, the loss and optimizer are implemented as described in Sec. 3.4. Moreover, the number of epochs is 50 and the initial learning rate is 0.01 (the learning rate is the multiplier for the gradient descent step on the optimizer). In Fig. 15, the performance of the algorithms is similar in both cases, it presents many peaks. This could be caused due to the limited amount of training data for neural nets and that could potentially lead to overfitting, having poor performance. In addition, during the training stage, the model parameters can find a model with a local minimum of the error rather than the global one. Considering the total score presented in the Tab. 4 the performance of CNN is better than FNN. This can significantly compromise their accuracy and prediction performance.

Then, a unique model was trained, considering the value of the curvilinear abscissa as an input parameter rather than one model per area. FNN and CNN are adapted accordingly to take in this extra parameter so the training paradigm remains unchanged. This data arrangement, rather than training one model for each location, seems more appropriate which is reflected in the increased performance in Fig. 16. It can then be concluded that having neighbooring information can be informative in regards to ice prediction, which would be seems to agree with the real world phenomena. Owing to the complexity of CNN, good hyperparameters of the training stage are harder to find than any other algorithm. This can be due to the creation of a new loss surface with multiple local minima when introducing the curvilinear abcissa variable. It is reasonable to assume good hyperparameters exist and the correct choice would allow AllCNN to exihibit the better behaviour.

The overall score of Neural Nets is lower when compared to simpler models. Due to their complexity and the apparent simplicity of the model, the use of Neural Nets is not the best option in this problem, since the training is hard and the results produced are not worth the effort. The overall results are present in

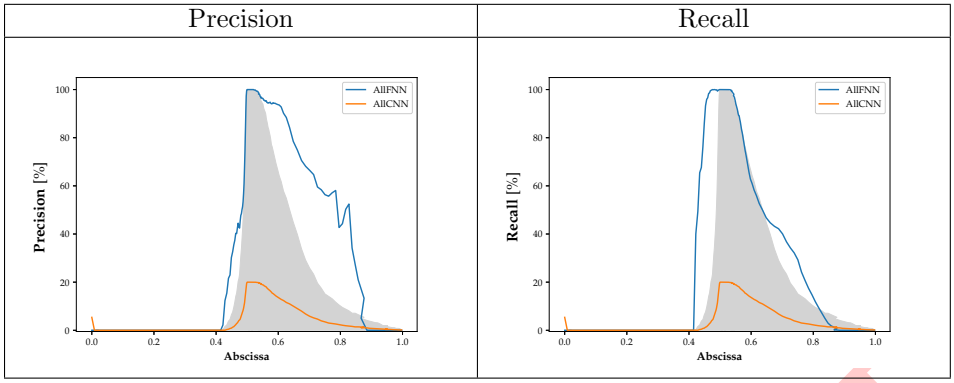


Figure 16: Abcissa continuous neural network precision and recall comparison for binary classification

Table 4: Comparison of the precision and recall metric integrals for different neural networks in binary classification

Metric	FNN	CNN	AllFNN	AllCNN
Precision	0.31	0.44	0.51	0.07
Recall	0.35	0.46	0.40	0.07

Tab. 4. Although CNN exihbits the best behaviour for recall and AllFNN the best precision, the average of the two metrics is similar. Since roughness is not desirable, as small vsariations along the curvilinear abcissa shouldn't change greatly the model metrics, the best model is AllFNN. It is plotted in Fig. 17 with the variance arising from cross-validation.

6. Conclusions

The goal of this contribution is to evaluate the potential of ML techniques in the context of ice accretion in aeronautical applications. It has been presented here a proof of concept for the future real-time management of ice protection systems.

In this work, we rely on a synthetic data set for the training of ML models. This is to avoid dealing with issues related to experimental measurements such as inaccuracies or incompleteness. In the future, synthetic data are expected to be substituted with measurements from real experiments.

A comparison between performance of different binary classification and regression algorithms shows that regressors usually return improved, though too conservative, icing predictions. For this reason, binary classification offers a better scenario where ensemble methods, consisting of decision trees as base estimators, score the best among all the considered algorithms. In this case, both the precision and recall metrics score reasonably well.

Almost all the considered algorithms perform poorly in predicting the ice formation in the aft portion of the airfoil pressure side, namely for values of the

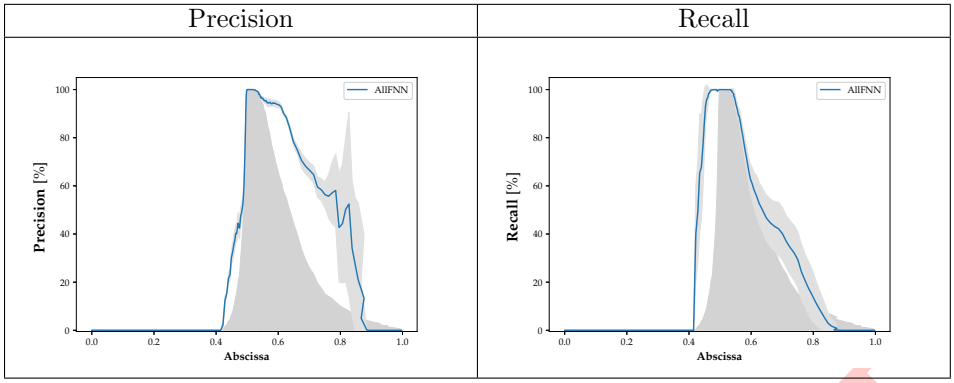


Figure 17: Statistical error metrics, score average and standard deviation, obtained with the best performing neural network.

curvilinear abscissa included in between 0.7 and 1. The inclusion of additional data, especially concerning flights with large AoA, Mach number and MVD, might help improving performances in such areas, to ultimately increase the confidence in this approach.

Next, the model could be extended to 3D wings, generating synthetic data for impingement in a similar way as described throughout the paper. In addition, the identified uncertainties concerning the measuring of the input data must be addressed. In this case, ML models could be trained to predict the probability of ice formation rather than a 1/0 value. Towards the implementation of the system into aircraft, test campaigns are crucial to produce real training and testing data sets.

The advantage of using ML algorithms in this context lies in the fact that, once the algorithms are trained, predictions can be obtained at an insignificant computational cost. Indeed, it takes approximately 20 minutes to carry out a complete simulation of a two-dimensional airfoil whereas an almost immediate evaluation is possible using ML tools. Furthermore, the gap is destined to explode when dealing with a fully three-dimensional aircraft configuration. Therefore, the reduction of the computational time granted by ML techniques in predicting ice formation makes real-time applications a closer reality.

Acknowledgments

This work is fully funded by the European Commission H2020 programme, through the UTOPIAE Marie Curie Innovative Training Network, H2020-MSCA-ITN-2016, Grant Agreement number 722734. The authors would like to thank Mr. Tathagata Basu and Dr. James Scoggins for their valuable insights.

[1] Skorch documentation. `https://skorch.readthedocs.io/en/stable/#`. Accessed: 2019-05-14.

[2] D Baumgardner, JL Brenguier, A Bucholtz, H Coe, P DeMott, TJ Garrett, JF Gayet, M Hermann, A Heymsfield, A Korolev, et al. Airborne instruments to measure atmospheric aerosol particles, clouds and radiation: A cook's tour of mature and emerging technology. *Atmospheric Research*, 102(1-2):10–29, 2011.

[3] Tommaso Bellosta. A Lagrangian 3D Particle Tracking Solver for In-Flight Ice Accretion. Master's thesis, Politecnico di Milano, Italy, 2019.

[4] Colin S Bidwell and Mark G Potapczuk. *Users manual for the NASA Lewis three-dimensional ice accretion code (LEWICE 3D)*, 1993.

[5] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, Aug 1996.

[6] D. Cireşan, U. Meier, and J. Schmidhuber. Multi-column Deep Neural Networks for Image Classification. *ArXiv e-prints*, February 2012.

[7] Dan Cireşan, Ueli Meier, Jonathan Masci, and Jürgen Schmidhuber. Multi-column deep neural network for traffic sign classification. *Neural Networks*, 32:333 – 338, 2012. Selected Papers from IJCNN 2011.

[8] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sep 1995.

[9] D Dussin, Marco Fossati, Alberto Guardone, and Luigi Vigevano. Hybrid grid generation for two-dimensional high-reynolds flows. *Computers & Fluids*, 38(10):1863–1875, 2009.

[10] Karen J Finstad, Edward P Lozowski, and Lasse Makkonen. On the median volume diameter approximation for droplet collision efficiency. *Journal of the atmospheric sciences*, 45(24):4008–4012, 1988.

[11] George Forman and Ira Cohen. Learning from little: Comparison of classifiers given little training. In Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi, editors, *Knowledge Discovery in Databases: PKDD 2004*, pages 161–172, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[12] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[13] Giulio Gori, MARTA Zocca, M Garabelli, Alberto Guardone, and Giuseppe Quaranta. Polimice: A simulation framework for three-dimensional ice accretion. *Applied Mathematics and Computation*, 267:96–107, 2015.

[14] W. G. Habashi. Recent advances in cfd for in-flight icing simulations. *Japan Society of Fluid Mechanics*, 28(2):99–118, 2009.

[15] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated, 2014.

[16] Richard K Jeck. Advances in the characterization of supercooled clouds for aircraft icing applications. Technical report, Office of Aviation Research and Development, Federal Aviation Administration, 2008.

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[18] W. Lewis and A.R. Jones. Recommended values of meteorological factors to be considered in the design of aircraft ice-prevention equipment. Technical report, NACA, 1949.

[19] Warren S. McCulloch and Walter Pitts. Neurocomputing: Foundations of research. chapter A Logical Calculus of the Ideas Immanent in Nervous Activity, pages 15–27. MIT Press, Cambridge, MA, USA, 1988.

[20] Tim G Myers. Extension to the messinger model for aircraft icing. *AIAA journal*, 39(2):211–218, 2001.

[21] Francisco Palacios, Juan Alonso, Karthikeyan Duraisamy, Michael Colonno, Jason Hicken, Aniket Aranake, Alejandro Campos, Sean Copeland, Thomas Economon, Amrita Lonkar, et al. Stanford university unstructured (su2): an open-source integrated computational environment for multi-physics simulation and design. In *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, page 287, 2013.

[22] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[24] Gary A Ruff and Brian M Berkowitz. *Users manual for the NASA Lewis ice accretion prediction code (LEWICE)*, 1990.

[25] Farooq Saeed, Sylvain Gouttebroze, and Ion Paraschivoiu. Modified canice for improved prediction of airfoil ice accretion. In *8th Aerodynamic Symposium of 48th CASI Conference, Toronto, Canada, Apr*, 2001.

[26] Bram Van Leer. Towards the ultimate conservative difference scheme. v. a second-order sequel to godunov's method. *Journal of computational Physics*, 32(1):101–136, 1979.

[27] PG Verdin and CP Thompson. Automatic multi-stepping approach for ice predictions. *IASME Transactions, Issue1*, 2, 2005.

[28] Zhao Zhan, Wagdi Habashi, and M Fossati. Local reduced-order modeling and iterative sampling for parametric analyses of aero-icing problems. *AIAA Journal*, 53:1–12, 04 2015.

[29] Zhao Zhan, Wagdi G. Habashi, and Marco Fossati. Real-time regional jet comprehensive aeroicing analysis via reduced order modeling. *AIAA Journal*, 548(12):3787–3802, 2016.