32nd Congress of the International Council of the Aeronautical Sciences
September 6-10, 2021
Pudong Shangri-La, Shanghai, China

APPLICATION OF ARTIFICIAL NEURAL NETWORKS FOR EFFICIENT RELIABILITY-BASED DESIGN OPTIMIZATION

Xin Chen, Atif Riaz, Soufiane El Fassi

Cranfield University, Cranfield, MK43 0AL, United Kingdom

Abstract

This paper presents a novel approach for computing the probability of constraint satisfaction in the context of reliability-based design optimization of complex systems, such as an aircraft. The constraint satisfaction is formulated as a classification problem, where an artificial neural network is employed to predict the corresponding probabilities. Different from existing approaches which require a double loop paradigm for either training or applying the artificial neural networks, the proposed approach is able to give probability prediction for any arbitrary point in the entire design space, using only one loop of model evaluations. As a result, the computational cost is reduced. An illustrative example is used to highlight the differences between the existing and proposed approaches. The usefulness of the latter is further demonstrated with an aircraft sizing case study. The results show that the proposed approach can provide reasonably accurate predictions for a realistic design problem.

Keywords: Aircraft conceptual design, Uncertainty Quantification, Artificial Neural Networks, Machine Learning, Reliability-based Design Optimization

1. Introduction

Designing complex systems, such as an aircraft, involves a large number of computational models and design variables, which may be characterized by uncertainty. Such uncertainty could be, for instance, due to limited model fidelity, numerical errors, or unfixed parameters. If not handled properly, these sources of uncertainty will cause inaccurate computational results, which in turn could lead to inaccurate performance assessment of the aircraft and risk of unnecessary design iterations.

During the last few decades, methods have been developed for aircraft design under uncertainty using the probabilistic approach [1–4], where uncertain parameters are represented as probability distributions. In general, two design paradigms are widely adopted: Robust Design Optimization (RDO) [5–9] and Reliability Based Design Optimization (RBDO) [10,11]. The former intends to reduce performance variation while the latter aims to maintain required probabilities of constraints satisfaction. In both cases, a double-loop approach is normally utilized to obtained desired design solutions. The outer (optimization) loop iterates the values of design variables to search the design space, while the inner (uncertainty propagation [12–14]) loop calculates certain quantities of interest for each design point (e.g., output variances in RDO and probabilities in RBDO). In practice, this inner loop could be computationally very expensive, as it normally requires a large amount of repeated model evaluations, for example, to perturb the uncertain parameters according to predefined probability distributions. This is currently one of the major challenges in aircraft design under uncertainty.

In this research, we proposed a novel approach to compute the probabilities of constraint satisfaction for RBDO, using Artificial Neural Networks (ANN). Although ANN have already been applied in the context of RBDO, the proposed approach is different from the existing ones as it requires only a single loop, therefore improving the computational efficiency.

The remaining part of this paper is structured as follows: Section 2 reviews the basics of artificial neural networks and their current applications in RBDO. The proposed approach is presented in

Section 3. The major differences between the proposed and existing approaches in problem formulation are highlighted with an illustrative example in Section 4, where the proposed method is also evaluated with an aircraft design use-case to test its scalability and accuracy. Finally, the summary, conclusions, and future work are outlined in Section 5.

2. Literature review

2.1 Artificial Neural Networks (ANN)

An artificial neural network is a type of computing system which imitates the signal processing behaviors in a biological brain. It has been widely applied in different machine learning problems, such as data mining, Computer Vision (CV), Natural Language Processing (NLP), and so forth. In general, these applications can be categorized into two groups: regression and classification. In a regression problem, the ANN is employed in a similar way as a surrogate model to approximate and replace a certain mathematical function (e.g., a Computational Fluid Dynamics (CFD) model), which can be computationally expensive to execute. In a classification problem, the ANN is utilized to distinguish between different sets of data, according to their characteristics (e.g., to classify objects in an image).

The first mathematical model of ANN was proposed in 1943 [15], followed by many variants in different fields [16,17]. In this paper, we will be focusing on the perceptron model, which was initially developed for image recognition [18]. Later, it has been proven that the Multi-layer Perceptrons (MLP) are universal function approximators [19,20], which makes it one of the most widely used techniques in machine learning. In this section, the mathematical formulation of ANN is briefly introduced, while more elaborated descriptions can be found in [21,22].

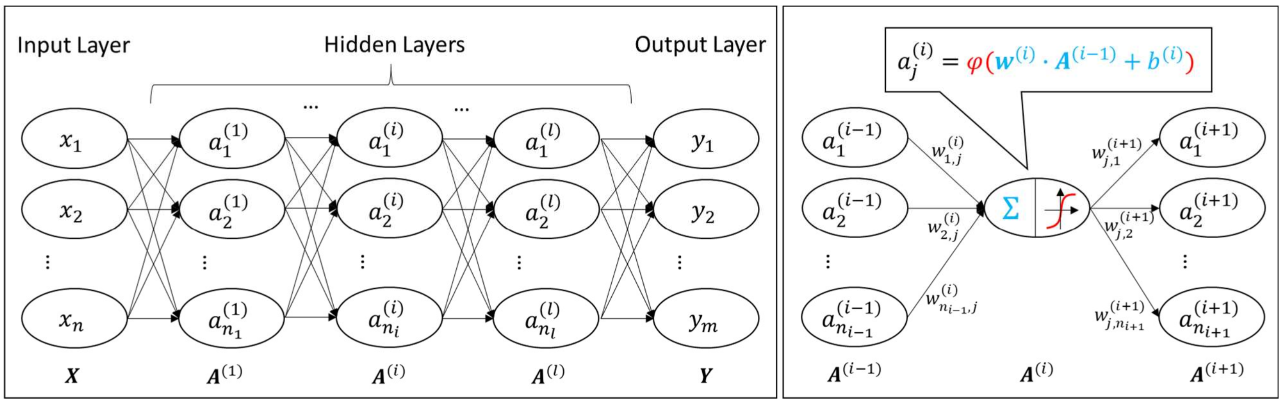

Figure 1 – (a) illustration of an artificial neural network. (b) Randomization of computational models

A typical MLP ANN is shown in Figure 1 (a), where the neurons are represented as ellipses. Each neuron is corresponding to a variable (also referred to as the activation), while a vertical stack of neurons is referred to as a layer. On the far left is the (global) input layer, which can be represented as a vector of n inputs variables, $\boldsymbol{X} = [x_1, x_2, \dots, x_n]^T$. The output layer is on the far right, which consists of m output variables as a vector, $\boldsymbol{Y} = [y_1, y_2, \dots, y_m]^T$. There could be one or more hidden layers between the input and output ones. The i_{th} hidden layer is represented as a vector, $\boldsymbol{A}^{(i)} = \left[a_1^{(i)}, a_2^{(i)}, \dots, a_{n_i}^{(i)}\right]^T$, where n_i is the number of neurons in that layer. These hidden layers enable the representation of complex mathematical features that cannot be directly captured by $\boldsymbol{X}$. The number of layers, l, is also referred to as the "depth" of the ANN, which characterizes the so-called "deep learning" process.

Figure 1 (b) illustrates the details of neuron connections, where each neuron takes all the activations in the previous layer as inputs and its output is fed to all the neurons in the next layer. For instance, the variable corresponding to the j_{th} neuron in the i_{th} hidden layer, $a_j^{(i)}$ is defined as:

$$a_j^{(i)} = \varphi(\boldsymbol{w}_j^{(i)} \cdot \boldsymbol{A}^{(i-1)} + b_j^{(i)}) \tag{1}$$

The part inside the bracket is a linear combination of the activations in the previous layer, $\boldsymbol{A}^{(i-1)} =$

$\left[a_1^{(i-1)}, a_2^{(i-1)}, \dots, a_{n_{i-1}}^{(i-1)}\right]^T$, while $\boldsymbol{w}_j^{(i)} = \left[w_{1,j}^{(i)}, w_{2,j}^{(i)}, \dots w_{n_{i-1},j}^{(i)}\right]^T$ is a vector of weight factors. As indicated in the Figure 1 (b), $w_{k,j}^{(i)}$ is corresponding to the connection between the k_{th} neuron in the $(i-1)_{th}$ layer and the j_{th} neuron in the i_{th} layer. There is also a bias term indicated by $b_j^{(i)}$. In equation (1), φ is the so-called activation function, which simulate a signal pulse in a biological neuron. There are different types of activation functions, which include: linear, Rectified Linear Unit (ReLU), logistic, hyperbolic tangent, etc. The reader is referred to [21,22] for more details. Note that equation (1) is for $a_j^{(i)}$ only. Therefore, to compute the entire vector $\boldsymbol{A}^{(i)} = \left[a_1^{(i)}, a_2^{(i)}, \dots, a_{n_i}^{(i)}\right]^T$. The equation becomes:

$$\boldsymbol{A}^{(i)} = \varphi(\boldsymbol{W}^{(i)} \cdot \boldsymbol{A}^{(i-1)} + \boldsymbol{b}^{(i)}) \tag{2}$$

where $\boldsymbol{W}^{(i)}$ is an $n_i \times n_{i-1}$ matrix of all the weight factors and $\boldsymbol{b}^{(i)}$ is an $n_i \times 1$ vector of all the biases. The activation function is applied elementwise for the term inside the bracket (which is an $n_i \times 1$ vector).

Given the input vector $\boldsymbol{X}$, the neuron activations $\boldsymbol{A}^{(i)}$ will be computed layer by layer with equation (2), and finally the output $\boldsymbol{Y}$ will be obtained. In a regression problem, the outputs are estimators of the original function; while in a classification problem, the outputs are posterior probabilities of an input dataset belonging to a category.

Before applying the ANN, a training set consisting of N pairs of observed $(\boldsymbol{X}_{Obs}, \boldsymbol{Y}_{Obs})$ needs to be produced. For a regression problem, this is achieved by repeated execution of the original function using a design of experiment. In the case of classification, the training set will be generated using samples with pre-defined labels. The training process of an ANN is essentially an optimization process, which updates all the weight factors to minimise the gap between the predicted output values and the observed ones. Because the problem dimension is massive (consider all the $\boldsymbol{W}^{(i)}$ matrices), an efficient algorithm called back-propagation [23–27] is normally applied. As the correct predictions are known *a priori*, this way of training is also referred to as "supervised learning".

2.2 Problem Formulation of Reliability-base Design Optimization (RBDO)

A RBDO problem could be formulated as:

$$\begin{aligned} &\text{Find: } \boldsymbol{d} \in R^n \\ &\text{To minimize: } f(\boldsymbol{d}, \boldsymbol{u}) \\ &\text{Subject to: } P\{g_i(\boldsymbol{d}, \boldsymbol{u}) \le 0\} \ge p_i, i = 1,2, \dots, N \end{aligned} \tag{3}$$

where f is the objective function and g_i is the i_{th} constraint. In practice, f and g_i are based on certain analysis models which will be executed in each design iteration. $P\{g_i(\boldsymbol{d}, \boldsymbol{u}) \le 0\}$ is the probability of constraint satisfaction and p_i is a pre-specified lower bound for that probability.

The inputs of all the models can be decomposed into two part: a vector of design variables, $\boldsymbol{d} = [d_1, d_2, \dots, d_n]$, and a vector of uncertain variables, $\boldsymbol{u} = [u_1, u_2, \dots u_m\ ]$. The former (at least the nominal value) is control by a designer or an optimizer, whilst the latter follows a joint probability distribution, $\Pi(u_1, u_2, \dots u_m)$.

Without loss of generality, the design variables and analysis models could also be influenced by uncertainty (e.g., due to manufacturing inaccuracy and model discrepancy, respectively). As illustrated in Figure 2, these types of uncertainty are captured by multiplying the original deterministic value of the design variable or model output with a corresponding random factor, and the latter should also be included as an element in the vector of uncertain variables, $\boldsymbol{u}$.

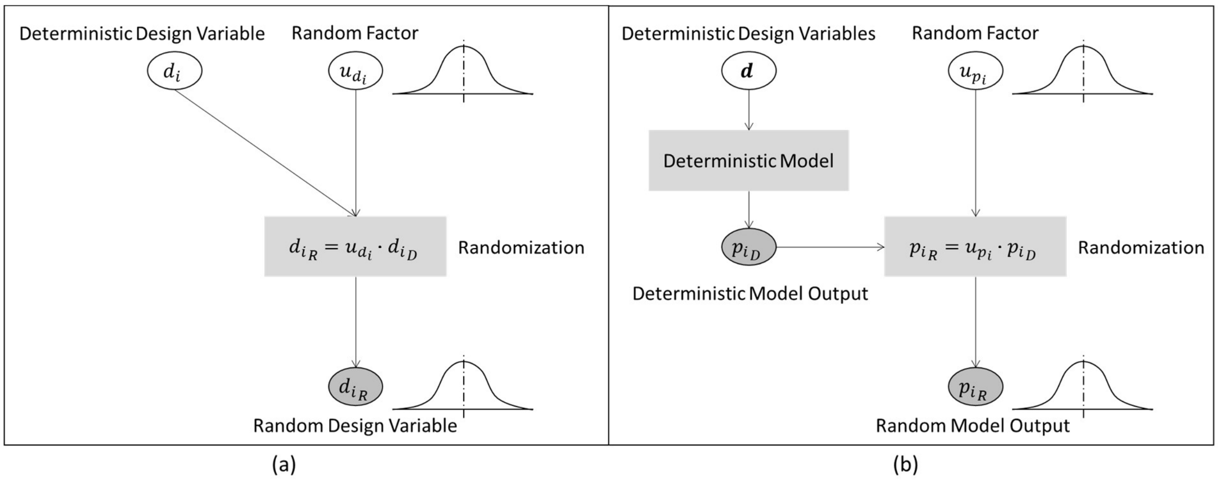

Figure 2 – (a) representation of an uncertain design variable. (b) representation of model uncertainty (adapted from [28])

Due to the existence of $\boldsymbol{u}$, both f and g_i are uncertain variables as well, which follow their corresponding probability distributions. As illustrated in Figure 3, if the Probability Density Function (PDF) of g_i is known (noted as $\pi(g_i)$), then $P\{g_i(\boldsymbol{d},\boldsymbol{u}) \leq 0\}$ could be obtained analytically by:

$$P\{g_i(\boldsymbol{d},\boldsymbol{u}) \leq 0\} = \int_{-\infty}^{0} \pi(g_i)\, dg_i \tag{4}$$

In practice, $\pi(g_i)$ is normally unknown, therefore requires some numerical solutions such as Monte Carlo Simulation (MCS) or other uncertainty propagation techniques [12–14].

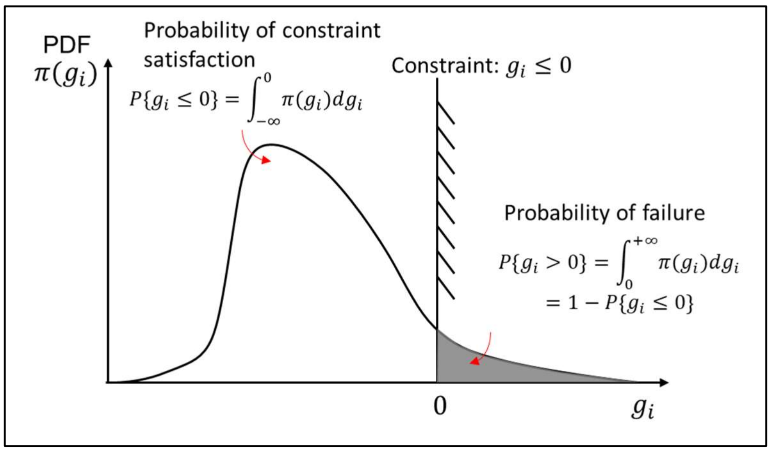
PDF
$\pi(g_i)$
Probability of constraint satisfaction
$P\{g_i \leq 0\} = \int_{-\infty}^{0} \pi(g_i) dg_i$
Constraint: $g_i \leq 0$
Probability of failure
$P\{g_i > 0\} = \int_{0}^{+\infty} \pi(g_i) dg_i$
$= 1 - P\{g_i \leq 0\}$
0
g_i

Figure 3 – Probability of constraint satisfaction for g_i

2.3 Application of ANN in RBDO

Early applications of ANN in reliability analysis can be found in [29,30], where the ANNs are used as surrogates to replace the original models in a MCS for computing the probability of failure. In [31], ANN is applied in the context of reliability-based design optimization, using two different approaches. In the first approach, an ANN is first trained using a double loop paradigm, where the outer loop is applied to sample the design space; while for each design point, apart from examining the deterministic constraints, an MCS is performed as the inner loop to check if the probability of constraint satisfaction is achieved. This ANN is then applied in an actual (single loop) optimization process to verify both the deterministic and probabilistic constraints. Although not mentioned explicitly, it can be inferred that this ANN is implemented as a classifier, as it has only one output prediction for all the constraints. In the second approach, the actual optimization is conducted with the double loop paradigm. The outer loop searches the design space to minimise the objective function. For each design point, a distinct ANN is trained by sampling only the uncertain variables (while fixing the design variables). This ANN is then utilized as an approximator to replace the original model for MCS in the inner loop (same as [30]). In [32], the same approach is extended to Reliability-based Robust Design Optimization (RRDO), where the ANN approximator is used in the

MCS to compute the standard deviations as well. In [33,34], subset simulation and Markov Chain Monte Carlo Simulation (MCMCS) are employed for more efficient sampling, whereas ANN is still used as a surrogate model.

Comparative studies were conducted in [35], to assess different types of ANN (MLP vs radial basis functions), formulation of cost function, training algorithms, and sampling strategies. In [35], the authors pointed out explicitly that the ANN could be used as a classifier. Instead of approximating the original function, the ANN predicts if an output is below or above the threshold value to meet a constraint. As mentioned above, the first approach in [31] might also have employed the ANN as a classifier. However, the difference is that: in [31], the ANN classifier is trained after assessing the probabilistic constraint using the original model in MCS, while in [35], the ANN classifier is trained first, then used in the MCS, where the probability of failure is computed by counting the number of 0's and 1's in the ANN predictions. Comparison studies show that the classification approach requires a much smaller training set compared with regression approach, as the former gives binary predictions rather than reproducing the entire function. As a result, more samples should be allocated on the areas close to the constraint, which is achieved in [36] by using Particle Swarm Optimization (PSO), maximum entropy, and Sobol' sequences. The classification approach was extended to random field and random process problems in [37,38]. Other implementations of the classification approach include [39,40] using Probabilistic Neural Networks (PNN) and [36,41,42] using Support Vectors Machines (SVM).

ANN is utilized in [43,44] to solve an inversed reliability problem, where design variables and/or distribution parameters (means and standard deviations) need to be identified, given a pre-specified probability of failure (or reliability index). A similar double loop paradigm is employed to produce the training set, where the outer loop produces random samples of the design variables and/or distribution parameters using Latin Hyper Cube (LHC), while for each sample, an inner loop is applied to compute the corresponding probability of failure using the First Order Reliability Method (FORM). These samples are then used for training the ANN, whereas the probability of failure is considered as an input whilst the design variables and/or distribution parameters are considered as outputs. This ANN is then utilized to map any desired probabilities of failure to proper values of the design variables and/or distribution parameters.

The existing approaches are categorized into four groups (A1 to A4) and summarized in Table 1. It can be seen that all the approaches utilize the double loop paradigm for either training or application of the ANN. In both cases, the computational cost (number of searching iterations multiplied by number of model evaluations for uncertainty propagation) could be very large, even when considering the reduction by applying the ANN and advanced sampling strategies.

Furthermore, as mentioned in Section 2.1, when the ANN is implemented as a classifier, the outputs themselves can be interpreted as posterior probabilities. This feature has not been exploited in the published literature, which motivates our proposed method that is presented in the next section.

Table 1 – Summary of the general approach

	• Outer Loop: Search Design Space • Inner Loop: ○ Train ANN ○ Reliability analysis using ANN	• Outer Loop: Search Design Space • Inner Loop: ○ Reliability analysis using original model • Train ANN
Regression	A1: [29,30,32–35], Second approach in [31]	A3: [43,44]
Classification	A2: [35–42]	A4: First approach in [31]

3. Methodology

The proposed approach uses the classification approach. However, different from the existing approaches which employ a double loop for training or application, it solves the problem with a single loop only, where the probability of constraint satisfaction is obtained as a direct output of the ANN.

The steps of the proposed approach are illustrated in Figure 3, using the notations defined in Section 2.2. The architecture of the ANN needs to be decided first. The number of input neurons is equal to that of the design variables, while the number of output neurons is equal to that of the constraints. Each output neuron provides the probability of satisfying the corresponding constraint. It should be noted that the uncertain variables are not considered as inputs for this ANN. There is currently no theoretical guidance for the number of hidden layers and the numbers of neurons in each hidden layer. According to the literature, one or two hidden layers are usually sufficient for problems of comparable scale, while the number of neurons in the hidden layer can be estimated as two times the number of ANN inputs. In this research, we use the logistic function as the activation function [22].

To prepare the training set, the design space is first sampled M times. This could be done with LHC, Sobol' sequences, pure random, or full factorial design of experiment. The samples are represented as a $M \times n$ matrix:

$$\boldsymbol{D} = \begin{bmatrix} d_1^{(1)} & d_2^{(1)} & \dots & d_n^{(1)} \\ d_1^{(2)} & d_2^{(2)} & \dots & d_n^{(2)} \\ \dots & \dots & \dots & \dots \\ d_1^{(M)} & d_2^{(M)} & \dots & d_n^{(M)} \end{bmatrix} \tag{5}$$

where $d_i^{(j)}$ is the j_{th} sample of the i_{th} design variable and each row is a design point in the multi-dimensional design space.

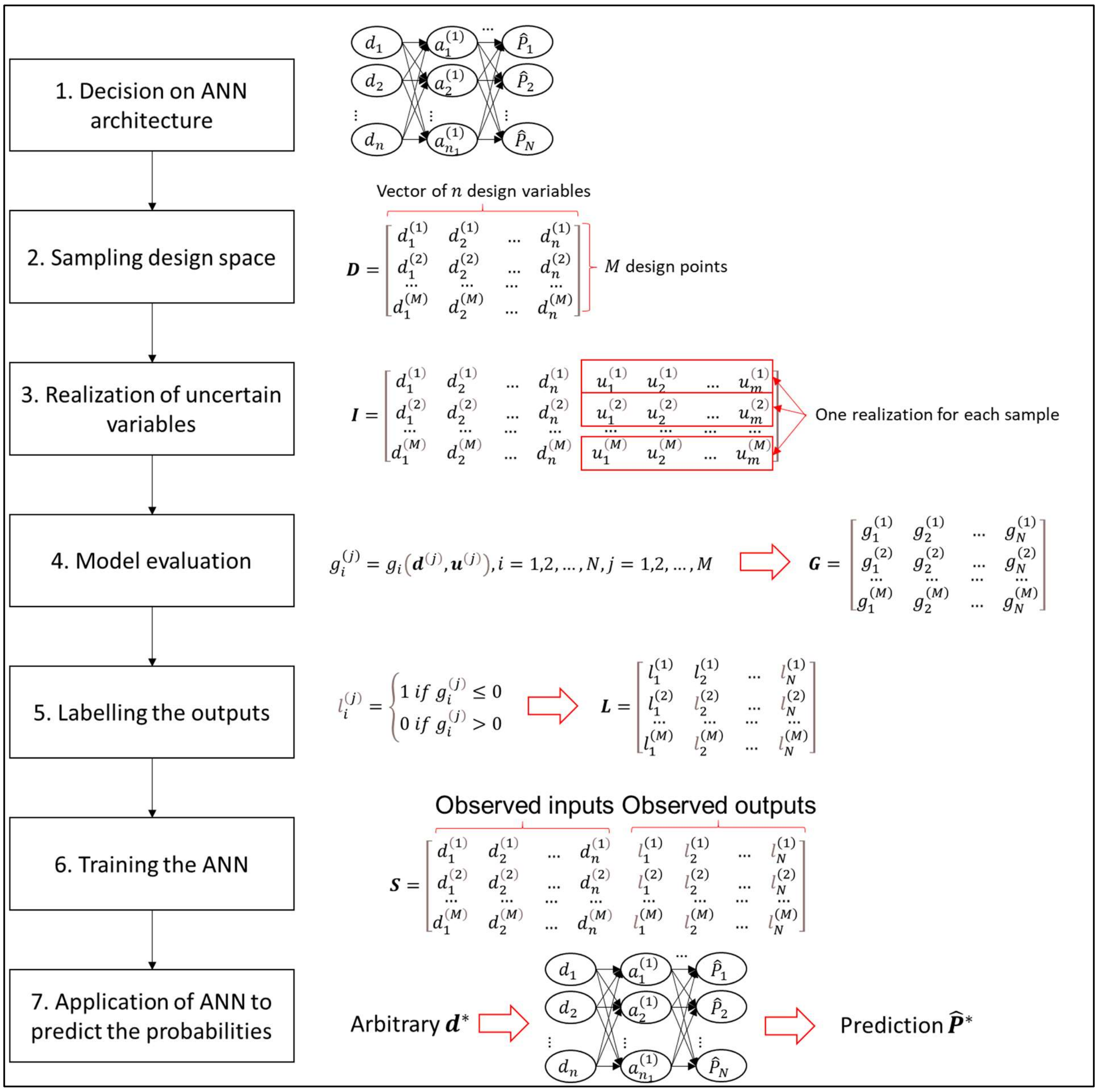
1. Decision on ANN architecture
2. Sampling design space
3. Realization of uncertain variables
4. Model evaluation
5. Labelling the outputs
6. Training the ANN
7. Application of ANN to predict the probabilities
Vector of n design variables
M design points
One realization for each sample
$g_i^{(j)} = g_i(\boldsymbol{d}^{(j)}, \boldsymbol{u}^{(j)}), i = 1,2,\dots,N, j = 1,2,\dots,M$
$l_i^{(j)} = \begin{cases} 1 \text{ if } g_i^{(j)} \le 0 \\ 0 \text{ if } g_i^{(j)} > 0 \end{cases}$
Observed inputs
Observed outputs
Arbitrary $\boldsymbol{d}^*$
Prediction $\widehat{\boldsymbol{P}}^*$

Figure 4 – Steps of the proposed method

For each design point, only one realization of the uncertain variables is drawn from the original joint PDF, $\Pi(u_1, u_2, \dots u_m)$, using either pure or quasi-random samples. Similarly, the result is noted as a matrix:

$$\boldsymbol{U} = \begin{bmatrix} u_1^{(1)} & u_2^{(1)} & \dots & u_m^{(1)} \\ u_1^{(2)} & u_2^{(2)} & \dots & u_m^{(2)} \\ \dots & \dots & \dots & \dots \\ u_1^{(M)} & u_2^{(M)} & \dots & u_m^{(M)} \end{bmatrix} \quad (6)$$

By combining the design and uncertain variables, an $M \times (n+m)$ matrix is obtained, where each row is a complete input vector:

$$\boldsymbol{I} = \begin{bmatrix} d_1^{(1)} & d_2^{(1)} & \dots & d_n^{(1)} & u_1^{(1)} & u_2^{(1)} & \dots & u_m^{(1)} \\ d_1^{(2)} & d_2^{(2)} & \dots & d_n^{(2)} & u_1^{(2)} & u_2^{(2)} & \dots & u_m^{(2)} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ d_1^{(M)} & d_2^{(M)} & \dots & d_n^{(M)} & u_1^{(M)} & u_2^{(M)} & \dots & u_m^{(M)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{d}^{(1)}, \boldsymbol{u}^{(1)} \\ \boldsymbol{d}^{(2)}, \boldsymbol{u}^{(2)} \\ \dots \\ \boldsymbol{d}^{(M)}, \boldsymbol{u}^{(M)} \end{bmatrix} \quad (7)$$

For each row, the analysis models are evaluated.

$$g_i^{(j)} = g_i\left(\boldsymbol{d}^{(j)}, \boldsymbol{u}^{(j)}\right), i = 1,2, \dots, N, j = 1,2, \dots, M \quad (8)$$

Note that the models will be executed M times and there are N different constraints in total. Therefore, the output is an $M \times N$ matrix:

$$\boldsymbol{G} = \begin{bmatrix} g_1^{(1)} & g_2^{(1)} & \dots & g_N^{(1)} \\ g_1^{(2)} & g_2^{(2)} & \dots & g_N^{(2)} \\ \dots & \dots & \dots & \dots \\ g_1^{(M)} & g_2^{(M)} & \dots & g_N^{(M)} \end{bmatrix} \quad (9)$$

By verifying each constraint, a label matrix could be produced, where 1 indicates constraint satisfaction, while 0 means failure to satisfy the constraint:

$$\boldsymbol{L} = \begin{bmatrix} l_1^{(1)} & l_2^{(1)} & \dots & l_N^{(1)} \\ l_1^{(2)} & l_2^{(2)} & \dots & l_N^{(2)} \\ \dots & \dots & \dots & \dots \\ l_1^{(M)} & l_2^{(M)} & \dots & l_N^{(M)} \end{bmatrix} \quad (10)$$

$$l_i^{(j)} = \begin{cases} 1 \; if \; g_i^{(j)} \le 0 \\ 0 \; if \; g_i^{(j)} > 0 \end{cases} \quad (11)$$

The final training set contains all the design variables and labels, which are considered as the observed inputs and outputs, respectively. However, the uncertain variables are not included in the training set. As mentioned earlier, the uncertain variables are not used as the inputs of this ANN, instead, they are regarded as noises in the sampling process.

$$\boldsymbol{S} = \begin{bmatrix} d_1^{(1)} & d_2^{(1)} & \dots & d_n^{(1)} & l_1^{(1)} & l_2^{(1)} & \dots & l_N^{(1)} \\ d_1^{(2)} & d_2^{(2)} & \dots & d_n^{(2)} & l_1^{(2)} & l_2^{(2)} & \dots & l_N^{(2)} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ d_1^{(M)} & d_2^{(M)} & \dots & d_n^{(M)} & l_1^{(M)} & l_2^{(M)} & \dots & l_N^{(M)} \end{bmatrix}$$

In this research, cross-entropy [21,22] is employed as the cost function for training, while the backpropagation algorithm is used to update the weight factors (as defined in Section 2.1). After training, the ANN could be used to predict the probability of satisfying each constraint given any arbitrary vector of design variables.

$$\widehat{\boldsymbol{P}}^* = ANN(\boldsymbol{d}^*)$$

where $\boldsymbol{d}^* = [d_1^*, d_2^*, \dots, d_n^*]$ is an arbitrary point of interest in the design space, while $\widehat{\boldsymbol{P}}^* = [\hat{P}_1, \hat{P}_2, \dots, \hat{P}_N]$. Note that there is no need to produce any realization of the uncertain variables or run MCS on this ANN.

4. Evaluation

4.1 Illustrative Example

In this section, an illustrative example is utilized to highlight the differences in mathematical formulations between the proposed and existing approaches. Consider a problem with two design variables, $\boldsymbol{d} = [d_1, d_2]$, and two uncertain variables, $\boldsymbol{u} = [u_1, u_2]$. The design variables are bounded in the intervals $[0,1]$ and both uncertain variables follow independent Gaussian distributions $\mathcal{N}(0, 0.5)$, with means, $\mu_{u_1}, \mu_{u_2} = 0$, and standard deviations, $\sigma_{u_1}, \sigma_{u_2} = 0.5$. The constraint function is defined as:

$$g(d_1, d_2, u_1, u_2) = (u_1 + 1) \cdot (d_1 + d_2) + u_2 - 1 \tag{12}$$

If both u_1 and u_2 are fixed at their mean values ($u_1, u_2 \equiv 0$), the feasible design space is illustrated in Figure 5 (a). However, because the values of u_1 and u_2 are varying randomly, every point in the designs space has a probability of failure, even if it is located inside the feasible region. For instance, given a design point DP_1 at (0.25, 0.25), to satisfy the constraint ($g \leq 0$), the uncertain variables need to fulfill the following relationship:

$$g \leq 0 \Leftrightarrow (u_1 + 1) \cdot (0.25 + 0.25) + u_2 - 1 \leq 0 \Leftrightarrow 0.5u_1 + u_2 - 0.5 \leq 0 \tag{13}$$

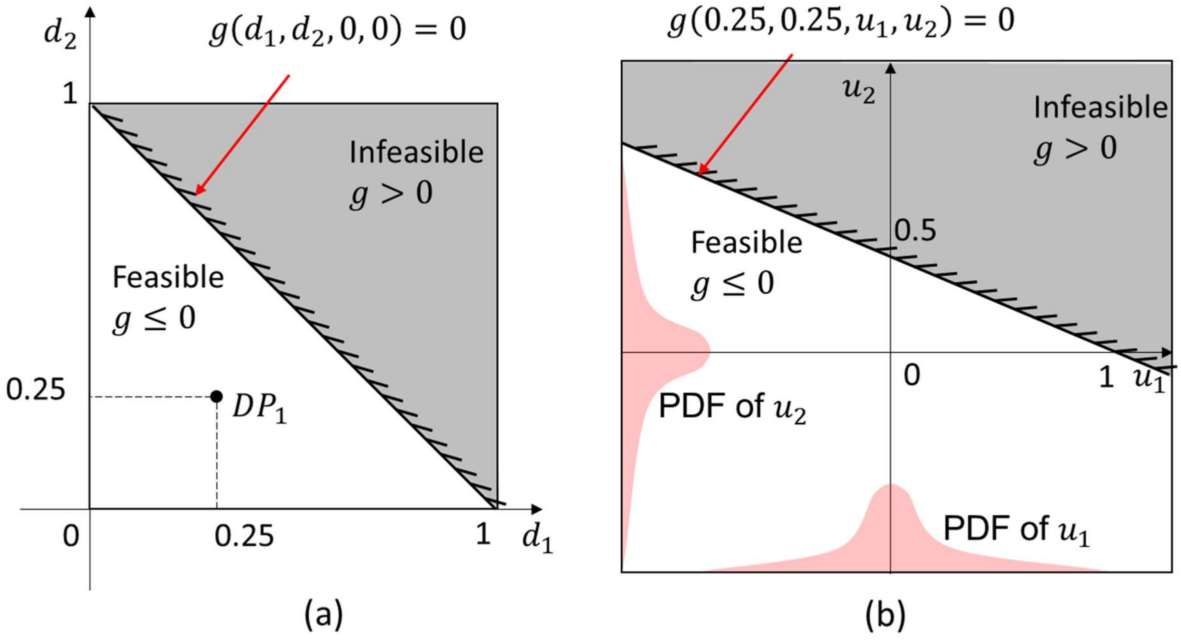

Figure 5 – Visualization of the design space (a) and uncertain variable space (b)

As illustrated in Figure 5 (b), the probability of constraint satisfaction corresponding to the highlighted design point DP_1 can be computed by:

$$P\{g \leq 0\} = P\{0.5u_1 + u_2 \leq 0.5\} = \int_{-\infty}^{+\infty} \int_{-\infty}^{0.5-0.5u_1} \pi_{u_1}(u_1)\pi_{u_2}(u_2) du_1 du_2 \tag{14}$$

In this case, $0.5u_1 + u_2$ is a linear combination of Gaussian distributions. Therefore, equation (14) can be solved by considering $u = 0.5u_1 + u_2$ as a new uncertain variable:

$$P\{u \leq 0.5\} = P\left\{u^* \leq \frac{0.5}{\sqrt{0.5^2 * \sigma_{u_1}^2 + \sigma_{u_2}^2}}\right\} = \Phi(0.8944) = 0.8145 \tag{15}$$

where u^* follows a standard Gaussian distribution, and Φ is the corresponding Cumulative Density Function (CDF), for which the value can be obtained from a lookup table.

The steps to solve the illustrative problem with the proposed and existing approaches (A1 to A4) are summarized in Table 2.

In A1 and A2, the inputs of the ANN are the uncertain variables $\boldsymbol{u} = [u_1, u_2]$. A Monte Carlo simulation is applied on this ANN by varying u_1, u_2 with fixed d_1, d_2. Therefore, each design point requires a distinct ANN and MCS.

In A3, A4, and the proposed approach, the inputs of the ANN are the design variables $\boldsymbol{d} = [d_1, d_2]$. Therefore, only one ANN is needed. However, A3 and A4 require to conduct reliability analysis for each design point using the original model, that is, the training process requires a double loop of model execution. Furthermore, this (inner loop) reliability analysis is not needed in the proposed approach.

Another important difference between the existing approaches and the proposed one is that: for the former, the safe and failure points are "perfectly separable", and the boundary is "noise-free" [36], while in the proposed approach, the boundary is "noisy", because the uncertain variables are realized only once for each design point. This "blurred" boundary is intended, so that the reliability problem becomes similar to an image recognition one, where the uncertain variables are considered as noises and the outputs of the ANN classifier converge to posterior probabilities as more training points are used.

For instance, the training sets used in the proposed approach and A2 are illustrated in Figure 6 (a) and (b), respectively. In the former, the points are sampled in the design space $[d_1, d_2]$, while in the latter, the samples are in the uncertain variable space $[u_1, u_2]$, associated to a specific point in the design space (e.g., $DP1$ in Figure 5). It can be noted that, in the proposed approach, the feasible and infeasible samples are mixed together, while there is a clear boundary in the case of A2.

The resulting ANN classifier is shown in Figure 6 (c) and (d), respectively. The ANN obtained by the proposed approach is smoother, and its output is directly the probability of constraint satisfaction for any arbitrary design point (e.g., $\hat{P}\{g \leq 0\} \approx 0.7458$ for $DP1$). The ANN obtained by A2 is steeper and can be used to assess the probability of constraint satisfaction for $DP1$ only. To do that, one additional MCS is applied on this ANN and the probability could be estimated by the number of predictions larger than 0.5 divided by the total number of MCS samples. Note that the 0.5 threshold is used in [35], and that other rules exist depending on the type and implementation of the ANN.

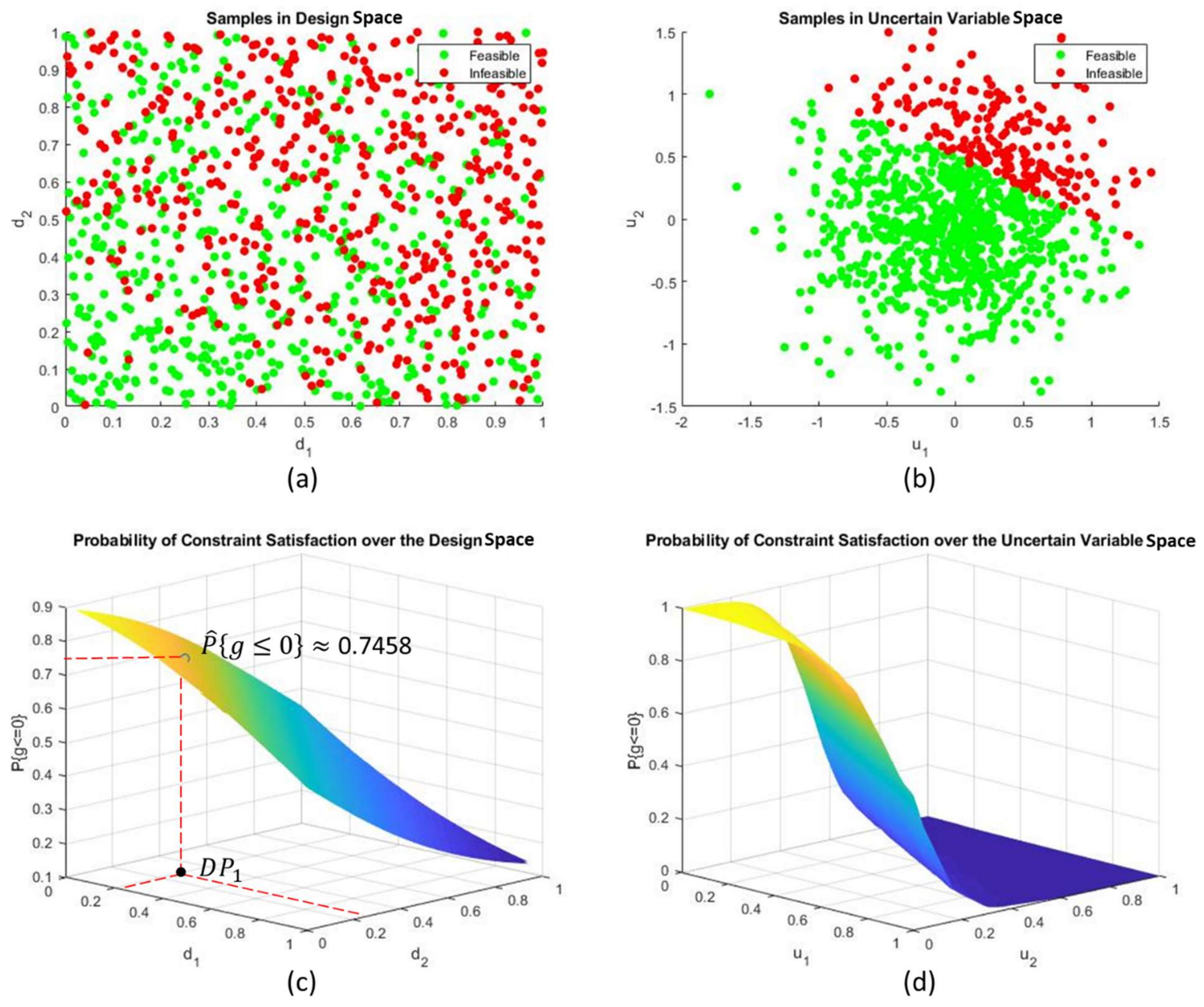

Figure 6 – Comparison of the training set and resulting ANN classifier.

Table 2 – Comparison of the proposed and existing approaches

Steps	Proposed Approach	A1	A2	A3	A4
Sample the design space	Produce M pairs of $[d_1, d_2]$	Produce M_{Out_x} pairs of $[d_1, d_2]$			
Sample the uncertain variable space for each design point	Produce one realization of $[u_1, u_2]$ for each pair of $[d_1, d_2]$	Produce M_{In_x} realizations of $[u_1, u_2]$ for each pair of $[d_1, d_2]$			
ANN training for each design point	N/A	Train the ANN using the M_{In_x} samples of $[u_1, u_2]$		N/A	
		ANN predicts an approximation $\hat{g}$.	ANN predicts a value $\hat{l}$ between 1 or 0 to indicate if $g \leq 0$.		
Applying reliability analysis for each design point	N/A	Produce additional M_{MCS_x} pairs of $[u_1, u_2]$ for MCS using the ANN.		Conduct reliability analysis using the original model with the M_{In_x} samples of $[u_1, u_2]$	
		$\hat{P}\{g \leq 0\} = \dfrac{Number\ of\ (\hat{g} \leq 0)}{M_{MCS_x}}$	$\hat{P}\{g \leq 0\} = \dfrac{Number\ of\ (\hat{l} \geq 0.5)}{M_{MCS_x}}$		
ANN training for entire design space	Train the ANN using the M samples of $[d_1, d_2]$. ANN predicts $\hat{P}\{g \leq 0\}$, given a pair of $[d_1, d_2]$	N/A		Train the ANN using the M_{Out_x} samples of $[d_1, d_2]$	
				ANN predicts a value $\hat{l}$ between 1 or 0 to indicate if $\hat{P}\{g \leq 0\} \geq p_{Req}$, given a pair of $[d_1, d_2]$	ANN predicts $\hat{P}\{g \leq 0\}$, given a pair of $[d_1, d_2]$. (In [43,44] an inverse problem is solved, where ANN predicts d_1 or d_2, given a predefined probability p_{Req})
Total Cost	M	$M_{Out_x} \cdot M_{In_x}$			

The computational cost is measured by the number of evaluating the original models. In Table 2, M is the number of model evaluations for the proposed method. For this specific problem, the predicted probability against the computational cost is shown in Figure 7. It can be seen that the error is reduced to 6% with 100 samples and 3% with 1000 samples. For this illustrative example, not much effort has been devoted to refining the ANN architecture and training process, therefore, there is still a large room for improving the efficiency.

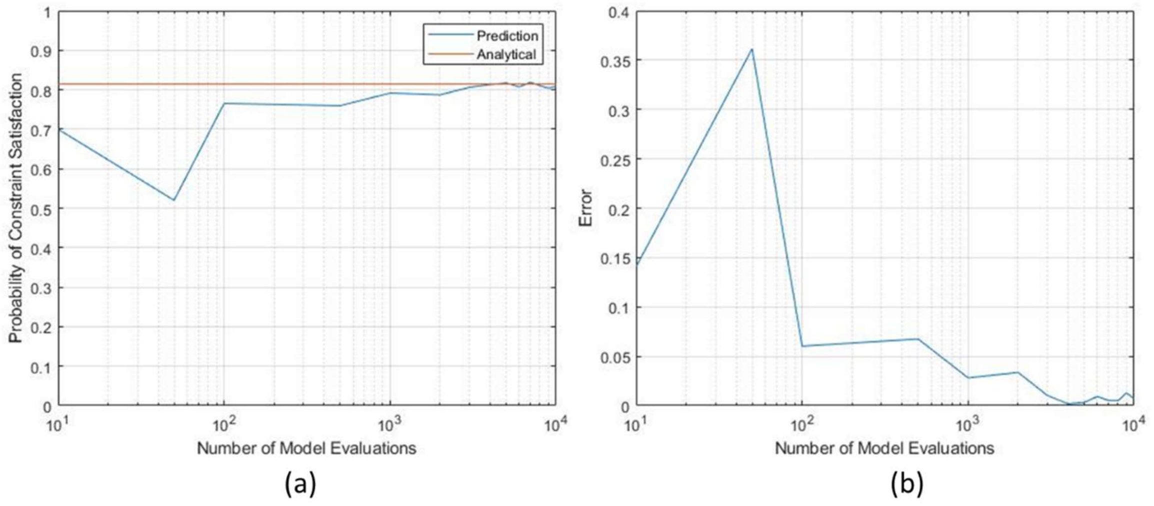

Figure 7 – Convergence of the prediction and error against computational cost

For the existing approaches, M_{Out_x} and M_{In_x} are the numbers of model evaluations in the outer and inner loops, for the x_{th} approach, respectively. In A1 and A2, there are M_{MSC_x} additional samples for the MCS in the inner loop. However, these samples are evaluated using the ANN as the surrogate which is much faster than the original model in a real application. Therefore, these samples are ignored in measuring the computational cost.

Specific values of these M_{Out_x} and M_{In_x} need further investigation. Here a quantitative assessment is applied. For A1 and A2, M_{Out_1} and M_{Out_2} are the numbers of iterations in the optimization process, therefore they are expected to be much lower than M, especially if a gradient based solver is used. M_{In_1} and M_{In_2} are the number of points used to train the ANN in each design iteration. For this simple problem, M_{In_1} and M_{In_2} are expected to be lower than M, where M_{In_2} is expected to be lower than M_{In_1}.

A3 and A4 require to explore the design space, therefore M_{Out_3} and M_{Out_4} are expected to be higher than M_{Out_1} and M_{Out_2}, but could be lower than M, because the samples in A3 and A4 are less noisy. M_{In_3} and M_{In_4} could be higher than M_{In_1} and M_{In_2}, because reliability analysis normally requires more points than training the ANN, otherwise there is no need to use the ANN as a surrogate. However, in the specific implementation of [43,44], where FORM is used, M_{In_4} should be lower than M_{In_3}. If MCS is used, they are expected to be the same.

In summary a rough estimated relationship is as below:

$$M_{Out_1}, M_{Out_2} \ll M_{In_1}, M_{In_2}, M_{Out_3}, M_{Out_4} < M_{In_3}, M_{In_4}, M \qquad (16)$$

With the current assumptions, the computational cost of the proposed approach M could be higher than that for any inner or outer loops. However, as the total cost of the existing approaches is computed by $M_{Ou\ x} \cdot M_{In_x}$, the proposed approach can still be more efficient in potential. Especially if the objective function is complex and requires a large number of $M_{Ou\ x}$ to explore the design space.

4.2 Realistic Test-case

A realistic but not real problem is utilized to test the scalability and accuracy of the proposed approach. The computational model is an aircraft sizing code called USMAC [45].

As summarized in Table 3, the problem has 6 design variables, 5 uncertain variables, and 4 constraints. Because the aim is to explore the entire design space, no objective function is defined.

Table 3 – Summarize of the realistic test-case.

Category	Variable Name	Symbol	Value/Distribution/Constraint
Design Variables	Sea Level Static Thrust $[N]$	$SLST$	$[110000, 130000]$
	Bypass Ratio	BPR	$[5, 6]$
	Wing Area $[m^2]$	S_W	$[110, 130]$
	Span $[m]$	b	$[35, 40]$
	Quarter Chord Sweep Angle $[degree]$	$\Lambda_{0.25}$	$[25, 30]$
	Thickness to Chord Ratio	t/c	$[0.08, 0.12]$
Uncertain Variables	Cruise Altitude $[ft]$	h_{CRZ}	Uniform: $\mathcal{U}(35000, 36000)$
	Cruise ISA deviation $[K]$	ΔT	Triangular: $\mathcal{Tri}(-5, 0, 5)$
	Cruise Specific Fuel Consumption Calibration Factor	k_{SFC}	Gaussian: $\mathcal{N}(1, 0.03)$
	Cruise Lift over Drag Calibration Factor	k_{LoD}	Gaussian: $\mathcal{N}(1, 0.03)$
	Empty Weight Calibration Factor	k_{W_E}	Gaussian: $\mathcal{N}(1, 0.03)$
Constraint	Maximum Take-off Weight $[kg]$	W_{TO}	$W_{TO} \leq 80000$
	Range $[nm]$	R	$R \geq 3300$
	Take-off Field Length $[m]$	$TOFL$	$TOFL \leq 2700$
	Approach Velocity $[kts]$	V_{APP}	$V_{APP} \leq 135$

The selected ANN architecture has 6 inputs, 4 outputs, and one hidden layer with 12 neurons. This ANN is trained with a gradually increasing number of samples (10, 50, 100, 500, 1000, 5000, 10000) in the (6-dimensional) design space.

After each stage of training, the accuracy of the ANN is measured by computing the Root Mean Square Error ($RMSE$) between the predicted and reference probabilities ($\hat{P}_{ij}$ and P^*_{ij}) for all the 4 constraints at 50 randomly selected points in the (6-dimensional) design space. The reference probabilities are obtained by conducting a MCS with 10000 samples in the (5-dimensional) uncertain variable space for each design points (50*10000 samples in total):

$$RMSE_i = \sqrt{\frac{1}{50}\sum_{j=1}^{50}(\hat{P}_{ij} - P^*_{ij})}, i = 1,2,3,4 \tag{17}$$

$$RMSE_{Total} = \sqrt{\frac{1}{50 \cdot 4}\sum_{j=1}^{50}\sum_{i=1}^{4}(\hat{P}_{ij} - P^*_{ij})} \tag{18}$$

The plot of $RMSE$ against the number of model evaluations (training samples) is shown in Figure 8. It can be seen that the average errors are reduced to 0.05 with roughly 500 points.

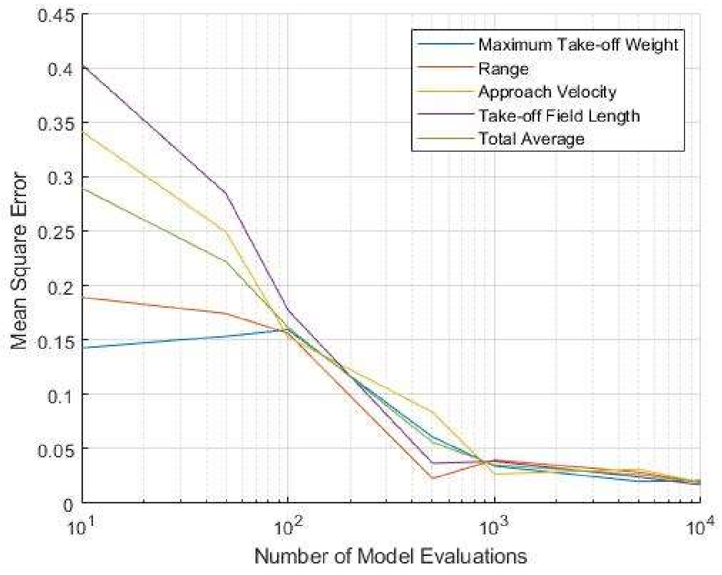

Figure 8 – Root mean square error of the ANN against number of model evaluations

For visualization purposes, a reduced-order problem is performed over a 2-dimensional design space of the span and wing area, while fixing the values of the other design variables. Figure 9 illustrates the validation points (in red) and the ANN predictions (as surfaces) for the 4 constraints. It can be seen that this ANN (based on 500 model evaluations) has a good agreement with validation points.

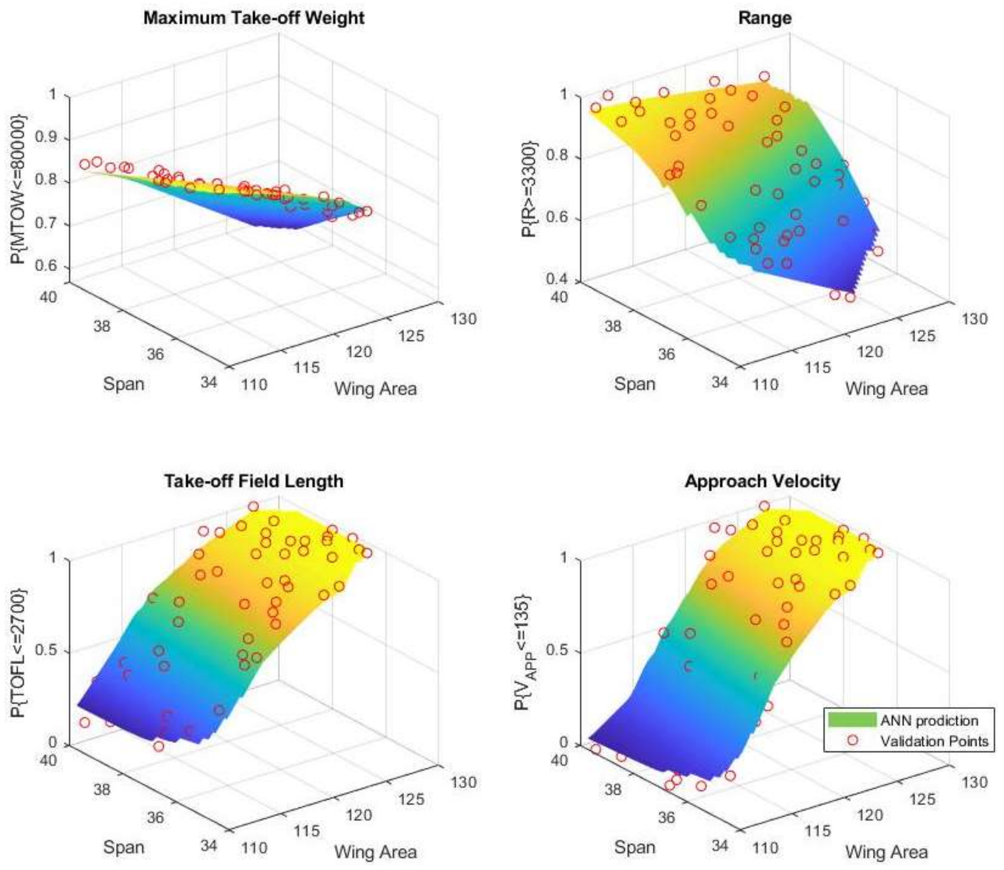

Figure 9 – ANN prediction and validation points for each constraint

Figure 10 illustrates the contour lines corresponding to different requirements on the probabilities of constraint satisfaction (from 60% to 80%). The area shaded in light blue is the feasible region where the design points can meet all the constraints with the required probability. In this specific case, if the requirement is increased to 80%, there will be no feasible region left, which requires to change the design variable in the other dimensions or relax some of the constraints.

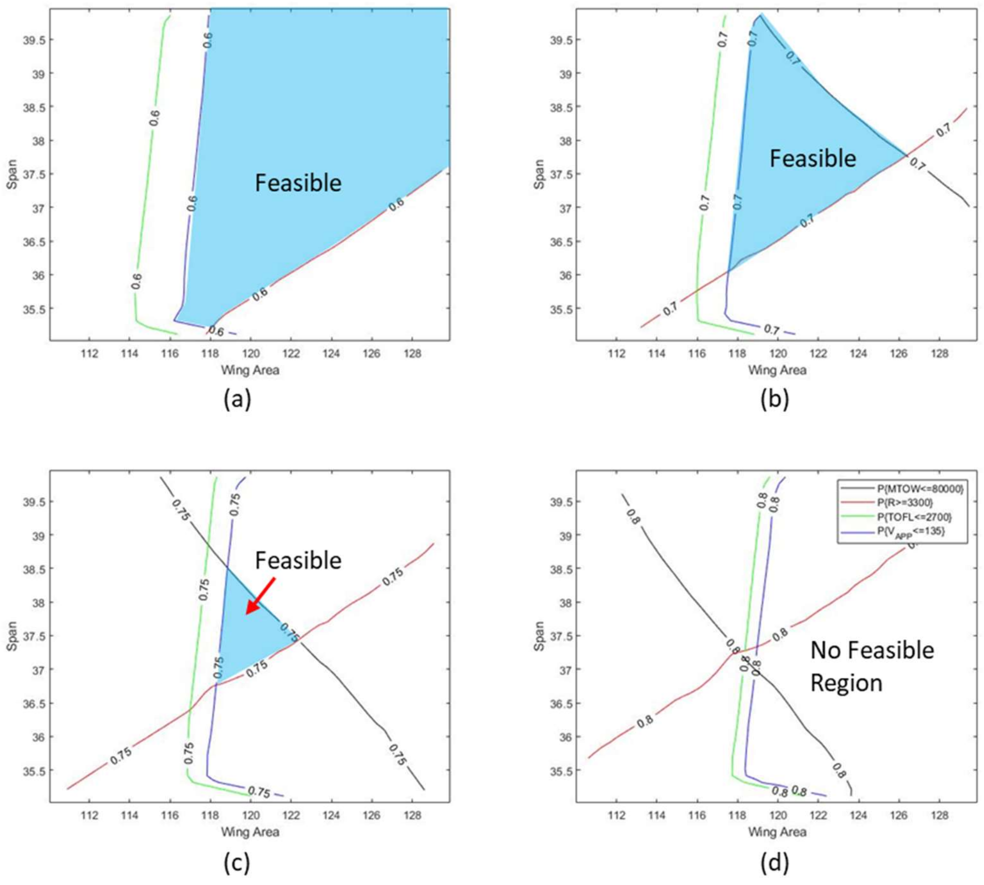

Figure 10 – Probability contour lines corresponding to different requirements.

5. Conclusion

Presented in this paper is a novel approach for computing the probability of constraint satisfaction using artificial neural networks. In this approach, the constraint satisfaction is formulated as a classification problem and the corresponding probability is obtained as a direct output of the artificial neural network. While the existing approaches require a double-loop paradigm for either training or applying the artificial neural networks, the proposed one is able to give predictions for any arbitrary point over the entire design space using only one loop of model evaluations, which could potentially reduce the computational cost.

The differences in problem formulation between the existing and proposed approaches are highlighted with an illustrative example, while the usefulness of the latter is further evaluated with an aircraft sizing test-case. It has been demonstrated that the proposed approach is able to give reasonably accurate predictions for realistic design problem. The current limitation is that the proposed approach cannot reproduce the probability distributions for the output. In addition, the ANN classifier will introduce some numerical errors by itself.

Future work will explore different ANN architectures and training procedures to further improve the accuracy and efficiency of the proposed approach. A quantitative comparison study will be performed to investigate the pros and cons of the existing and proposed approaches. The expected impact factors include the amount of uncertainty and the mathematical characteristics of the objective and constraint functions. For instance, a complex objective function will increase the number of outer loop iterations in the existing approaches, which makes the proposed approach a better option in terms of computational cost. Another direction of this research is to implement the proposed approach with genetic algorithms for optimization or set-based design space exploration [46–48], as these two processes generate large numbers of design samples which can be reused for the analysis of uncertainty.

References

[1] Zang, T. A., Hemsch, M. J., Hilburger, W., Kenny, S. P., Luckring, J. M., Maghami, P., Padula, L., and Jefferson, W. “Needs and Opportunities for Uncertainty-Based Multidisciplinary Design Methods for Aerospace Vehicles.” *NASA Technical Report*, 2002.

[2] Rocquigny, E. de., Devictor, Nicolas., and Tarantola, Stefano. *Uncertainty in Industrial Practice : A Guide to Quantitative Uncertainty Management*. John Wiley & Sons, Chichester, 2008.

[3] Schuëller, G. I., and Jensen, H. A. “Computational Methods in Optimization Considering Uncertainties – An Overview.” *Computer Methods in Applied Mechanics and Engineering*, Vol. 198, No. 1, 2008, pp. 2–13. https://doi.org/10.1016/j.cma.2008.05.004.

[4] Yao, W., Chen, X., Luo, W., van Tooren, M., and Guo, J. “Review of Uncertainty-Based Multidisciplinary Design Optimization Methods for Aerospace Vehicles.” *Progress in Aerospace Sciences*, Vol. 47, No. 6, 2011, pp. 450–479. https://doi.org/10.1016/J.PAEROSCI.2011.05.001.

[5] Taguchi, G. *Introduction to Quality Engineering: Designing Quality into Products and Processes*. Asian Productivity Organization, Tokyo, 1986.

[6] Parkinson, A., Sorensen, C., and Pourhassan, N. “A General Approach for Robust Optimal Design.” *Journal of Mechanical Design, Transactions of the ASME*, Vol. 115, No. 1, 1993, pp. 74–80. https://doi.org/10.1115/1.2919328.

[7] Park, G.-J., Lee, T.-H., Lee, K. H., and Hwang, K.-H. “Robust Design: An Overview.” *AIAA Journal*, Vol. 44, No. 1, 2006, pp. 181–191. https://doi.org/10.2514/1.13639.

[8] Chen, W., Wiecek, M. M., and Zhang, J. “Quality Utility—A Compromise Programming Approach to Robust Design.” *Journal of Mechanical Design*, Vol. 121, No. 2, 1999, pp. 179–187. https://doi.org/10.1115/1.2829440.

[9] Du, X., and Chen, W. “Efficient Uncertainty Analysis Methods for Multidisciplinary Robust Design.” *AIAA Journal*, Vol. 40, No. 3, 2002, pp. 545–552. https://doi.org/10.2514/2.1681.

[10] Aoues, Y., and Chateauneuf, A. “Benchmark Study of Numerical Methods for Reliability-Based Design Optimization.” *Structural and Multidisciplinary Optimization*, Vol. 41, No. 2, 2010, pp. 277–294. https://doi.org/10.1007/s00158-009-0412-2.

[11] Valdebenito, M. A., and Schuëller, G. I. A Survey on Approaches for Reliability-Based Optimization. *Structural and Multidisciplinary Optimization*. 5. Volume 42, 645–663. Accessed May 30, 2021.

[12] Lee, S. H., and Chen, W. “A Comparative Study of Uncertainty Propagation Methods for Black-Box-Type Problems.” *Structural and Multidisciplinary Optimization*, Vol. 37, No. 3, 2009, pp. 239–253. https://doi.org/10.1007/s00158-008-0234-7.

[13] Padulo, M., Campobasso, M. S., and Guenov, M. D. Comparative Analysis of Uncertainty Propagation Methods for Robust Engineering Design, In *Proceedings of ICED 2007, the 16th International Conference on Engineering Design*, Paris, France, 2007.

[14] Mares, C., Friswell, M., Mottershead, J., Kingdom, U., and Hill, B. “Review of Parameter Uncertainty Propagation Methods in Structural Dynamic Analysis.” *Proceedings of ISMA2002*, 2002, pp. 1853–1860.

[15] McCulloch, W. S., and Pitts, W. “A Logical Calculus of the Ideas Immanent in Nervous Activity.” *The Bulletin of Mathematical Biophysics*, Vol. 5, No. 4, 1943, pp. 115–133. https://doi.org/10.1007/BF02478259.

[16] Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A. E., and Arshad, H. State-of-the-Art in Artificial Neural Network Applications: A Survey. *Heliyon*. 11. Volume 4, e00938. Accessed May 17, 2021.

[17] Schmidhuber, J. Deep Learning in Neural Networks: An Overview. *Neural Networks*. Volume 61, 85–117. Accessed May 19, 2021.

[18] Rosenblatt, F. “The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain.” *Psychological Review*, Vol. 65, No. 6, 1958, pp. 386–408. https://doi.org/10.1037/h0042519.

[19] Cybenko, G. “Approximation by Superpositions of a Sigmoidal Function.” *Mathematics of Control, Signals, and Systems*, Vol. 2, No. 4, 1989, pp. 303–314. https://doi.org/10.1007/BF02551274.

[20] Hornik, K., Stinchcombe, M., and White, H. “Multilayer Feedforward Networks Are Universal Approximators.” *Neural Networks*, Vol. 2, No. 5, 1989, pp. 359–366. https://doi.org/10.1016/0893-6080(89)90020-8.

[21] Haykin, S. *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, USA, 1998.

[22] Aggarwal, C. C. *Neural Networks and Deep Learning*. Springer, 2018.

[23] Werbos, P. J. *Beyond Regression New Tools for Prediction and Analysis in the Behavioral Sciences.* Cambridge, MA, 1974.

[24] Parker, D. B. *Learning-Logic: Casting the Cortex of the Human Brain in Silicon*. Cambridge, MA, 1985.

[25] LeCun, Y. Une Procedure d’apprentissage Pour Reseau a Seuil Asymmetrique (A Learning Scheme for Asymmetric Threshold Networks). 1985.

[26] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning Internal Representations by Error Propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*, MIT Press, Cambridge, MA, USA, 1986, pp. 318–362.

[27] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. “Learning Representations by Back-Propagating Errors.” *Nature*, Vol. 323, No. 6088, 1986, pp. 533–536. https://doi.org/10.1038/323533a0.

[28] Molina-Cristóbal, A., Nunez, M., Guenov, M. D., Laudan, T., and Druot, T. Black-Box Model Epistemic Uncertainty at Early Design Stage. An Aircraft Power-Plant Integration Case Study. In *29th Congress of the International Council of the Aeronautical Sciences*, St Petersburg, Russia, 2014.

[29] Chapman, O. J. v., and Crossland, A. D. Neural Networks in Probabilistic Structural Mechanics. In *Probabilistic*

Structural Mechanics Handbook, Springer US, 1995, pp. 317–330.

[30] Papadrakakis, M., Papadopoulos, V., and Lagaros, N. D. "Structural Reliability Analyis of Elastic-Plastic Structures Using Neural Networks and Monte Carlo Simulation." *Computer Methods in Applied Mechanics and Engineering*, Vol. 136, No. 1–2, 1996, pp. 145–163. https://doi.org/10.1016/0045-7825(96)01011-0.

[31] Papadrakakis, M., and Lagaros, N. D. "Reliability-Based Structural Optimization Using Neural Networks and Monte Carlo Simulation." *Computer Methods in Applied Mechanics and Engineering*, Vol. 191, No. 32, 2002, pp. 3491–3507. https://doi.org/10.1016/S0045-7825(02)00287-6.

[32] Lagaros, N. D., Plevris, V., and Papadrakakis, M. "Neurocomputing Strategies for Solving Reliability-Robust Design Optimization Problems." *Engineering Computations (Swansea, Wales)*, Vol. 27, No. 7, 2010, pp. 819–840. https://doi.org/10.1108/02644401011073674.

[33] Giovanis, D. G., Papadopoulos, V., Lagaros, N. D., and Papadrakakis, M. Structural Reliability Analysis Using Subset Simulation and Neural Networks. In *10th International Conference on Structural Safety and Reliability (ICOSSAR2009)*, Osaka, Japan, pp. 13-17, 2009.

[34] Papadopoulos, V., Giovanis, D. G., Lagaros, N. D., and Papadrakakis, M. "Accelerated Subset Simulation with Neural Networks for Reliability Analysis." *Computer Methods in Applied Mechanics and Engineering*, Vols. 223–224, 2012, pp. 70–80. https://doi.org/10.1016/j.cma.2012.02.013.

[35] Hurtado, J. E., and Alvarez, D. A. "Neural-Network-Based Reliability Analysis: A Comparative Study." *Computer Methods in Applied Mechanics and Engineering*, Vol. 191, No. 1–2, 2001, pp. 113–132. https://doi.org/10.1016/S0045-7825(01)00248-1.

[36] Hurtado, J. E., and Alvarez, D. A. "An Optimization Method for Learning Statistical Classifiers in Structural Reliability." *Probabilistic Engineering Mechanics*, Vol. 25, No. 1, 2010, pp. 26–34. https://doi.org/10.1016/j.probengmech.2009.05.006.

[37] Hurtado, J. E. "Analysis of One-Dimensional Stochastic Finite Elements Using Neural Networks." *Probabilistic Engineering Mechanics*, Vol. 17, No. 1, 2001, pp. 35–44. https://doi.org/10.1016/S0266-8920(01)00011-X.

[38] Hurtado, J. E. "Neural Networks in Stochastic Mechanics." *Archives of Computational Methods in Engineering*, Vol. 8, No. 3, 2001, pp. 303–342. https://doi.org/10.1007/BF02736646.

[39] Patel, J., and Choi, S. K. "Classification Approach for Reliability-Based Topology Optimization Using Probabilistic Neural Networks." *Structural and Multidisciplinary Optimization*, Vol. 45, No. 4, 2012, pp. 529–543. https://doi.org/10.1007/s00158-011-0711-2.

[40] Gorguluarslan, R., Kim, E. S., Choi, S. K., and Choi, H. J. "Reliability Estimation of Washing Machine Spider Assembly via Classification." *International Journal of Advanced Manufacturing Technology*, Vol. 72, No. 9–12, 2014, pp. 1581–1591. https://doi.org/10.1007/s00170-014-5745-3.

[41] Hurtado, J. E., and Alvarez, D. A. "Classification Approach for Reliability Analysis with Stochastic Finite-Element Modeling." *Journal of Structural Engineering*, Vol. 129, No. 8, 2003, pp. 1141–1149. https://doi.org/10.1061/(ASCE)0733-9445(2003)129:8(1141).

[42] Gorguluarslan, R. M., and Choi, S. K. Predicting Reliability of Structural Systems Using Classification Method. In *ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Oregon, USA, Vol 2A, 2013.

[43] Lehký, D., and Novák, D. "Solving Inverse Structural Reliability Problem Using Artificial Neural Networks and Small-Sample Simulation." *Advances in Structural Engineering*, Vol. 15, No. 11, 2012, pp. 1911–1920. https://doi.org/10.1260/1369-4332.15.11.1911.

[44] Lehký, D., Slowik, O., and Novák, D. "Reliability-Based Design: Artificial Neural Networks and Double-Loop Reliability-Based Optimization Approaches." *Advances in Engineering Software*, Vol. 117, 2018, pp. 123–135. https://doi.org/10.1016/j.advengsoft.2017.06.013.

[45] Value Improvement through a Virtual Aeronautical Collaborative Enterprise (VIVACE). http://cordis.europa.eu/project/rcn/72825_en.html. Accessed Jun. 11, 2017.

[46] Singer, D. J., Doerry, N., and Buckley, M. E. "What Is Set-Based Design?" *Naval Engineers Journal*, Vol. 121, No. 4, 2009, pp. 31–43. https://doi.org/10.1111/j.1559-3584.2009.00226.x.

[47] Sobek, D. K., Ward, A., and Liker, J. "Toyota's Principles of Set-Based Concurrent Engineering." *Sloan Management Review*, 1999. https://doi.org/10.1017/CBO9781107415324.004.

[48] Riaz, A., Guenov, M. D., and Molina-Cristóbal, A. "Set-Based Approach to Passenger Aircraft Family Design." *Journal of Aircraft*, Vol. 54, No. 1, 2017, pp. 310–326. https://doi.org/10.2514/1.C033747.