aerospace

MDPI

Article

Mission-Driven Inverse Design of Blended Wing Body Aircraft with Machine Learning

Rohan S. Sharma * and Serhat Hosder

Department of Mechanical and Aerospace Engineering, Missouri University of Science and Technology, Rolla, MO 65409, USA; hosders@umsystem.edu
* Correspondence: rssgdm@umsystem.edu

Abstract: The intent of this work was to investigate the feasibility of developing machine learning models for calculating values of airplane configuration design variables when provided time-series, mission-informed performance data. Shallow artificial neural networks were developed, trained, and tested using data pertaining to the blended wing body (BWB) class of aerospace vehicles. Configuration design parameters were varied using a Latin-hypercube sampling scheme. These data were used by a parametric-based BWB configuration generator to create unique BWBs. Performance for each configuration was obtained via a performance estimation tool. Training and testing of neural networks was conducted using a K-fold cross-validation scheme. A random forest approach was used to determine the values of predicted configuration design variables when evaluating neural network accuracy across a blended wing body vehicle survey. The results demonstrated the viability of leveraging neural networks in mission-dependent, inverse design of blended wing bodies. In particular, feed-forward, shallow neural network architectures yielded significantly better predictive accuracy than cascade-forward architectures. Furthermore, for both architectures, increasing the number of neurons in the hidden layer increased the prediction accuracy of configuration design variables by at least 80%.

Keywords: inverse design; neural networks; airplane design; blended wing body; design space; performance analysis; mission profile; random forest

Citation: Sharma, R.S.; Hosder, S. Mission-Driven Inverse Design of Blended Wing Body Aircraft with Machine Learning. *Aerospace* **2024**, *11*, 137. https://doi.org/10.3390/aerospace11020137

Academic Editors: Olivia J. Pinon Fischer and Sergey Leonov

Received: 1 December 2023
Revised: 14 January 2024
Accepted: 29 January 2024
Published: 5 February 2024

1. Introduction

Preliminary airplane sizing involves the synthesis of market requirements and objectives (MR&Os) [1] while applying constraints and spatial considerations to configure an integrated vehicle. For the conventional tube and wing (TAW) airplane sizing problem, design space exploration is typically leveraged in this early portion of the design cycle to accelerate the assessment of configurations and aid in configuration design decision making [2], such as deciding on a suitable wingspan (b) and aspect ratio (AR). By doing so, an airplane designer can converge on an optimum configuration that satisfies MR&O targets and can proceed to the more detailed design phase. This process has been well studied and understood regarding TAW aircraft; however, as MR&Os become more stringent around aircraft performance and its environmental impacts [3], exploration of more novel concepts is warranted. A promising concept that has been researched and is still being studied by both academia and the industry is the blended wing body (BWB). As stated by Wakayama et al. [4], the BWB requires a design approach that departs from the conventional TAW methodology—rather than decomposing the airplane into distinct pieces, the wing, fuselage, engines, and empennage are all tightly integrated to achieve a substantial performance improvement. Furthermore, although design space exploration of novel concepts, such as the BWB, have been studied previously [5,6], the lack of validated models for these novel concepts and their application to traditional direct design problems can result in selecting a point-design configuration prematurely, or even incorrectly as

Kallou et al. [7] pointed out. In this context, contrary to typical direct design approaches of BWBs, an examination into the application of inverse design of BWBs is warranted.

The motivation for the work described in this paper is to determine whether machine learning models can be used for the mission-informed inverse design of unconventional airplane configurations while leveraging unique, physics-derived data types to generate credible results. In such a scenario, the neural network models are designed to predict the values of a set of airplane configuration design parameters, such as b, wing loading (W/L), maximum takeoff weight ($MTOW$), etc., when supplied with specific time-dependent performance metrics, such as lift-to-drag ratio as a function time ($L/D(t)$). Here, each time step reflects a different point in the vehicle's mission, which is typically composed of taxi, take-off, climb, cruise, descent, approach, landing, and reserve segments. An advantage of inverse design is that the mission-based objectives are fixed while a suitable design is calculated, while with a typical direct optimization scheme, a design is iterated to arrive at performance that may be close, but not an exact match, to those objectives. One reason for exploring the viability of machine learning models is traditional surrogate models, such as response surface functions, are typically not suitable for mapping multiple multi-dimensional inputs to more than one singular output, for example, when the inputs are a set of time-series-based variables that correspond to a unique combination of configuration design data. Artificial neural networks (ANNs), in particular, offer an advantage over response surface models because, when architected and trained appropriately, they can robustly handle multi-dimensional, highly non-linear data relationships well, especially when enough training data are provided [8]. This implies that if architected correctly, an ANN could accurately generate a set of outputs when provided multiple multi-dimensional inputs, even if the inputs (e.g., time-series aircraft performance data) have highly non-linear relationships with the outputs (e.g., aircraft configuration design data).

The author's previously published research [9] focused on developing a framework to support the aforementioned objective, albeit within the context of applying it to conventional, TAW, single-aisle, twin-engine, commercial transport category airplane configurations such as those in the same class as the Boeing 737 Next Generation family. The work outlined in this paper is an expansion and adaptation of the authors' previous work, focusing specifically on application to the inverse design of unconventional airplane configurations, namely BWBs. Compared to previously published research, changes have been introduced to the framework, namely a larger airplane design and performance database; more robust neural network model generation, training, and testing methods; and an improved approach for extensibility analysis.

Within the context of airplane configuration design, inverse design is the antithesis of a direct design approach. In an inverse-design process, configuration design parameters are instead obtained through a query of a design space from a performance target standpoint. Such a scenario would typically start with a set of MR&Os, and the result would be an airplane configuration that closely satisfies, if not meets, the MR&Os [10]. In short, it differs from other design methods in that the design parameters are a result of the method, rather than an input into it. Gibbs et al. [11] developed an airplane inverse-design method, where the input was a desired fixed operating cost per passenger, and the output was the vehicle's corresponding geometry—fuselage length, wingspan, horizontal and vertical stabilizer spans, and thrust. However, the use of neural network models for inverse design of airplane configurations has been very limited.

The use of models based on ANNs has emerged as a potential contender for response surface models in the context of inverse-design problems. This is in part due to their improved efficiency in terms of computational wall-clock time needed for calibration and their prediction accuracy compared to other surrogate models, as highlighted by Sekar et al. [12]. Rai et al. [13] have also shown that through adjustment of neural network architecture (number of layers and neurons in each layer) and neural network numerical methods (regularization algorithms, training functions, and optimization routines), overfitting ten-

dencies induced by highly non-linear, multi-dimensional features in the training database can be reduced.

Leveraging neural network models for inverse-design problems has been widely researched, particularly for airfoil design scenarios. Compared to a direct optimization scheme, Barrett et al. [14] illustrated promise in an inverse-design methodology as it applied to airfoil geometry definition. Kharal et al. [15] demonstrated promise in the concept by developing models capable of shape generation based on values of lift coefficient (C_L), drag coefficient (C_D), and pitching moment coefficient (C_M) at various angles of attack (α). Extending this research, Yilmaz et al. [16] showed that by leveraging a deep learning model instead, shape parameterization can be avoided thereby operating at a lower level of abstraction by detecting patterns and features in the data themselves. Glaws et al. [17] expanded on previous airfoil inverse-design research by effectively showing that an invertible neural network (INN) could be developed, which is suitable for both direct and inverse airfoil design.

Outside the realm of airfoil inverse design, Yu et al. [18] demonstrated promise in the idea of developing neural network models for the inverse design of rocket nozzles when provided a desired pressure distribution, showcasing its excellent predictive accuracy. Additionally, Oddiraju et al. [19] demonstrated the viability of developing neural network models to aid in the design of metamaterials when using desired bandgap specifications. Li et al. [20] showcased the ability to use deep neural networks for prediction of 3-D wing shape designs when provided C_L, C_D, C_M, and pressure coefficient (C_P) distributions, and how such a model could be leveraged by a gradient-based optimization framework for various wing design scenarios.

More broadly in the aerospace field, neural networks have been widely implemented in Reynolds-averaged Navier–Stokes (RANS) solutions. While Thuerev et al. [21] focused on neural network applications to RANS solutions, a more extensible neural network development framework was leveraged by Singh et al. [22], who showed merit in using neural networks to augment the Spalart–Allmaras turbulence models ultimately aiding in surface pressures predictions. Li et al. [23] comprehensively summarized how machine learning has solved some challenges in aerodynamic space optimization (ASO), specifically as it pertains to geometric design space [24], aerodynamic evaluations—namely, aerodynamic coefficient approximation [25–28] and flow field modeling [29]—and optimization architecture [30,31]. However, there are only a few previous works detailing the use of neural network models for the analysis of airplane configurations; for example, Secco et al. [32] developed a neural network model effectively replacing a full-potential code for prediction of aerodynamic coefficients when provided configuration design data and flight condition.

This paper is organized as follows: Section 2 describes the computational approach, including initial geometry parameterization, database generation methodology, neural network algorithm selection and architecture variation, training and testing through a K-fold cross-validation scheme, extensibility analysis of the neural networks across a BWB vehicle survey, and a random forest approach to the prediction of design variables. Section 3 details the results as it pertains to neural network performance and extensibility analysis. Finally, the conclusions of this study, and findings that warrant further investigation, are elaborated in Section 4.

2. Computational Approach and Methodology

The outline of the computational approach, along with what programming languages are used, is illustrated in Figure 1 (The second-segment thrust correction process is explained in Section 2.1.2). First, a baseline BWB vehicle is used to construct a configuration parametrization model. Then, design parameters are varied via a Latin-hypercube sampling scheme. The geometry parametrization model is then used to define a number of new, unique BWB configurations. These configurations are assessed using a physics-informed, Level-0 airplane performance assessment tool. Following this step, a database is created containing design and performance data for each configuration. Numerical scaling is

then performed on the database, and different neural network models are generated, each with its own unique architecture. Next, the scaled database is equally partitioned, where different portions are used to train and test the neural network models in a K-fold cross-validation scheme. Here, the neural networks are trained such that their inputs are BWB performance data and their targets are corresponding BWB configuration design parameter values, thus lending themselves towards the inverse-design objective.

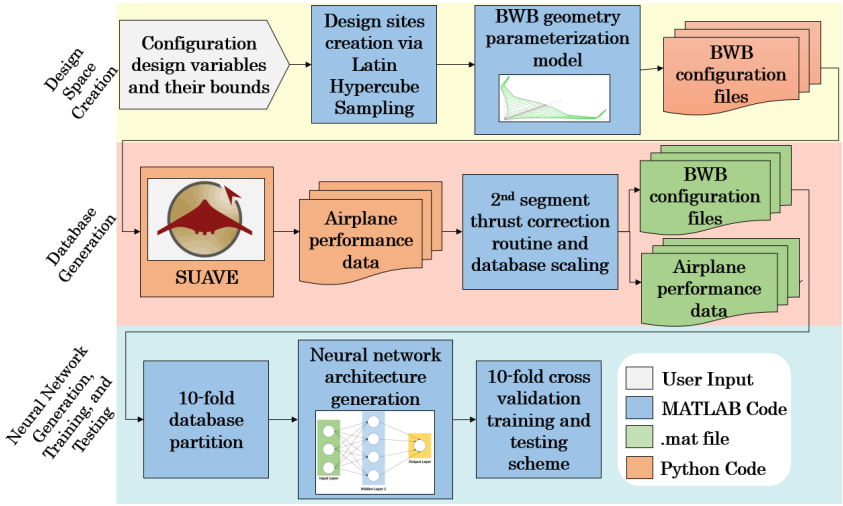

Figure 1. Computational approach overview.

To enable parametric-based airplane configuration definition and rapid performance estimation, SUAVE (data available online at https://suave.stanford.edu/ (accessed on 25 November 2022)) [33–36], a conceptual-level aircraft design environment built with the ability to rapidly configure and analyze both conventional and unconventional designs, was leveraged. SUAVE allows users to define details of an airplane configuration design programmatically via the specification of different design parameters, such as b, AR, fuselage length, $MTOW$, sea-level static thrust, etc. These values are stored and ultimately constitute an aircraft configuration design file. This file is then processed through SUAVE's mission solver, where a fixed mission profile is utilized to obtain performance data for a configuration—range, total weight, specific fuel consumption (SFC), C_L, C_D, L/D, and so on—all as a function of time steps in the vehicle's mission. SUAVE's comparatively low-fidelity aerodynamics and stability derivative solvers can be augmented with relatively higher-fidelity methods such as Athena Vortex Lattice (AVL (data available online at http://web.mit.edu/drela/Public/web/avl (accessed on 25 November 2022))) or SU2 (data available online at https://su2code.github.io/ (accessed on 25 November 2022)), thereby enabling multi-fidelity analysis. AVL is a tool for aerodynamic and flight-dynamic analysis of rigid aircraft, and can be used to analyze both TAW aircraft as well as unconventional configurations such as BWBs. It employs an extended vortex lattice model for the lifting surfaces, together with a slender-body model for fuselages and nacelles. The flight-dynamic analysis portion of AVL combines a full linearization of the aerodynamic model about any flight state, together with specified mass properties. For the purpose of this study, the authors have elected to use AVL's combined capability with SUAVE's solvers. Meanwhile, MATLAB's Deep Learning Toolbox (data available online at https://www.mathworks.com/help/deeplearning/ (accessed on 25 November 2022)) was used to manage neural network development, training, and testing.

An additional framework is created for assessing the accuracy of developed neural networks across unknown design sites, i.e., configurations not within the training and testing database. This extensibility analysis of the models applied to a more "diverse" set of BWB vehicles is conducted through a random forest approach. Figure 2 depicts an overview of this process. The following subsections describe these processes in more detail: geometry parameterization; database creation; neural network architecture generation; numerical

method selection and training; K-fold cross-validation training and testing scheme; and the random forest approach to extensibility analysis.

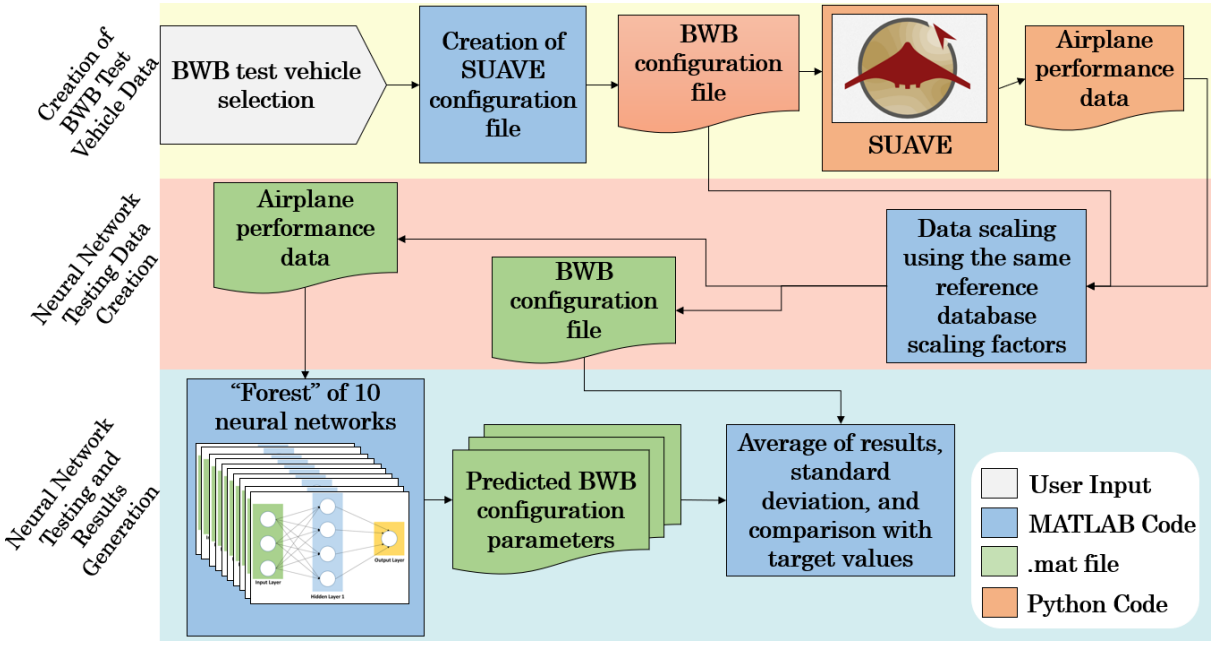

Figure 2. Overview of extensibility analysis via a random forest approach.

2.1. Design Space Creation

2.1.1. Geometry Parameterization

To develop neural network models capable of inverse design, i.e., predicting BWB configuration design parameter values when supplied with mission-dependent BWB performance data, a database was created containing two distinct parts—BWB configuration definition and its respective performance data. Typically, neural networks require a large amount of training data placing importance on the ability to generate such data in a rapid and automated manner—this is especially important in the conceptual-level airplane design cycle [37]. To enable the quick creation of BWB configurations, a parametric-based BWB geometry model was developed, where through the specification of wingspan and wing area, a new BWB vehicle design can be generated.

The design of BWBs is challenging, primarily due to the tight coupling between aerodynamic performance, trim, stability, and propulsion, not to mention the sheer number of design variables involved and the complexities with transonic flow conditions [38]. For this reason, careful consideration was placed on choosing a suitable baseline vehicle for which the parametric geometry model was calibrated on—Liebeck's BWB450 [39] was used. The baseline tail volume coefficient, $V_{v_{bl}}$, was calculated for the baseline vehicle using Equation (1). The distribution of semi-spans, $\overrightarrow{b_{bl}}$, wing areas, $\overrightarrow{S_{bl}}$, and quarter chord sweeps, $\overrightarrow{c_{\lambda_{bl}}}$, was calculated segmenting the BWB450's wing into 7 sections, including the vertical wing tip, as shown in Figure 3. Based on the properties of the BWB450, a few rule-of-thumb relationships were established that constrained the following parameters: center of gravity located at 60% root chord fraction, $CG = 0.6\psi$, two engines positioned at 90% root chord fraction and 9% wingspan fraction, $\omega_\psi = 0.9\psi$ and $\omega_\eta = 0.09\eta$, respectively. Additionally, the aerodynamic center (AC) was constrained at the same location as the center of gravity. This ensures that the vehicle is trimmed longitudinally.

$$V_{v_{bl}} = \frac{S_{tip}(AC_{tip} - CG)}{(\sum_{n=1}^{6} S_n)(\sum_{i=1}^{6} b_i)} \quad (1)$$

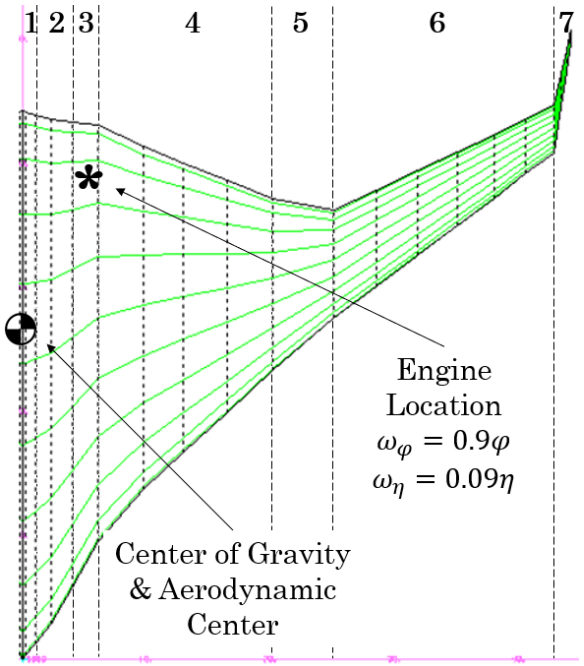

Figure 3. BWB semi-span segments.

These values ultimately inform the definition of new BWB configurations when the model is supplied with values of b and wing area, S. The overall computational process flow for this model is as follows: First, the model accepts new values of span and area as inputs—b_{in} and S_{in}. The wing and span are then distributed—$\overrightarrow{S}$ and $\overrightarrow{b}$—based on fixed proportions from the baseline design, namely $\overrightarrow{b_{bl}}$ and $\overrightarrow{S_{bl}}$. Chord length distributions are determined using the quarter chord sweep for each segment, $\overrightarrow{c_{\lambda_{bl}}}$. $\overrightarrow{S}$ is then used to calculate the tail volume coefficient for the new vehicle, V_v. This will inherently be different than $V_{v_{bl}}$ since the new configuration has a different b and S than the baseline, calibrated vehicle, namely the BWB450. Essentially, this indicates that the vertical tail area for the new configuration, S_7, is either undersized or oversized relative to b_{in} and S_{in}, and relative to the baseline vehicle. This could mean the new configuration has vertical tails that either provide insufficient longitudinal stability or more than needed. Constraining the vehicle to $V_{v_{bl}}$, the difference between the two tail volume coefficient quantities is calculated, and is used to calculate the adjusted vertical tip area and span, S_{7_a}, and span, b_{7_a}, respectively. These procedures are highlighted in Equations (2) and (3), where S_7' is the unadjusted vertical tail area, and c_{r_7} and c_{t_7} are the root chord and tip chord, respectively, of the vertical tip. Furthermore, here, $(AC_{tip} - CG)$ corresponds to the planar measurement between the aerodynamic center of the tip and the aircraft's center of gravity. Note that, depending on the difference between the two tail volume coefficients, the area and span of the tip are either reduced or increased while keeping c_{r_7} and c_{t_7} constant.

$$S_{7_a} = S_7' \pm \frac{(V_{v_{bl}}(\sum_{n=1}^{6} S_n)(\sum_{i=1}^{6} b_i))}{(AC_{tip} - CG)} \tag{2}$$

$$b_{7_a} = \frac{2S_{7_a}}{c_{r_7} + c_{t_7}} \tag{3}$$

Finally, by incorporating S_{7_a} and b_{7_a}, the span and area distributions for the new BWB configuration are adjusted—$\overrightarrow{S_a}$ and $\overrightarrow{b_a}$. These distributions are used to calculate the adjusted values of span and area, b_a and S_a, which are different than b_{in} and S_{in}. At this point, the new BWB configuration has been sized; however, it lacks other principal characteristics data such as appropriate values of $MTOW$ and sea-level static thrust available, T_A, which are tightly dependent on geometry for BWBs [40]. For this reason, polynomial response

surface functions of $MTOW$ and T_A were developed using a vehicle survey, consisting of a variety of configurations beyond the BWB450. The vehicle survey contained three categories of configurations, namely BWBs, hybrid wing bodies (HWBs), and integrated wing bodies (IWBs). The authors elected to leverage a variety of vehicle types, closely related to BWBs, in order to diversify the dataset and improve neural network generalization across a broader set of BWB vehicles. These vehicles are illustrated in Figure 4.

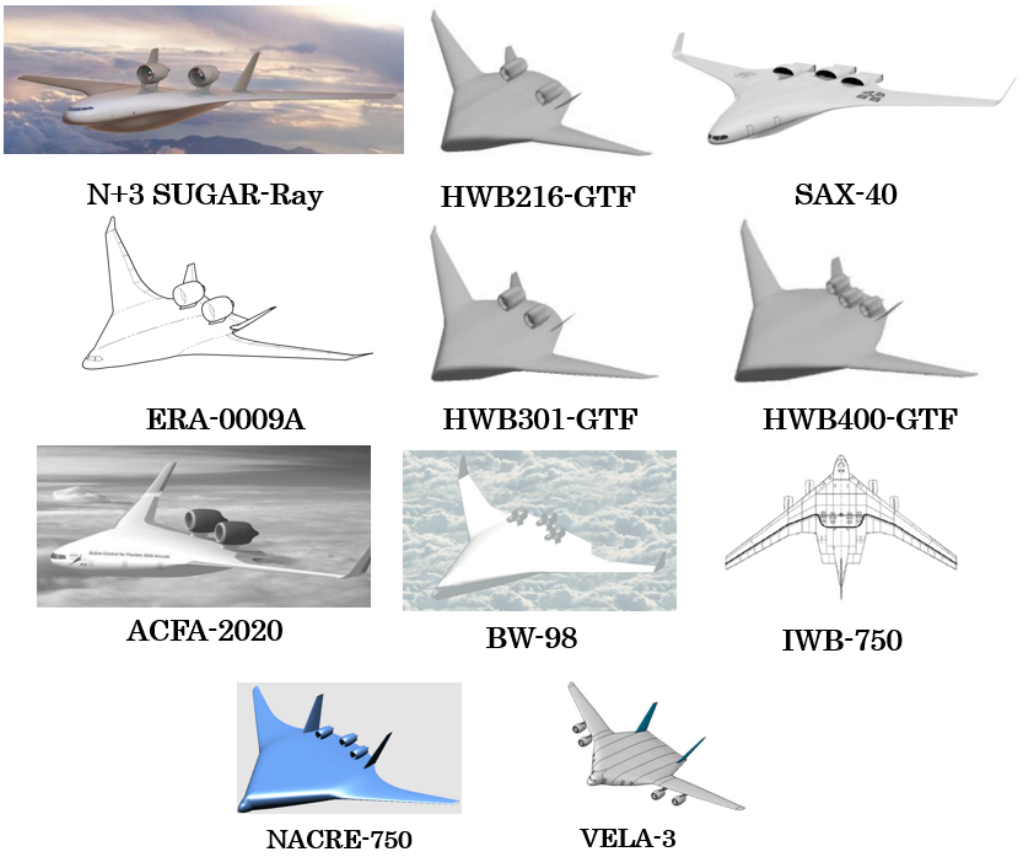

Figure 4. Illustration of BWB configurations in vehicle survey [41–49].

Using these vehicles and their configuration characteristics data—highlighted in Table 1—polynomial functions, of different orders, were developed for $MTOW$ and T_A, both expressed in pounds. For each function, the coefficient of determination, R^2 value, and root mean squared error (RMSE) were calculated. Ultimately, for both $MTOW$ and T_A, the functions with the highest R^2 value and lowest RMSE were selected. The equations for $MTOW$, as a function of S_a, ft^2, and b_a, ft., and the equation for T_A, as a function of $MTOW$ and S_a, ft^2, are expressed in Equation (4) and Equation (5), respectively. They both exhibit R^2 values of 0.999 and RMSE values of 0.33 and 0.21, respectively. From a first-principles BWB vehicle design standpoint, and based on the design variables selected for the design space, wing area and wingspan can fundamentally aid in determining $MTOW$. Similarly, thrust can be approximated using area and $MTOW$ of the vehicle.

$$MTOW(b_a, S) = (-4.663 \times 10^6) + (7.337 \times 10^4)b - 555.3S - 233.2b^2 - 2.494bS + (8.696 \times 10^2)S^2 + 0.027b^2S - (5.83 \times 10^{-4})bS^2 + (2.249 \times 10^{-6})S^3 \quad (4)$$

$$T_A(S_a, MTOW) = (-4.632 \times 10^5) + 157.6S - 0.6734MTOW - (9.835 \times 10^{-4})SMTOW - (1.065 \times 10^{-5})MTOW^2 - (1.418 \times 10^{-9})SMTOW^2 + (1.933 \times 10^{-11})MTOW^3 + (6.324 \times 10^{-16})SMTOW^3 - (9.181 \times 10^{-18})MTOW^4 \quad (5)$$

Table 1. BWB vehicle survey and surrogate model data [41–49].

Vehicle	*MTOW* [lbs]	S_{ref} [ft^2]	Span [ft]	Total Thrust [lbs]
N + 3 SUGAR-Ray	181,500	4136	168.5	56,000
HWB216-GTF	312,500	8221	220	92,000
SAX40	330,300	8998	221.6	92,500
ERA-0009A	411,250	8048	229.3	95,000
HWB301-GTF	533,000	10,169	250	134,500
HWB400-GTF	701,000	11,471	260	168,500
ACFA-2020	884,000	14,291	261.9	239,000
BW-98	1,060,000	14,968	254.27	296,500
IWB-750	1,262,000	17,093	328.08	353,000
NACRE-750	1,390,000	21,453	328.08	468,500
VELA-3	1,542,000	22,088	326.76	432,500

Using these functions, *MTOW* and T_A of the BWB airplane are calculated, and the parametric-based BWB configuration generation process is complete. The output of this model is a SUAVE configuration file, capable of being assessed through SUAVE's configuration assessment functionality. A process flow of this routine is shown in Figure 5.

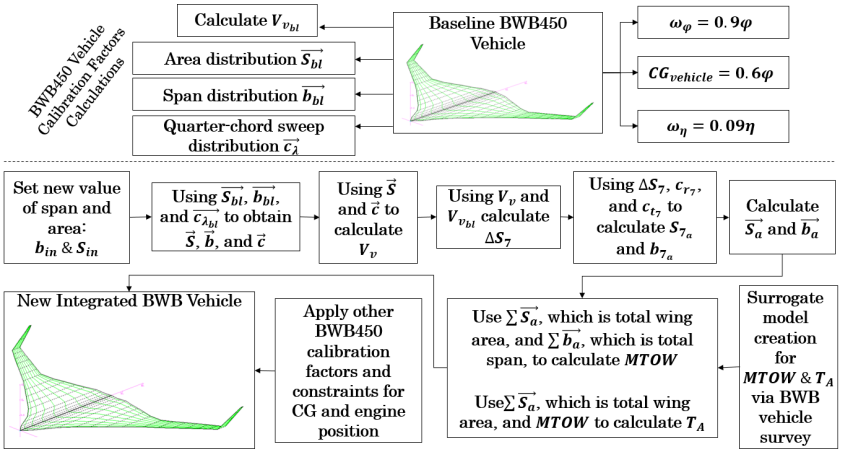

Figure 5. BWB parametric configuration creation model overview.

2.1.2. Database Creation

BWB configurations are initially varied and created using two design variables, namely b_{in} and S_{in}, via the parametric-based configuration generation model. This model utilizes these two parameters as inputs to ultimately create a unique BWB configuration defined by four design parameters—b_a, S_a, *MTOW*, and T_A. These four design variables, along with the vehicle's corresponding performance data, are used to train neural network models. In this sense, the objective of this work is to develop neural network models such that the aforementioned design parameters for a BWB configuration are calculated when supplied with the configuration's mission-informed, time-dependent performance data.

Four design parameters, namely b_a, S_a, *MTOW*, and T_A, were selected as they are generally regarded as high-level design parameters and are typically investigated in the preliminary and conceptual sizing portion of a BWB's design cycle. During this phase, design space exploration is conventionally utilized to explore the trade space and understand design sensitives as it relates to both the MR&Os and the aircraft manufacturer's product and technology capability [50]. The bounds for b_{in} and S_{in} design space were selected as 169 ft. $\leq b_{in} \leq$ 327 ft. and 4200 ft.^{2} $\leq S_{in} \leq$ 22,000 ft.^{2}, respectively. In general, referencing

the BWB vehicle survey table (Table 1), the N + 3 SUGAR-Ray configuration was used to inform the lower bounds of both b_{in} and S_{in}, while the VELA-3's configuration characteristics informed the upper limit for these values. Since the $MTOW$ and T_A polynomial response surface functions are also based on vehicles in Table 1, the bounds applied for the design space mitigated errors exhibited by these functions.

The method of numerical variation employed was the Latin-hypercube sampling (LHS) scheme. Compared to other statistical sampling methods, the LHS scheme provides broader coverage of the design space via the distribution of samples in equally spaced probability bins [51]. In the context of neural network training, this is particularly beneficial as it exposes the model to well-distributed data, which in turn can increase overall predictive accuracy and reduce the likelihood of overfitting [52]. Additionally, Loh [53] has shown that for problems characterized by multidimensionality, surrogate model development is typically faster when using data obtained from an LHS scheme as opposed to Monte Carlo sampling methods. Figure 6 depicts a visualization of 100 LHS-derived points of b_{in} and S_{in}.

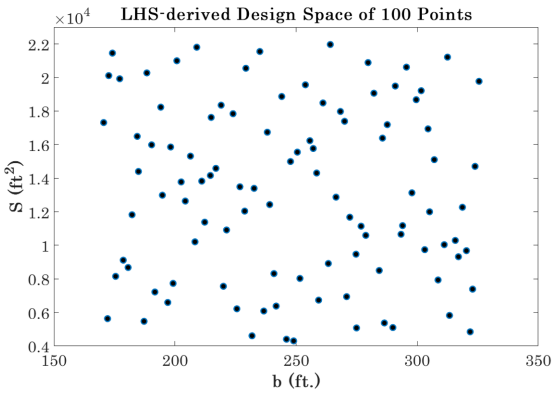

Figure 6. Visualization of 100 LHS-derived span and wing area combinations.

The LHS sample size, N_{LHS}, utilized was 30,000 as this was deemed to provide both adequate design space coverage and a suitable number of training points for each fold in the K-fold cross-validation scheme (method explained in Section 2.3). However, from a numerical sensitivity standpoint determining the optimal value of N_{LHS} was not a focus of this study. Each unique combination of b_{in} and S_{in} is used by the BWB parametric configuration generation model, which generates adjusted values of span and area, namely b_a and S_a, along with $MTOW$, and T_A. In conjunction with spatial integration constraints and assumptions, this model uses the aforementioned design parameters and programmatically generates 30,000 unique SUAVE airplane configuration files. An example of one such configuration is depicted in Figure 7, where b_a is 221.6 ft., S_a is 8998 ft^2, $MTOW$ is 408,300 lbs., and T_A is 116,000 lbs.

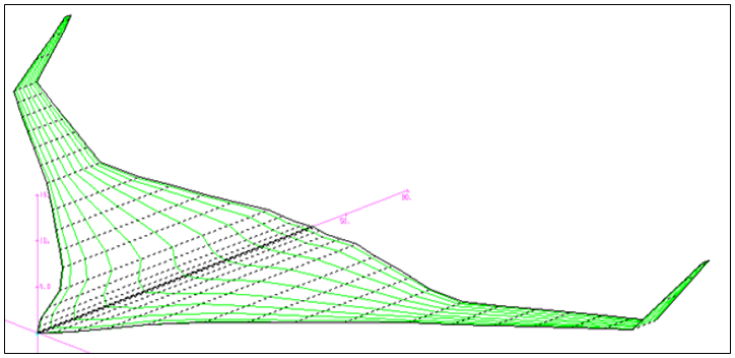

Figure 7. Visualization of a BWB generated using parametric configuration model.

These configuration files are then run through SUAVE's performance analysis routines, which assess every BWB configuration through the same mission constraints, and for that matter, the same mission flight profile. This flight profile represents a typical long-range, transport category, Part 25 mission with constraints on climb and descent gradients, rate of climbs and descents, speeds for different segments, and altitude restrictions [54]. The 6500 nautical mile mission is composed of first, second, and third climb segments; a cruise segment; and first, second, third, fourth, and fifth descent segments. The flight profile is visualized in Figure 8. The 6500 nautical mile range was chosen as this represents a notional, long-range mission that a vehicle similar to what the BWB450 may routinely fly in-service.

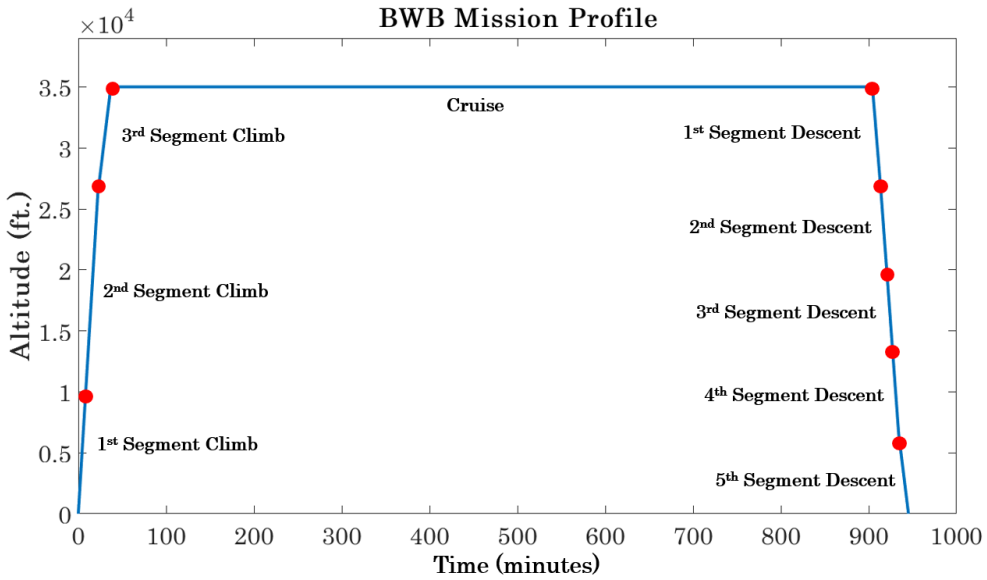

Figure 8. SUAVE flight profile for BWB performance assessment.

Coupled with AVL's aerodynamics and stability and control (S&C) analysis capabilities, SUAVE's mission solver assesses each configuration and generates its respective output files in an automated manner. For BWBs, the mission solver uses b_a and S_A for cruise C_L, maximum C_L, and L/D calculations and for S&C calculations since both design variables fundamentally inform the shape of the configuration. *MTOW* is used by SUAVE's in-built weight estimation routines to inform the weight of the vehicle at the start of the mission. T_A represents the total static thrust available for the airplane and primarily determines the engine mass flow and fuel flow at full throttle and adjustments to this quantity are made based on standard atmosphere fluctuations.

The SUAVE-generated output files contain different time-series airplane performance data, corresponding to the configuration assessed. These output files are processed to obtain the following performance parameters, all as a function of time: coefficient of drag attributed to compressibility effects (C_{D_c}), induced drag coefficient (C_{D_i}), miscellaneous drag coefficient (C_{D_m}), parasitic drag coefficient (C_{D_p}), L/D, weight (W), specific fuel consumption (SFC), and thrust required (T_R). Each parameter is composed of 144 discrete time steps, essentially meaning that each performance parameter is a row vector consisting of 144 elements. Here, the time steps are not linearly spaced from each other, i.e., the time difference between each time step is unique. This is because SUAVE's mission solver discretizes time based on the Chebyshev polynomial, which has been shown to improve convergence and data accuracy near transition portions of the mission and mitigate Runge's phenomenon. This is exhibited by each performance parameter having smaller time steps at the beginning and towards the end of each mission segment.

At this stage, design data—b_a, S_a, $MTOW$, and T_A—have been collected for each BWB configuration along with its associated airplane performance data—$\overrightarrow{C_{D_c}}$, $\overrightarrow{C_{D_i}}$, $\overrightarrow{C_{D_m}}$, $\overrightarrow{C_{D_p}}$,

$\overrightarrow{L/D}$, $\overrightarrow{W}$, $\overrightarrow{SFC}$, and $\overrightarrow{T_R}$. However, a numerical discrepancy may exist between T_A and values in $\overrightarrow{T_R}$. Although polynomial response surface models have been developed using a BWB vehicle survey, to calculate $MTOW$ and T_A as a function of b_a, and S_a, it is still theoretically possible to calculate a value for T_A that does not represent the optimal value for the vehicle. Here, the optimal value of T_A is defined as that value that provides minimal, i.e., as close to zero, second-segment climb excess thrust. Second-segment climb takes place after takeoff ground roll, rotation, liftoff, and first-segment climb (first-segment climb occurs after liftoff where the aircraft is still in a takeoff configuration). Under single-engine operation, the aircraft must be able to produce enough thrust to sustain a positive and steady climb gradient at the liftoff speed. Upon achieving this speed, the landing gear is retracted, which signals the completion of the first-segment climb and the start of the second-segment climb. During this phase, a two-engine aircraft is expected to produce enough thrust under single-engine operations (SEO) to achieve a steady gross climb gradient of no less than 2.4% at the takeoff safety speed, $V2$, until reaching 400 ft. above ground level (AGL) [55]. The second-segment climb constraints often become a critical engine sizing condition for the entire airplane, where insufficient thrust renders the airplane unable to satisfy the 2.4% climb gradient required under single-engine operating conditions. While an excessively positive thrust margin allows the airplane to achieve the necessary climb gradient, SFC is likely penalized for the rest of the airplane's flight profile since the airplane is utilizing an engine with more thrust than the airplane needs. For this reason, SUAVE's mission solver is adjusted to modify T_A as needed to achieve a minimal second-segment excess thrust condition. Therefore, rather than using T_A for neural network training and testing, the SUAVE-adjusted value of T_A is used—$T_{A_{opt}}$. This ensures that the neural network models developed are more likely to predict values of T_A that are optimal for a BWB configuration.

Next, two discrete databases are created—a BWB design and a BWB performance database. The BWB design database, matrix $\boldsymbol{D}$, contains values of b_a, S_a, $MTOW$, and $T_{A_{opt}}$ for all 30,000 BWB configurations. Therefore, $\boldsymbol{D}$ has dimensionality 30,000 × 4, where each row represents a unique BWB configuration. Similarly, the BWB performance database, matrix $\boldsymbol{P}$, contains the values of $\overrightarrow{C_{D_c}}$, $\overrightarrow{C_{D_i}}$, $\overrightarrow{C_{D_m}}$, $\overrightarrow{C_{D_p}}$, $\overrightarrow{L/D}$, $\overrightarrow{W}$, $\overrightarrow{SFC}$, and $\overrightarrow{T_R}$ for the same 30,000 BWB configurations. Here, each performance parameter is a row vector containing 144 elements, representing time step values for the entire mission. Since each performance parameter's dimensions is 1 × 144 and there are 8 performance parameters, $\boldsymbol{P}$ has dimensionality 30,000 × 1152. For matrices $\boldsymbol{D}$ and $\boldsymbol{P}$ each row corresponds to the same configuration where for example, the values of b_a, S_a, $MTOW$, and $T_{A_{opt}}$ in row 2600 of matrix $\boldsymbol{D}$ yielded the performance in row 2600 of matrix $\boldsymbol{P}$. Matrices $\boldsymbol{D}$ and $\boldsymbol{P}$ are expressed in Equations (6) and (7).

$$\boldsymbol{D} = \begin{pmatrix} b_{a1} & S_{a_1} & MTOW_1 & T_{Aopt_1} \\ b_{a2} & S_{a_2} & MTOW_2 & T_{Aopt_2} \\ b_{a3} & S_{a_3} & MTOW_3 & T_{Aopt_3} \\ \vdots & \vdots & \vdots & \vdots \\ b_{aN_{LHS}} & S_{aN_{LHS}} & MTOW_{N_{LHS}} & T_{Aopt_{N_{LHS}}} \end{pmatrix} \tag{6}$$

$$\boldsymbol{P} = \begin{pmatrix} \overrightarrow{C_{D_c}}_1 & \overrightarrow{C_{D_i}}_1 & \overrightarrow{C_{D_m}}_1 & \overrightarrow{C_{D_p}}_1 & \overrightarrow{L/D}_1 & \overrightarrow{W}_1 & \overrightarrow{SFC}_1 & \overrightarrow{T_R}_1 \\ \overrightarrow{C_{D_c}}_2 & \overrightarrow{C_{D_i}}_2 & \overrightarrow{C_{D_m}}_2 & \overrightarrow{C_{D_p}}_2 & \overrightarrow{L/D}_2 & \overrightarrow{W}_1 & \overrightarrow{SFC}_2 & \overrightarrow{T_R}_2 \\ \overrightarrow{C_{D_c}}_3 & \overrightarrow{C_{D_i}}_3 & \overrightarrow{C_{D_m}}_3 & \overrightarrow{C_{D_p}}_3 & \overrightarrow{L/D}_3 & \overrightarrow{W}_1 & \overrightarrow{SFC}_3 & \overrightarrow{T_R}_3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \overrightarrow{C_{D_c N_{LHS}}} & \overrightarrow{C_{D_i N_{LHS}}} & \overrightarrow{C_{D_m N_{LHS}}} & \overrightarrow{C_{D_p N_{LHS}}} & \overrightarrow{L/D_{N_{LHS}}} & \overrightarrow{W_{N_{LHS}}} & \overrightarrow{SFC_{N_{LHS}}} & \overrightarrow{T_{R N_{LHS}}} \end{pmatrix} \tag{7}$$

Once $\boldsymbol{D}$ and $\boldsymbol{P}$ were created, a scaling operation was performed to create $\boldsymbol{D_S}$ and $\boldsymbol{P_S}$. The motivation for scaling stemmed from demonstrated improvement in how it reduces

neural network training time, via enhanced optimization algorithm convergence, and how it yields a reduction in prediction errors [56]. Nayak et al. suggested that to achieve the aforementioned performance improvements, the entire input and output databases for training and testing neural networks should be scaled by some reference quantities to normalize values between 0 and 1 [57]. Another benefit of scaling, as it applies to this study, is that it can adjust quantities that numerically differ by several orders of magnitude, such as T_R and SFC, as well as $MTOW$ and b. This ensures that the model's learning method is not skewed by substantially higher or lower values. In general, from a numerical standpoint, scaling was conducted by dividing variables by 1.05 multiplied by the maximum value of variables that exist within each matrix—$\boldsymbol{D}$ and $\boldsymbol{P}$. Specifically, for $\boldsymbol{D}$, this entailed dividing the $MTOW$ value in each row by 1.05 multiplied by the maximum value of $MTOW$ that appears within $\boldsymbol{D}$, i.e., $1.05 \cdot MTOW_{max}$. A new matrix, $\boldsymbol{D_S}$, was formed by repeating this operation for the other design variables in $\boldsymbol{D}$ and this is expressed in Equation (8). The value of 1.05 was informed by applying a 5% growth factor to each data point. This ensures all data points are between 0 and 1 and reduces the risk of overfitting. Once BWB configuration data was generated by the neural network model as an output, a re-scaling operation was not performed to obtain absolute values. This was to ensure that predicted values were all in the same range (0 to 1) prior to error calculation, thereby mitigating numerical sensitivity issues.

$$\boldsymbol{D_S} = \begin{pmatrix} \frac{b_{a1}}{1.05 \cdot b_{a,max}} & \frac{S_{a1}}{1.05 \cdot S_{a,max}} & \frac{MTOW_1}{1.05 \cdot MTOW_{max}} & \frac{T_{Aopt_1}}{1.05 \cdot T_{Aopt,max}} \\ \frac{b_{a2}}{1.05 \cdot b_{a,max}} & \frac{S_{a2}}{1.05 \cdot S_{a,max}} & \frac{MTOW_2}{1.05 \cdot MTOW_{max}} & \frac{T_{Aopt_2}}{1.05 \cdot T_{Aopt,max}} \\ \frac{b_{a3}}{1.05 \cdot b_{a,max}} & \frac{S_{a3}}{1.05 \cdot S_{a,max}} & \frac{MTOW_3}{1.05 \cdot MTOW_{max}} & \frac{T_{Aopt_3}}{1.05 \cdot T_{Aopt,max}} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{b_{aN_{LHS}}}{1.05 \cdot b_{a,max}} & \frac{S_{aN_{LHS}}}{1.05 \cdot S_{a,max}} & \frac{MTOW_{N_{LHS}}}{1.05 \cdot MTOW_{max}} & \frac{T_{Aopt_{N_{LHS}}}}{1.05 \cdot T_{Aopt,max}} \end{pmatrix} \tag{8}$$

A similar scaling operation was performed to obtain $\boldsymbol{P_S}$. Here, each time step value of every performance parameter was divided by 1.05 multiplied by the maximum value of that performance parameter within $\boldsymbol{P}$. For example, in order to scale L/D, each time step value of L/D in every row was divided by 1.05 multiplied by the maximum value of L/D that appears in $\boldsymbol{P}$. This operation was repeated for all 8 performance parameters in each row of $\boldsymbol{P}$ to construct $\boldsymbol{P_S}$ and is expressed in Equation (9). At this point, the databases for neural network training and testing have now been generated.

$$\boldsymbol{P_S} = \begin{pmatrix} \frac{\overrightarrow{C_{Dc1}}}{1.05 \cdot C_{Dc,max}} & \frac{\overrightarrow{C_{Di1}}}{1.05 \cdot C_{Di,max}} & \frac{\overrightarrow{C_{Dm1}}}{1.05 \cdot C_{Dm,max}} & \cdots & \frac{\overrightarrow{T_{R1}}}{1.05 \cdot T_{R,max}} \\ \frac{\overrightarrow{C_{Dc2}}}{1.05 \cdot C_{Dc,max}} & \frac{\overrightarrow{C_{Di2}}}{1.05 \cdot C_{Di,max}} & \frac{\overrightarrow{C_{Dm2}}}{1.05 \cdot C_{Dm,max}} & \cdots & \frac{\overrightarrow{T_{R2}}}{1.05 \cdot T_{R,max}} \\ \frac{\overrightarrow{C_{Dc3}}}{1.05 \cdot C_{Dc,max}} & \frac{\overrightarrow{C_{Di3}}}{1.05 \cdot C_{Di,max}} & \frac{\overrightarrow{C_{Dm3}}}{1.05 \cdot C_{Dm,max}} & \cdots & \frac{\overrightarrow{T_{R3}}}{1.05 \cdot T_{R,max}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\overrightarrow{C_{Dc\,N_{LHS}}}}{1.05 \cdot C_{Dc,max}} & \frac{\overrightarrow{C_{Di\,N_{LHS}}}}{1.05 \cdot C_{Di,max}} & \frac{\overrightarrow{C_{Dm\,N_{LHS}}}}{1.05 \cdot C_{Dm,max}} & \cdots & \frac{\overrightarrow{T_{R\,N_{LHS}}}}{1.05 \cdot T_{R,max}} \end{pmatrix} \tag{9}$$

Outside of neural network training, these matrices can be used to illustrate the numerical sensitivity of a performance parameter to a change in a configuration design parameter, and vice versa. Figure 9 highlights the L/D values across an entire mission for two BWB configurations. The blue line represents the L/D values for a BWB with $b_a = 221.6$ ft. $S_a = 8998$ ft^2, $MTOW$ = 408,300 lbs., and $T_{A_{opt}}$ = 116,000 lbs., while the orange line represents another BWB configuration with $b_a = 229.3$ ft., $S_a = 8048$ ft^2, $MTOW$ = 405,000 lbs., and $T_{A_{opt}}$ = 111,400 lbs.

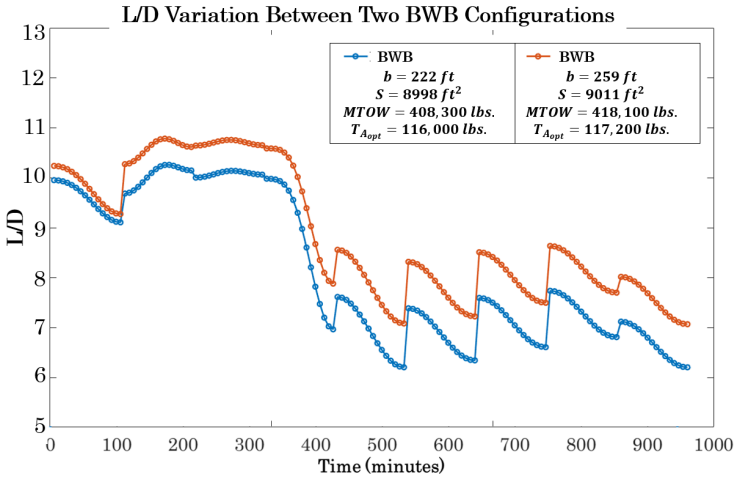

Figure 9. L/D variation across two generated BWB configurations.

2.2. Neural Network Architecture Generation, Numerical Method Selection, and Training

Neural network models, in their most basic form, are governed by a set of algorithms and control variables that map, or correlate, an input to a target. They are composed of multiple layers—input, hidden, and output—each with nodes. Nodes from one layer are linked to nodes from other layers, for example, a node in the first hidden layer is connected to all nodes in the input layer. Each linkage has its own weight value while layers themselves have their own bias values. These values help inform how data are translated, i.e., changed, as they pass from one layer to the next. For example, the input, $net_{h_{1_1}}$, into the first node of the first hidden layer, h_{1_1}, is a summation of all of the outputs from each input node multiplied by the weight value of each nodal link to that node in h_1, plus the bias value associated with the hidden layer. Throughout training, these weight values are adjusted such that the difference between the predicted values and the actual target values are within a specified tolerance.

For this study, the authors investigated shallow ANNs of two different types and of varied architectures. Shallow ANNs are characterized by being composed of only one hidden layer. Typically, shallow neural networks are used to better understand the feasibility of deploying machine learning models for a problem of interest. They are relatively simple, inexpensive models to develop, in terms of computational resources required, and often serve as better surrogate models than other comparatively more complex neural network architectures for problems not involving image or video processing. Theoretically, a shallow ANN with enough neurons in the hidden layer can adequately capture complex features of a database, such as non-linearity and multi-dimensionality [58]. Within the subset of shallow ANNs, two were selected for this study—feed-forward and cascade-forward neural networks. A feed-forward neural network is considered one of the simplest neural network architectures in that each neuron in the input layer is mapped, or linked, to every neuron in the hidden, which is then mapped to every neuron in the output layer. Compared to this architecture, a cascade-forwards neural network has an additional mapping from each neuron in the input layer directly to every neuron in the output layer. This type of ANN has provided favorable predictive accuracy in scenarios involving time-series data [9]. Figure 10 illustrates the architectures for both shallow feed-forward and shallow cascade-forward ANNs. Within each model type, the number of neurons in the hidden layer, μ, was varied from 1 to 10 to better understand predictive accuracy sensitivity as it relates to the number of neurons. Increasing the number of neurons can impact training time and computational cost—central processing unit (CPU) speed and random access memory (RAM) usage. The trade-off, however, is the potential for higher predictive accuracy.

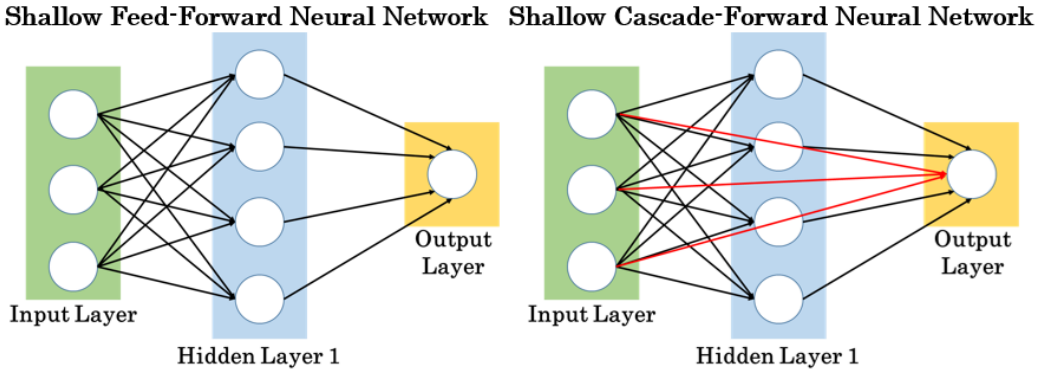

Figure 10. Feed-forward and cascade-forward neural network architectures.

With regard to the input and output layers, the number of neurons in each is dependent on the dimensionality of the training database. In this case, since the goal is to develop a neural network suitable for inverse aircraft design, the input for training is BWB performance data, namely $\boldsymbol{P_S}$, while the output is corresponding BWB configuration data, i.e., $\boldsymbol{D_S}$. The dimensionality of $\boldsymbol{P_S}$ is 30,000 × 1152, which implies that each BWB configuration is characterized by 1152 vehicle performance data points—composed of 8 individual performance parameters. For this reason, the input layer is composed of 1152 neurons, where each neuron represents an individual performance data point. For example, the second neuron in the input layer represents the second time step for C_{D_c}. Meanwhile, since the dimensionality of $\boldsymbol{D_S}$ is 30,000 × 4, this dictates that the output layer contains 4 neurons, where each one represents a different performance parameter. For example, the first neuron in the output layer represents b_a for a BWB configuration.

After neural network models have been architected, training can commence. The training scheme employed was supervised learning as it involved training data, $\boldsymbol{P_S}$ and $\boldsymbol{D_S}$, that is labeled, and each input corresponds to one specific set of outputs [59]. The intent of training is to allow the neural network to "learn" features of the data such that it can predict output values when provided inputs. In the process of doing so, weight values are numerically adjusted to reduce the error between predicted and target values. Figure 11 provides an overview of this process. This involves passing input and output data in its entirety, iteratively through the neural network. This is referred to as an epoch—one complete pass of data to be learned by a neural network.

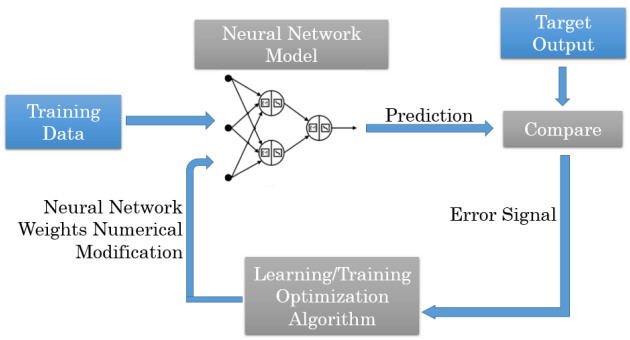

Figure 11. Neural network training process overview.

The size of the epoch is dictated by how many training points from $\boldsymbol{P_S}$ and $\boldsymbol{D_S}$ are used for training, which will be explained in the following section. Neural network training typically involves multiple epochs before a favorable predictive performance is converged on. For example, Equation (10) represents the calculation of $net_{h_{1_1}}$ for the first BWB configuration used as training in an epoch, namely $\boldsymbol{P_S}^1$ and $\boldsymbol{D_S}^1$. Here $\boldsymbol{P_S}_n^1$ represents the value of BWB airplane performance parameter in the first row and nth column of the $\boldsymbol{P_S}$ matrix. These values are associated with neurons in the input layer where $\boldsymbol{P_S}_1^1$, which

is the first time step value of C_{D_c}, represents the value of the first neuron while $\boldsymbol{P}_{S_{1152}}^{1}$, the last time step value of T_R, represents the value of the last neuron. $I_{w_n}^{1}$ are the weight values associated with each neuron linkage to h_1 and b_h is the bias value for the entire hidden layer.

$$net_{h1} = \left[\sum_{n=1}^{n=1152} \left(I_{w_n}^{1} \{P_{Scaled}\}_{n}^{1} \right) \right] + b_h \tag{10}$$

How data are passed through each neuron in the hidden layer, whether it is 1 neuron or 10, is governed by a numerical algorithm known as an activation function. Fundamentally, an activation function uses the input value of a neuron to calculate the output value of the neuron. There are many activation functions to choose from, each with its own trade-offs in accuracy, generalization across a wider training set, and computational wall-clock time needed for training, to name a few. For this study, the authors elected to use the tan-sigmoid function, also known as the hyperbolic tangent function, expressed in Equation (11), where out_{h_i} is the output from the ith neuron in the hidden layer. A benefit of the tan-sigmoid activation function is that its derivative is steeper at most points compared to the derivatives of other activation functions. This enables larger numerical changes in weight values during training, which can significantly reduce training time [60]. The training process will be described in more detail in the following paragraphs.

$$out_{hi} = \frac{2}{1 + e^{-2net_{hi}}} - 1 = \tanh(net_{hi}) \tag{11}$$

Ultimately this value, in conjunction with the weights associated with neuron linkages from the hidden layer neurons to the output layer neurons, O_{w_n} and the bias value corresponding to the output layer, b_o, is used to calculate the inputs into the output layer nodes, expressed in Equation (12), where net_{o_1} is the input into the first neuron of the output layer and μ is the number of neurons in the hidden layer. Figure 12 provides a schematic representation of a feed-forward shallow neural networking, highlighting some of the aforementioned variables.

$$net_{o_1} = \left[\sum_{n=1}^{n=\mu} \left(O_{w_n}^{1} out_{hin}^{1} \right) \right] + b_o \tag{12}$$

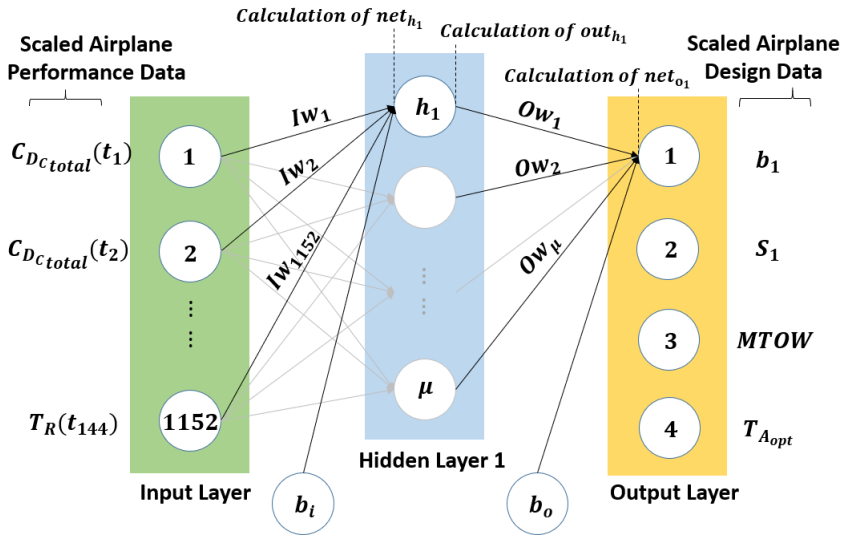

Figure 12. Feed-forward neural network calculations schematic.

Using $\boldsymbol{P}_S^1$, net_o is calculated for each neuron in the output layer. These values can be compared to target values that are $\boldsymbol{D}_S^1$. There will inherently be a difference between the two quantities, which is defined as the total error. Improving neural network predictive

accuracy is based on determining the sensitivity of the total error with respect to each weight value through calculation of the partial derivative of the total error with respect to each weight. This procedure is called backpropagation and involves tracing back outputs of the model through different neurons that were involved in generating that output and ultimately back to the weights that were applied. For this reason, the derivative of the activation function plays a crucial role in training. Once partial derivatives have been calculated, optimization algorithms are leveraged to determine numerical changes to weight values with the objective of minimizing total error [61]. Based on previous work, the authors elected to use the Levenburg–Marquardt (LM) optimization algorithm with Bayesian regularization. One benefit of the application of Bayesian regularization to the LM scheme is that it improves overall predictive performance through adjustments of a linear combination of squared errors and weight values [62,63].

While there are several ways to express neural network performance and training stoppage criteria, root mean squared error (RMSE) is often used as it can express accuracy across a wider set of design points. Compared to net error (NE), which is simply the absolute difference between the target and predicted value, RMSE expresses the standard deviation of the predicted errors across N_S sample points, shown in Equation (13), where y and y' are the prediction and truth values respectively. In this sense, a low RMSE value is indicative of low numerical noise as it pertains to predictive accuracy [64]. For training purposes, once a convergence in averaged RMSE was exhibited over 3 epochs, i.e., no improvement in predictive performance [65], training was stopped.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N_s}(y - y')^2}{N_s}} \tag{13}$$

2.3. K-Fold Cross-Validation Training and Testing Scheme

Neural network training and testing were conducted using a K-fold cross-validation scheme. Cross-validation is a statistical method that partitions data into subsets to then subsequently train a model using data from a subset and using the other subset for evaluation of the model's performance [66]. Typically, to reduce variability, multiple rounds of cross-validation are performed using different subsets from the same database. From these rounds, validation results are combined to yield an estimate of the model's overall predictive performance. Essentially, as it applies to neural network training and testing, cross-validation is a means to minimize prediction error through training and testing of all data points in a large dataset [67].

K-fold cross-validation involves randomly dividing a database into k subsets, or folds, each of the same size, i.e., the same number of data points. For the first batch, the first $k-1$ folds are used for training while the kth fold is used for testing. This process is repeated k times each time using a different fold for testing—for example, the kth batch would use the first fold for testing while the rest of the folds, folds two through k, are used for training [68]. Effectively, the process is repeated k times, with the model getting trained and testing across all folds, so the RMSE for cross-validation is computed by taking the average of RMSE values across all k folds.

For this study, a 10-fold cross-validation scheme is used. This scheme offers a key advantage in that every data point is used for training nine times and testing once. This in turn reduces bias and reduces the variance in the prediction errors [69]. Its obvious disadvantage is the need to train and test a model ten times, which can be computationally intensive.

$\boldsymbol{P_S}$ and $\boldsymbol{D_S}$ were divided into 10 folds, where each fold contained 3000 data points. Since each BWB configuration and performance metrics were generated using an LHS scheme, the folds were simply created by dividing $\boldsymbol{P_S}$ and $\boldsymbol{D_S}$ uniformly into 10 folds, as opposed to having to randomly select points to form a fold. Effectively, neural network models were trained with 9 folds (27,000 data points) and tested with 1 fold (3000 data points) for a given batch, which resulted from the ordering of the training and testing folds, as shown in Figure 13.

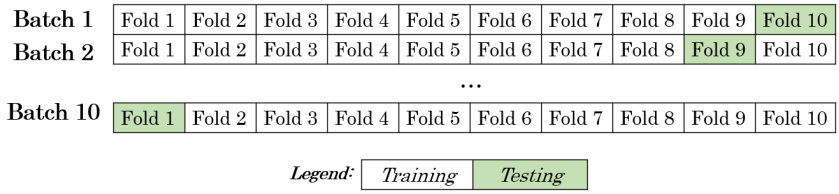

Figure 13. Ten-fold cross-validation training and testing matrix split by batches and fold numbers.

This scheme was applied to both neural network types, namely feed-forward and cascade-forward, and for all architectures, i.e., 1 to 10 neurons in the hidden layer. Therefore, for each neural network type, 100 neural network models were trained and tested. As mentioned in the previous subsection, the averaged RMSE was then calculated for each of these neural network models.

2.4. Random Forest Approach for Extensibility Analysis

Once 10-fold cross-validation neural network training and testing is complete, a neural network with the best predictive performance, across all architectures, can be identified. While the neural network models are trained and tested across parametrically-generated configurations bounded by the limits informed by Table 1, they have not been exposed to the exact configurations listed in this table. Additionally, since the geometry parametrization model was calibrated around the BWB450 vehicle, the configurations generated for training and testing purposes have similar configuration characteristics as the BWB450.

For this reason, the authors wanted to better understand the model's extensibility characteristics, i.e., predicting the value of configuration design parameters, when the model is supplied with BWB performance data for a vehicle not resembling the BWB450. This testing was accomplished through a random forest classification (RFC) scheme. At its core, RFC involves using a collection of surrogate models, working together, to classify or solve a problem [70]. Rather than relying on one model, the RFC scheme leverages several models, where each model is trained on their subspace but, when combined, can show a monotonic improvement in classification [71].

A result of the 10-fold cross-validation method is that even after a suitable neural network architecture is identified, for both neural network types, there are still 10 versions of each model, making it suitable for RFC. Leveraging this method, the extensibility analysis was conducted as follows: First, a BWB vehicle was selected from Table 1. This vehicle's configuration was modeled using SUAVE, and its performance was obtained using SUAVE's mission solver coupled with AVL. Both the configuration design variables—b, S, $MTOW$, and T_A—and performance parameters—C_{D_c}, C_{D_i}, C_{D_m}, C_{D_p}, L/D, W, SFC, and T_R—were then scaled using the same reference scaling quantities applied to generate the $\boldsymbol{P_S}$ and $\boldsymbol{D_S}$ matrices. Next, the scaled performance parameters were fed as inputs to all 10 neural networks of the same type and architecture, where each model subsequently generated values for configuration design variables. Lastly, the average was calculated for each of the four configuration design variables. Once un-scaled using the reference scaling quantities, these averages were compared against the target values, i.e., the actual values of b, S, $MTOW$, and T_A for the selected BWB.

3. Inverse-Design Prediction Results

3.1. Performance of Neural Networks

In general, the averaged RMSE value associated with neural network prediction is an indication of the model's ability to accurately predict BWB configuration design characteristics when provided with performance data. Note that the RMSE is averaged since the output of the neural network is four BWB design variables, and this average calculation is performed after rescaling to absolute values. A lower averaged RMSE value exhibits precise correlation and accurate predictive performance while a higher averaged RMSE value indicates poor classification. Figures 14 and 15 present the averaged RMSE for

all 10 folds, different numbers of neurons in the hidden layer ($1 \leq \mu \leq 10$), and for both neural network types—feed-forward and cascade-forward.

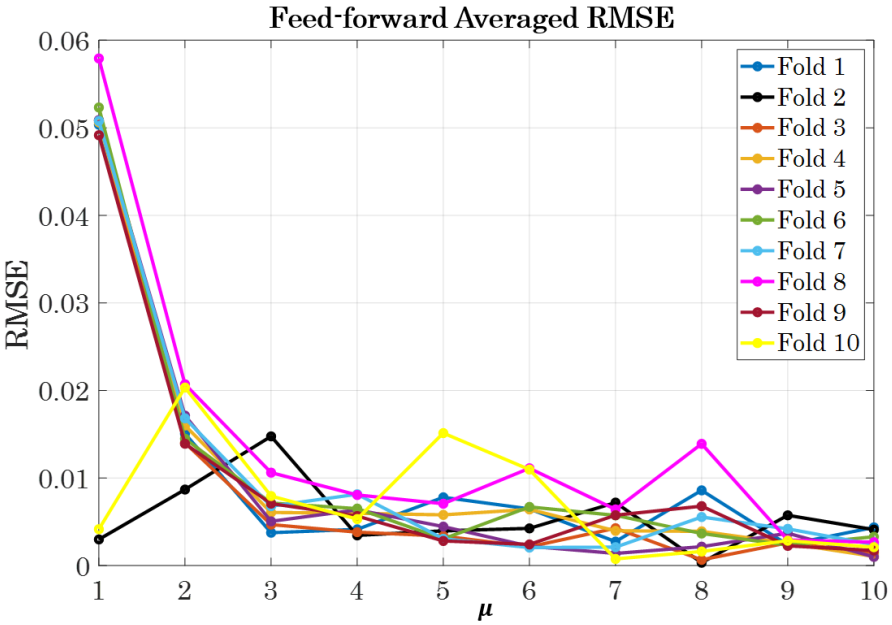

Figure 14. Feed-forward averaged RMSE for all 10 folds.

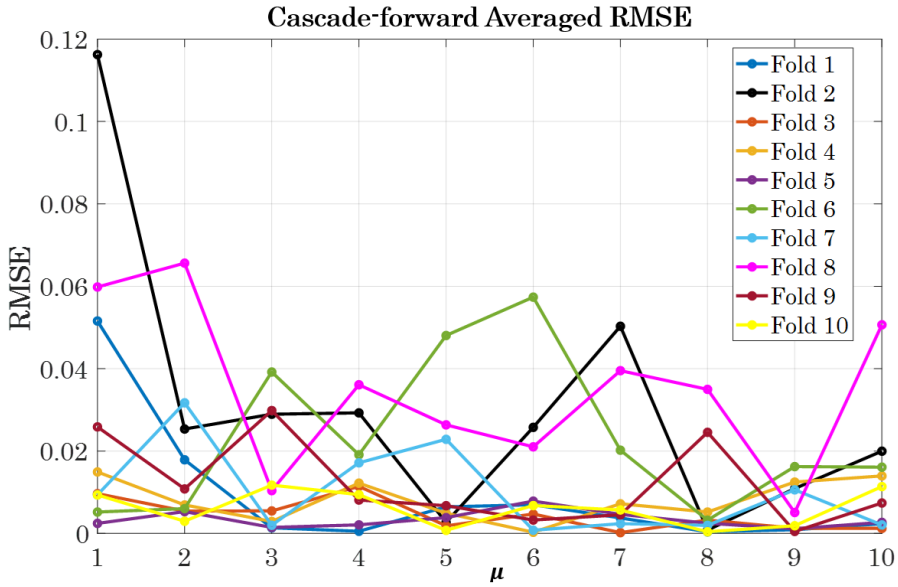

Figure 15. Cascade-forward averaged RMSE for all 10 folds.

To better aid in determining a suitable neural network architecture for extensibility analysis, elaborated on in Section 2.4 it was first important to determine prediction accuracy sensitivity to μ. For this reason, the fold-averaged RMSE was calculated across all 10 folds, for each μ value and for both neural network types, as shown in Figure 16.

The feed-forward neural network architectures exhibit a range in fold-averaged RMSE values from approximately 0.042 to 0.0025 while the cascade-forward neural network architectures range from approximately 0.032 to 0.006. For both neural network types, an increase in μ yielded a decrease in prediction errors, as is indicated by the decrease in averaged RMSE values. This is evidence of the benefit that additional neurons in the hidden layer can provide, which is additional hidden layer processing in a scenario with highly complex, non-linear data mapping. Note that these results are obtained for N_{LHS} = 30,000, and it is likely that different results can be obtained for a different number of training points.

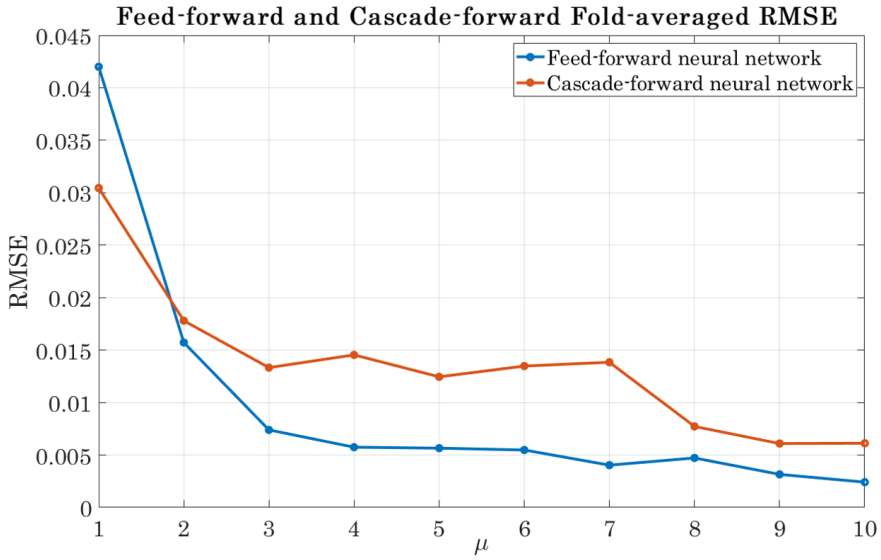

Figure 16. Feed-forward and cascade-forward fold-averaged RMSE.

When a neural network has too few neurons it lacks the ability to learn enough about the underlying patterns of the training data. Increasing the number of neurons allows the model to adequately learn more features and trends that may exist in the data. However, increasing μ beyond a certain value no longer has a large effect on predictive performance, as is indicated by the asymptotic convergence after $\mu = 4$ for feed-forward architectures, and after $\mu = 8$ for cascade-forward neural networks.

In general, feed-forward architectures yielded better predictive performance compared to cascade-forward architectures across almost all values of μ. Futhermore, feed-forward architectures also exhibited a significantly lower variance across all μ values, as shown by the green shaded region in Figure 17. On the other hand, cascade-forward neural networks are sensitive to overfitting when trained with too many training samples [72]—recall that each model is being trained with 27,000 training samples (9 folds for training, each with 3000 data points). Low error rates and high variance in predicted outputs are two indicators of overfitting, which is exhibited by cascade-forward architectures and is illustrated by the larger green shaded region in Figure 18 [73].

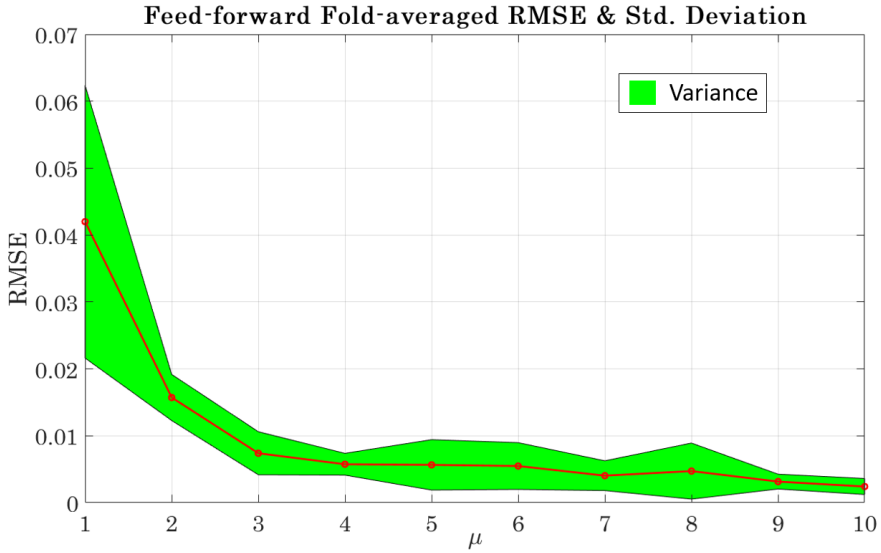

Figure 17. Feed-forward fold-averaged RMSE with a standard deviation overlay.

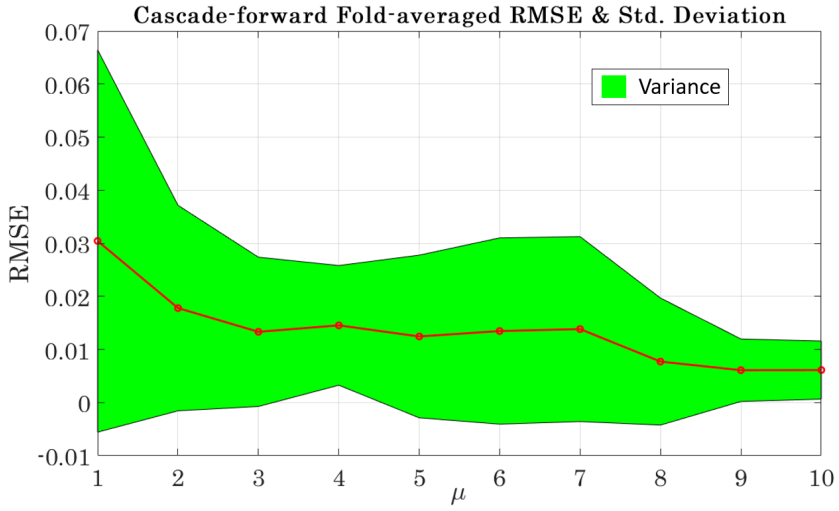

Figure 18. Cascade-forward fold-averaged RMSE with a standard deviation overlay.

3.2. Extensibility Analysis via Random Forest

Examining Figure 16 more closely, a shallow feed-forward neural network with $\mu = 10$ and a shallow cascade-forward neural network with $\mu = 10$ was chosen for the extensibility analysis via the random forest approach—described in Section 2.4 Although convergence in predictive accuracy was observed after $\mu = 4$ for feed-forward models and $\mu = 8$ for cascade-forward models, the most accurate models were chosen, which was at $\mu = 10$ for both neural network types. Across all 10 folds, these models exhibit the best predictive performance, i.e., yielding the lowest RMSE values.

Table 2 details all of the BWB vehicles that were part of the extensibility analysis. This table details the actual values of b, S, $MTOW$, and T_A for each BWB configuration, random forest averaged predicted values of b, S, $MTOW$, and T_A for both feed-forward and cascade-forward neural network architectures, and their respective errors as a percentage of the actual values. For feed-forward neural networks with $\mu = 10$, the error in b ranged from 1.6×10^{-5}% to 0.06%, with the average across all vehicles being 0.011%. Similarly, for S, the errors ranged from 2×10^{-4}% to 0.084%, with an average of 0.04%; $MTOW$—0.005% to 0.51%, with an average of 0.15%; and T_A—0.005% to 0.42%, with an average error of 0.11%. The errors exhibited by cascade-forward neural networks with $\mu = 10$ are comparatively much higher: b—0.45% to 3.61%, with an average of 0.96%; S—0.06% to 3.49%, with an average of 0.73%; $MTOW$—5.95×10^{-4}% to 3.35%, with an average of 1.89%; and T_A—0.013% to 5.92%, with an average error of 2.13%.

Table 2. Extensibility performance with $\mu = 10$ FF (feed-forward) and CF (cascade-forward) neural networks across BWB vehicle survey.

Design Variable	Actual Value	FF Predicted Value	FF % Error	CF Predicted Value	CF % Error
N + 3 SUGAR-Ray [41]					
b, ft.	168.5	168.5	0.004	167.9	3.61
S, ft^2	4136	4139.5	0.084	3991.7	3.49
$MTOW$, lbs.	181,500	181,446	0.03	187,580	3.35
T_A, lbs.	56,000	55,980	0.03	57,719	3.07
HWB216-GTF [42]					
b, ft.	220	220.01	0.007	218.7	0.61
S, ft^2	8221	8221.2	0.002	8193.9	0.33
$MTOW$, lbs.	312,500	314,094	0.51	320,000	2.40
T_A, lbs.	92,000	92,005	0.005	95,404	0.37

Table 2. *Cont.*

Design Variable	Actual Value	FF Predicted Value	FF % Error	CF Predicted Value	CF % Error
			SAX40 [43]		
b, ft.	221.6	221.6	0.004	220.4	0.54
S, ft^2	8997.6	8997.03	0.006	8987.7	0.11
$MTOW$, lbs.	330,300	330,283	0.005	338,788	2.57
T_A, lbs.	92,500	92,505	0.005	95,414	3.15
			ERA-009A [44]		
b, ft.	229.3	229.25	0.02	228.2	0.47
S, ft^2	8048.0	8047.9	2×10^{-4}	8018.2	0.37
$MTOW$, lbs.	411,250	411,003	0.06	424,862	3.31
T_A, lbs.	95,000	94,905	0.10	100,624	5.92
			HWB301-GTF [42]		
b, ft.	250	250.0	0.01	248.5	0.59
S, ft^2	10,169	10,168.8	0.002	10,182.1	0.13
$MTOW$, lbs.	533,000	533,213	0.04	545,419	2.33
T_A, lbs.	134,500	134,729	0.17	137,432	2.18
			HWB400-GTF [42]		
b, ft.	260	260.0	0.007	258.3	0.66
S, ft^2	11,471	11,467.0	0.03	11,494.0	0.2
$MTOW$, lbs.	701,000	701,561	0.08	713,197	1.74
T_A, lbs.	168,500	168,534	0.02	171,247	1.63
			ACFA-2020 [45]		
b, ft.	261.9	261.9	1.6×10^{-5}	260.7	0.45
S, ft^2	14,290.6	14,296.4	0.04	14,299.2	0.06
$MTOW$, lbs.	884,000	883,885	0.013	902,918	2.14
T_A, lbs.	239,000	238,785	0.09	248,775	4.09
			BW-98 [46]		
b, ft.	254.3	254.4	0.06	258.6	1.7
S, ft^2	14,968.2	14,991.7	0.16	15,200.8	1.55
$MTOW$, lbs.	1,060,000	1,064,770	0.45	1,088,832	2.72
T_A, lbs.	296,500	297,449	0.32	301,274	1.61
			IWB-750 [47]		
b, ft.	328.1	328.1	0.002	325.8	0.70
S, ft^2	17,093.1	17,089.2	0.023	17,063.4	0.17
$MTOW$, lbs.	1,262,000	1,264,145	0.17	1,262,038	0.003
T_A, lbs.	353,000	354,483	0.42	355,259	0.64
			NACRE-750 [47]		
b, ft.	328.1	328.1	0.005	325.7	0.71
S, ft^2	21,452.5	21,448.9	0.02	21,269.3	0.85
$MTOW$, lbs.	1,390,000	1,393,892	0.28	1,393,336	0.24
T_A, lbs.	468,500	468,641	0.03	472,014	0.75
			VELA-3 [49]		
b, ft.	326.8	326.8	4×10^{-4}	325.0	0.55
S, ft^2	22,087.5	22,099.3	0.05	21,907.8	0.81
$MTOW$, lbs.	1,542,000	1,542,463	0.03	1,542,009	5.95×10^{-4}
T_A, lbs.	432,500	432,760	0.06	432,556	0.013

Practically, for every BWB configuration, the feed-forward neural network was significantly more accurate than the cascade-forward architecture in all 4 design variables. Applying the feed-forward architecture to a broader set of vehicles—BWBs, HWBs, and

IWBs—confirms that the feed-forward neural networks were not only able to adequately learn features and trends that exist between b, S, $MTOW$, and T_A and the vehicle's performance data, but also be able to generalize effectively across a broader design space. Additionally, vehicles that exhibit low prediction error are closer in resemblance to the training data the neural networks were exposed to. In this sense, those vehicles with lower errors likely exhibit similar configuration design characteristics as the BWB450. Furthermore, the variability of the results for the cascade-forward neural networks, as exhibited by the standard deviation across each μ value, also confirms that such an architecture did indeed struggle with overfitting during the training process. As such, given the amount of training data points this neural network type was exposed to, it is not suitable for usage in a broader, more diverse design space. Furthermore, the extensibility analysis shows an increase in error as input points begin to fall outside of the range of data exposed to the model during training. This results in values greater than 1 after scaling, which does influence how such a model can be used. Careful consideration must be taken when using it to extrapolate outside of the design space it was trained with.

3.3. Envisioned Usage in Design Space Exploration

The results of this study have showcased the viability of developing machine learning models, namely neural networks, capable of generating BWB configuration design parameters when provided with time-series, mission-informed BWB performance data. The data used to train and test these neural networks were generated via Level-0 airplane configuration definition and performance estimation tools, which are typically leveraged in the preliminary design cycle phase. Traditionally, this phase also involves design space exploration where conventional techniques dictate iteratively exploring the design space while converging to an optimum configuration with desired performance. Here, the aforementioned neural networks can be advantageous over traditional surrogate models in that the neural network is trained to handle performance specified over the entire mission of the airplane. In this sense, a designer can individually modify performance parameters, such as L/D, in specific portions of the flight envelope, such as climb, while leaving L/D values in the cruise segment constant, and instantly observe its effect on the configuration design via the prediction of design parameters from the neural network. Additionally, in the context of optimization within design space exploration, such a model could be advantageous in identifying a suitable starting point—baseline airplane to begin optimizing. This, in turn, can reduce the number of iterations, thereby decreasing computational costs.

Additionally, the developed neural networks could be employed in uncertainty quantification (UQ) scenarios where changes to performance parameters can be viewed through the lens of configuration design changes. UQ problems typically leverage analytical tools of varied fidelity—dependent on the nature of the problem. Conducting airplane performance analysis across an entire flight profile using traditional assessment tools can be computationally intensive. In this sense, leveraging the mission-informed neural network models could allow designers to more rapidly explore the sensitivity of the design space. Furthermore, exposing the model to a larger set of design parameters can aid in understanding more about tightly coupled trends and relationships that may exist between configuration design and performance parameters.

Since the neural networks were trained with data from a BWB configuration parameterization model, seemingly, to adopt these neural networks for a different vehicle class would simply require replacing the configuration parameterization model. For example, a TAW configuration parameterization model could be leveraged for the configuration generation process. This broadly implies that the computational framework can easily be adopted to any vehicle by simply replacing the parameterization model.

4. Conclusions and Future Work

This paper has presented the development of neural networks, the data generated to train them, the prediction accuracy of these models, and extensibility analysis for BWB

inverse-design space exploration to investigate the feasibility of the design approach for unconventional aircraft configurations. The models were developed to handle time-dependent, mission-informed BWB airplane performance data—C_{D_c}, C_{D_i}, C_{D_m}, C_{D_p}, L/D, W, SFC, and T_R—as inputs and generate BWB configuration data—b, S, $MTOW$, and T_A—as outputs. The neural networks were trained with BWB configuration data generated through a BWB configuration parameterization model, which is calibrated using the BWB450 vehicle. These configurations were then run through SUAVE, a low-fidelity airplane performance assessment tool, to obtain BWB performance data, which was also used for neural network training. Two shallow neural network architectures were tested, namely feed-forward and cascade-forward. Within each type, the number of neurons in the hidden layer, μ, was varied from 1 to 10. Each model leveraged the tan-sigmoid activation function, and the Levenburg–Marquardt optimization algorithm with Bayesian regularization was selected to adjust weight quantities during the training process. A total of 30,000 data points were generated, i.e., 30,000 BWB configurations were generated, each with their own unique performance values. Training and testing of each neural network was conducted using a 10-fold cross-validation scheme, where each fold contained 3000 BWB configuration-performance pairs. By doing so, an optimal neural network type and architecture was identified. The number of data points chosen, i.e., 30,000, was informed by providing adequate design space coverage and a suitable number of training points for each fold; however, it was not the focus of this study.

The results obtained proved the feasibility of developing inverse-design, mission-informed neural networks specifically for an unconventional airplane configuration, namely the BWB. While individual models could have been developed for each one of the four design variables, the intent was to investigate the viability of developing only one neural network for the prediction of all four configuration parameters, which was successfully demonstrated. Neural network type and architecture both had a significant influence on prediction accuracy. Ultimately, the 10-fold cross-validation training and testing scheme helped reveal that a feed-forward neural network with $\mu = 10$ exhibited the highest predictive accuracy for all neural networks developed. While not nearly as accurate as its feed-forward counterpart, the cascade-forward neural networks, also with $\mu = 10$, had the highest predictive accuracy among the cascade-forward architectures. It is worth noting, however, that prediction accuracy convergence did not significantly decrease after $\mu = 4$ for feed-forward architectures and $\mu = 8$ for cascade-forward architectures. Considering the minimization of computational costs, it may be more suitable to leverage these models instead.

After training, These models were chosen for extensibility analysis via a random forest scheme—rather than using only one neural network for the prediction of configuration values, a collection of neural networks, of the same architecture and type, can cohesively express predicted configuration values. Specifically, all 10 feed-forward neural networks with $\mu = 10$, from each of the 10 folds were used to construct an averaged prediction of configuration values when provided with a set of test BWB vehicles. A similar scheme was employed for all 10 cascade-forward neural networks with $\mu = 10$.

The feed-forward neural network architecture demonstrated significantly better prediction accuracies than the cascade-forward architecture and with substantially lower variance. This indicates the cascade-forward architectures suffered from a degree of overfitting while the feed-forward architecture exhibited better performance in terms of prediction accuracy in a broader design space. Across the BWB test vehicles, relatively larger errors were seen for vehicles that have drastically different configuration characteristics and trends, such as distribution of wing area along the span, than the vehicle used to calibrate the models, namely the BWB450.

The developed framework presented in this paper was geared towards unconventional configurations, and the neural networks were exposed to data derived from low-fidelity, Level-0 configuration and performance assessment tools. A possible avenue to explore, which may lead to a reduction in some of the errors ultimately seen in the extensibility

analysis, is the exposure to mixed data types and specifically mixed-fidelity. Leveraging a higher-fidelity tool for performance estimation in conjunction with a lower-order tool warrants further investigation of its effect on overall prediction accuracies. This could also prove to be a key step in expanding the number of performance and design parameters the neural network models can be trained and tested with, which could, in turn, permit the exploration of larger and more multi-dimensional design spaces. It is also worth investigating the effect of incorporating static airplane performance data into the training dataset, for example, performance parameters such as approach speed, V_{app}, or takeoff field length, $TOFL$.

A complication of inverse-design methods is that they are often ill-posed, i.e., one output can be mapped to more than one input. In this study, the authors have mitigated these risks by carefully constraining the training data, leveraging a configuration parameterization model, which is calibrated using the BWB450 vehicle, and not changing flight profiles across the dataset. The exploration of making the vehicle's flight profile dynamic rather than fixing it warrants further investigation and could ultimately lead to the usage of such models in airplane design optimization settings. More examination in determining which design parameters to vary would certainly be beneficial and could be achieved via an analysis of variance tests from a larger set of design parameters. These are all significant areas worthy of investigation and could help pave the road towards the development of robust mission-informed predictive surrogate models capable of being deployed in airplane inverse-design scenarios.

Author Contributions: Conceptualization, R.S.S. and S.H.; methodology, R.S.S. and S.H.; software, R.S.S. and S.H.; validation, R.S.S. and S.H.; formal analysis, R.S.S. and S.H.; investigation, R.S.S. and S.H.; resources, R.S.S. and S.H.; data curation, R.S.S. and S.H.; writing—original draft preparation, R.S.S. and S.H.; writing—review and editing, R.S.S. and S.H.; visualization, R.S.S. and S.H.; supervision, S.H.; project administration, R.S.S. and S.H.; funding acquisition, R.S.S. and S.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

Nomenclature

b	=	wingspan
S	=	wing area
AR	=	aspect ratio
W/L	=	wing loading
T/W	=	thrust-to-weight ratio
C_L	=	lift coefficient
C_D	=	drag coefficient
C_M	=	pitching moment coefficient
C_P	=	pressure coefficient
α	=	angle of attack
L/D	=	lift-to-drag ratio
V_v	=	tail volume coefficient
$\overrightarrow{b}$	=	wingspan distribution
$\overrightarrow{S}$	=	wing area distribution
$\overrightarrow{c}$	=	root chord length distribution
$\overrightarrow{b_{bl}}$	=	wingspan distribution for BWB450
$\overrightarrow{S_{bl}}$	=	wing area distribution for BWB450
$\overrightarrow{c_{\lambda_{bl}}}$	=	quarter chord sweep distribution for BWB450
c_r	=	root chord length
AC	=	aerodynamic center

ϕ	=	root chord fraction
η	=	wingspan fraction
w_ϕ	=	position of engine in terms of root chord fraction
w_η	=	position of engine in terms of wingspan fraction
$\overrightarrow{b_a}$	=	adjusted wingspan distribution
$\overrightarrow{S_a}$	=	adjusted wing area distribution
T_A	=	total static thrust available
T_{Aopt}	=	2nd-segment optimized total static thrust available
R^2	=	coefficient of determination
C_{D_c}	=	compressibility effects drag coefficient
C_{D_i}	=	induced drag coefficient
C_{D_m}	=	miscellaneous drag coefficient
C_{D_p}	=	parasitic drag coefficient
W	=	weight
T_R	=	thrust required
SFC	=	specific fuel consumption
$\overrightarrow{C_{D_c}}, \overrightarrow{C_{D_i}}, \overrightarrow{C_{D_m}}, \overrightarrow{C_{D_p}}, \overrightarrow{L/D}$ $\overrightarrow{W}, \overrightarrow{SFC}, \overrightarrow{T_R}$	=	time-domain airplane performance vectors
$\boldsymbol{D}$	=	configuration design data matrix
$\boldsymbol{P}$	=	airplane performance matrix
$MTOW_{max}, b_{a,max}, S_{a,max}, T_{Aopt,max}$ $C_{D\ c,max}, C_{D\ i,max}, C_{D\ m,max}, C_{D\ p,max}$ $L/D_{max}, W_{max}, SFC_{max}, T_{R,max}$	=	max reference quantitites used for scaling
$\boldsymbol{D_S}$	=	scaled configuration design data matrix
$\boldsymbol{P_S}$	=	scaled airplane performance matrix
N_{LHS}	=	Latin-ypercube sample size
net_{h_1}	=	net input into the first node of the first hidden layer
h_1	=	first hidden layer
μ	=	number of neurons in the hidden layer of a shallow neural network
Iw	=	weight values for input-to-hidden layer nodal connections
b_h	=	bias value of the hidden layer
out_{h_i}	=	output from the ith neuron in the hidden layer
Ow	=	weight values for hidden-to-output layer nodal connections
b_o	=	bias value of the output layer
V_{app}	=	approach speed

References

1. Dorsey, A.; Uranga, A. Design Space Exploration of Blended Wing Bodies. In Proceedings of the AIAA AVIATION 2021 Forum, Virtual Event, 2–6 August 2021. [CrossRef]
2. Raymer, D.P. Aircraft Design: A Conceptual Approach. In *AIAA Education Series, AIAA*; American Institute of Aeronautics and Astronautics, Inc.: Reston, VA, USA, 1992. [CrossRef]
3. Strathoff, P.; Zumegen, C.; Stumpf, E.; Klumpp, C.; Jeschke, P.; Warner, K.L.; Gelleschus, R.; Bocklisch, T.; Portner, B.; Moser, L.; et al. On the Design and Sustainability of Commuter Aircraft with Electrified Propulsion Systems. In Proceedings of the AIAA AVIATION 2022 Forum, Chicago, IL, USA, 27 June–1 July 2022. [CrossRef]
4. Wakayama, S.; Kroo, I. The challenge and promise of blended-wing-body optimization. In Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, MO, USA, 2–4 September 1998. [CrossRef]
5. Cinar, G.; Cai, Y.; Bendarkar, M.V.; Burrell, A.I.; Denney, R.K.; Mavris, D.N. System Analysis and Design Space Exploration of Regional Aircraft with Electrified Powertrains. *J. Aircr.* **2022**, *60*, 1–28. [CrossRef]
6. Biser, S.; Filipenko, M.; Boll, M.; Kastner, N. Design Space Exploration Study and Optimization of a Distributed Turbo-Electric Propulsion System for a Regional Passenger Aircraft. In Proceedings of the AIAA Propulsion and Energy 2020 Forum, Virtual Event, 24–28 August 2020. [CrossRef]
7. Kallou, E.; Sarojini, D.; Mavris, D.N. Application of Set-Based Design Principles on Multi-Level Aircraft Design Space Exploration. In Proceedings of the AIAA AVIATION 2022 Forum, Chicago, IL, USA, 27 June–1 July 2022. [CrossRef]

8. Wang, J.; Wu, J.; Ling, J.; Iaccarino, G.; Xiao, H. *Physics-Informed Machine Learning for Predictive Turbulence Modeling: Towards a Complete Framework*; Sandia National Lab.: Albuquerque, NM, USA, 2016. [CrossRef]
9. Sharma, R.S.; Hosder, S. Investigation of Mission-Driven Inverse Aircraft Design Space Exploration with Machine Learning. *J. Aerosp. Inf. Syst.* **2021**, *18*, 774–789. [CrossRef]
10. Sun, G.; Sun, Y.; Wang, S. Artificial neural network based inverse design: Airfoils and wings. *Aerosp. Sci. Technol.* **2015**, *42*, 415–428. [CrossRef]
11. Gibbs, J.; Gollnick, V. Inverse Aircraft Design. In Proceedings of the AIAA AVIATION 2020 FORUM, Virtual, 15–19 June 2020. [CrossRef]
12. Sekar, V.; Zhang, M.; Shu, C.; Khoo, B.C. Inverse Design of Airfoil Using a Deep Convolutional Neural Network. *AIAA J.* **2019**, *57*, 993–1003. [CrossRef]
13. Rai, M.M.; Madavan, N.K. Aerodynamic Design Using Neural Networks. *AIAA J.* **2000**, *38*, 173–182. [CrossRef]
14. Barrett, T.R.; Bressloff, N.W.; Keane, A.J. Airfoil Shape Design and Optimization Using Multifidelity Analysis and Embedded Inverse Design. *AIAA J.* **2006**, *44*, 2051–2060. [CrossRef]
15. Kharal, A.; Saleem, A. Neural networks based airfoil generation for a given C_p using Bezier–PARSEC parameterization. *Aerosp. Sci. Technol.* **2012**, *23*, 330–344. [CrossRef]
16. Yilmaz, E.; German, B. A Deep Learning Approach to an Airfoil Inverse Design Problem. In Proceedings of the 2018 Multidisciplinary Analysis and Optimization Conference, Atlanta, GA, USA, 25–29 June 2018. [CrossRef]
17. Glaws, A.; King, R.N.; Vijayakumar, G.; Ananthan, S. Invertible Neural Networks for Airfoil Design. *AIAA J.* **2022**, *60*, 3035–3047. [CrossRef]
18. Yu, K.; Chen, C.; Chen, Y. Inverse Design of Nozzle Using Convolutional Neural Network. *J. Spacecr. Rocket.* **2022**, *59*, 1161–1170. [CrossRef]
19. Oddiraju, M.; Behjat, A.; Nouh, M.; Chowdhury, S. Efficient Inverse Design of 2D Elastic Metamaterial Systems using Invertible Neural Networks. In Proceedings of the AIAA AVIATION 2021 FORUM, Virtual Event, 2–6 August 2021. [CrossRef]
20. Li, J.; Zhang, M. Data-based approach for wing shape design optimization. *Aerosp. Sci. Technol.* **2021**, *112*, 106639. [CrossRef]
21. Thuerey, N.; Weißenow, K.; Prantl, L.; Hu, X. Deep Learning Methods for Reynolds-Averaged Navier–Stokes Simulations of Airfoil Flows. *AIAA J.* **2020**, *58*, 25–36. [CrossRef]
22. Singh, A.P.; Medida, S.; Duraisamy, K. Machine-Learning-Augmented Predictive Modeling of Turbulent Separated Flows over Airfoils. *AIAA J.* **2017**, *55*, 2215–2227. [CrossRef]
23. Li, J.; Du, X.; Martins, J. Machine learning in aerodynamic shape optimization. *Prog. Aerosp. Sci.* **2022**, *134*, 100849. [CrossRef]
24. Li, J.; Zhang, M. Adjoint-Free Aerodynamic Shape Optimization of the Common Research Model Wing. *AIAA J.* **2021**, *59*, 1990–2000. [CrossRef]
25. Bouhlel, M.; He, S.; Martins, J. Calable gradient-enhanced artificial neural networks for airfoil shape design in the subsonic and transonic regime. *Struct. Multidiscip. Optim.* **2020**, *61*, 1363–1376. [CrossRef]
26. Du, X.; He, P.; Martins, J. A B-spline-based generative adversarial network model for fast interactive airfoil aerodynamic optimization. In Proceedings of the AIAA SciTech Forum, AIAA, Orlando, FL, USA, 6–10 January 2020. [CrossRef]
27. Barnhart, S.; Narayanan, B.; Gunasekaran, S. Blown wing aerodynamic coefficient predictions using traditional machine learning and data science approaches. In Proceedings of the AIAA SciTech Forum, Online, 11–15 January 2021. [CrossRef]
28. Karali, H.; Inalhan, G.; Demirezen, M.U.; Yukselen, M.A. A new nonlinear lifting line method for aerodynamic analysis and deep learning modeling of small unmanned aerial vehicles. *Int. J. Micro Air Veh.* **2021**, *13*, 1–24. [CrossRef]
29. Cai, S.; Wang, Z.; Fuest, F.; Jeon, Y.; Gray, C.; Karniadakis, G.E. Flow over an espresso cup: Inferring 3-d velocity and pressure fields from tomographic background oriented schlieren via physics-informed neural networks. *J. Fluid Mech.* **2021**, *915*, A102. [CrossRef]
30. Yilmaz, E.; German, B. Conditional generative adversarial network framework for airfoil inverse design. In Proceedings of the AIAA AVIATION Forum, Online, 15–19 June 2020. [CrossRef]
31. Achour, G.; Sung, W.J.; Pinon-Fischer, O.; Mavris, D. Development of a conditional generative adversarial network for airfoil shape optimization. In Proceedings of the AIAA Scitech 2020 Forum, Orlando, FL, USA, 6–10 January 2020. [CrossRef]
32. Secco, N.; de Mattos, B. Artificial neural networks to predict aerodynamic coefficients of transport airplanes. *Aerosp. Sci. Technol.* **2017**, *89*, 211–230. [CrossRef]
33. Lukaczyk, T.W.; Wendorff, A.D.; Colonno, M.; Economon, T.D.; Alonso, J.J.; Orra, T.H.; Ilario, C. SUAVE: An Open-Source Environment for Multi-Fidelity Conceptual Vehicle Design. In Proceedings of the 16th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Dallas, TX, USA, 22–26 June 2015. [CrossRef]
34. Botero, E.M.; Wendorff, A.; MacDonald, T.; Variyar, A.; Vegh, J.M.; Lukaczyk, T.W.; Alonso, J.J.; Orra, T.H.; Ilario, C. SUAVE: An Open-Source Environment for Conceptual Vehicle Design and Optimization. In Proceedings of the 54th AIAA Aerospace Sciences Meeting, San Diego, CA, USA, 4–8 January 2016. [CrossRef]
35. MacDonald, T.; Botero, E.; Vegh, J.M.; Variyar, A.; Alonso, J.J.; Orra, T.H.; Ilario, C. SUAVE: An Open-Source Environment Enabling Unconventional Vehicle Designs through Higher Fidelity. In Proceedings of the 55th AIAA Aerospace Sciences Meeting, Grapevine, TX, USA, 9–13 January 2017. [CrossRef]

36. MacDonald, T.; Clarke, M.; Botero, E.M.; Vegh, J.M.; Alonso, J.J. SUAVE: An Open-Source Environment Enabling Multi-Fidelity Vehicle Optimization. In Proceedings of the 18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Denver, CO, USA, 5–9 June 2017. [CrossRef]
37. Tanio, T.; Takeda, K.; Yu, J.; Hashimoto, M. Training Data Reduction using Support Vectors for Neural Networks. In Proceedings of the 2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), Lanzhou, China, 18–21 November 2019. [CrossRef]
38. Lyu, Z.; Martins, J.R.R.A. Aerodynamic Design Optimization Studies of a Blended-Wing-Body Aircraft. *J. Aircr.* **2014**, *51*, 1604–1617. [CrossRef]
39. Liebeck, R.H. Design of the Blended Wing Body Subsonic Transport. *J. Aircr.* **2004**, *41*, 10–25. [CrossRef]
40. Brown, M.; Vos, R. Conceptual Design and Evaluation of Blended-Wing Body Aircraft. In Proceedings of the 2018 AIAA Aerospace Sciences Meeting, Kissimmee, FL, USA, 8–12 January 2018. [CrossRef]
41. Nickol, C.; Haller, W. Assessment of the Performance Potential of Advanced Subsonic Transport Concepts for NASA's Environmentally Responsible Aviation Project. In Proceedings of the 54th AIAA Aerospace Sciences Meeting, San Diego, CA, USA, 4–8 January 2016. [CrossRef]
42. Hileman, J.; Spakovszky, Z.; Drela, M.; Sargeant, M.; Jones, A. Airframe Design for Silent Fuel-Efficient Aircraft. *AIAA J. Aircr.* **2010**, *47*, 956–969. [CrossRef]
43. Bonet, J.; Schellenger, H.; Rawdon, B.; Elmer, K.; Wakayama, S.; Brown, D. *Environmentally Responsible Aviation (ERA) Project—N + 2 Advanced Vehicle Concepts Study and Conceptual Design of Subscale Test Vehicle (STV) Final Report*; Report No.: NASA/CR-2011-216519; NASA Dryden Flight Research Center: Edwards, CA, USA, 2011.
44. Maier, R. ACFA 2020—An FP7 project on active control of flexible fuel efficient aircraft configurations. *Prog. Flight Dyn. Guid. Navig. Control. Fault Detect. Avion.* **2013**, *6*, 585–600. [CrossRef]
45. Smith, H. College of Aeronautics Blended Wing Body Development Programme, ICAS-2000-1.1.4. In Proceedings of the 22nd International Congress of the Aeronautical Sciences, Harrogate, UK, 27 August–1 September 2000.
46. Bolsunovsky, A.; Buzoverya, N.; Gurevich, B.; Denisov, V.; Dunaevsky, A.; Shkadov, L. Flying wing—Problems and Decisions. *AIAA J. Aircr. Des.* **2001**, *4*, 193–210. [CrossRef]
47. Godard, J. Semi-Buried Engine Installation: The Nacre Project Experience, ICAS-2010-4.4.3. In Proceedings of the 27th International Congress of the Aeronautical Sciences, Nice, France, 19–24 September 2010.
48. Frota, J.; Nicholls, K.; Whurr, J.; Müller, M.; Gall, P.E.; Loerke, J.; Macgregor, K.; Schmollgruber, P.; Russell, J.; Hepperle, M.; et al. *Final Activity Report. New Aircraft Concept Research (NACRE)*; Technical Report; SIXTH FRAMEWORK PROGRAMME PRIORITY 4, Aeronautics and Space, FP6-2003-AERO-1; NACRE Consortium: Blagnac, France, 2010.
49. Hepperle, M. The VELA Project. 2005. Available online: https://www.dlr.de/as/en/Portaldata/5/Resources/dokumente/projekte/vela/The_VELA_Project.pdf (accessed on 27 November 2023).
50. Fusaro, R.; Viola, N. Influence of High Level Requirements in Aircraft Design: From scratch to sketch. In Proceedings of the 2018 Aviation Technology, Integration, and Operations Conference, Atlanta, GA, USA, 25–29 June 2018. [CrossRef]
51. Chrisman, L. Latin Hypercube vs. Monte Carlo Sampling. March 2020. Available online: https://lumina.com/latin-hypercube-vs-monte-carlo-sampling/ (accessed on 4 January 2024).
52. Aistleitner, C.; Hofer, M.; Tichy, R. A Central Limit Theorem for Latin Hypercube Sampling with Dependence and Application to Exotic Basket Option Pricing. *Int. J. Theor. Appl. Financ.* **2012**, *15*, 1250046. [CrossRef]
53. Loh, W.L. On Latin Hypercube Sampling. *Ann. Stat.* **1996**, *24*, 2058–2080. [CrossRef]
54. Raymer, D.P. RDS^{win} : Seamlessly-Integrated Aircraft Conceptual Design for Students & Professionals. In Proceedings of the 54th AIAA Aerospace Sciences Meeting, San Diego, CA, USA, 4–8 January 2016. [CrossRef]
55. Beard, J.E.; Takahashi, T.T. Revisiting Takeoff Obstacle Clearance Procedures: An Argument for Extended Second Segment Climb. In Proceedings of the 17th AIAA Aviation Technology, Integration, and Operations Conference, Denver, CO, USA, 5–9 June 2017. [CrossRef]
56. Stöttner, T. Why Data Should Be Normalized before Training a Neural Network. May 2019. Available online: https://towardsdatascience.com/why-data-should-be-normalized-before-training-a-neural-network-c626b7f66c7d (accessed on 20 January 2023).
57. Nayak, S.; Misra, B.; Behera, H. Impact of Data Normalization on Stock Index Forecasting. *Int. J. Comput. Inf. Syst. Ind. Manag. Appl.* **2014**, *6*, 357–369.
58. Kim, D.E.; Gofman, M. Comparison of shallow and deep neural networks for network intrusion detection. In Proceedings of the 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 8–10 January 2018; pp. 204–208. [CrossRef]
59. Deng, L.; Li, X. Machine Learning Paradigms for Speech Recognition: An Overview. *IEEE Trans. Audio Speech Lang. Process.* **2013**, *21*, 1060–1089. [CrossRef]
60. Lin, C.W.; Wang, J.S. A digital circuit design of hyperbolic tangent sigmoid function for neural networks. In Proceedings of the 2008 IEEE International Symposium on Circuits and Systems, Seattle, WA, USA, 18–21 May 2008; pp. 856–859. [CrossRef]
61. Nicholson, C. A Beginner's Guide to Neural Networks and Deep Learning. Available online: https://wiki.pathmind.com/neural-network (accessed on 23 November 2020).

62. Gill, P.R.; Murray, W.; Wright, M.H. The Levenburg-Marquardt Method. In *Practical Optimization*; Emerald Group Publishing Limited: Bingley, UK, 1981; Volume 4, pp. 136–137.
63. Aburaed, N.; Atalla, S.; Mukhtar, H.; Al-Saad, M.; Mansoor, W. Scaled Conjugate Gradient Neural Network for Optimizing Indoor Positioning System. In Proceedings of the 2019 International Symposium on Networks, Computers and Communications (ISNCC), Istanbul, Turkey, 18–20 June 2019; pp. 1–4. [CrossRef]
64. Kumar, U.A. Comparison of neural networks and regression analysis: A new insight. *Expert Syst. Appl.* **2005**, *29*, 424–430. [CrossRef]
65. Papila, N.; Shyy, W.; Fitz-Coy, N.; Haftka, R. Assessment of neural net and polynomial-based techniques for aerodynamic applications. In Proceedings of the AIAA 17th Applied Aerodynamics Conference, Norfolk, VA, USA, 28 June–1 July 1999. [CrossRef]
66. Campagnini, S.; Liuzzi, P.; Galeri, S.; Montesano, A.; Diverio, M.; Cecchi, F.; Falsini, C.; Langone, E.; Mosca, R.; Germanotta, M.; et al. Cross-Validation of Machine Learning Models for the Functional Outcome Prediction after Post-Stroke Robot-Assisted Rehabilitation. In Proceedings of the 2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC), Glasgow, UK, 11–15 July 2022; pp. 4950–4953. [CrossRef]
67. Powers, D.M.W.; Atyabi, A. The Problem of Cross-Validation: Averaging and Bias, Repetition and Significance. In Proceedings of the 2012 Spring Congress on Engineering and Technology, Xi'an, China, 27–30 May 2012; pp. 1–5. [CrossRef]
68. Yadav, S.; Shukla, S. Analysis of k-Fold Cross-Validation over Hold-Out Validation on Colossal Datasets for Quality Classification. In Proceedings of the 2016 IEEE 6th International Conference on Advanced Computing (IACC), Bhimavaram, India, 27–28 February 2016; pp. 78–83. [CrossRef]
69. Alippi, C.; Roveri, M. Virtual k-fold cross validation: An effective method for accuracy assessment. In Proceedings of the 2010 International Joint Conference on Neural Networks (IJCNN), Barcelona, Spain, 18–23 July 2010; pp. 1–6. [CrossRef]
70. More, A.S.; Rana, D.P. Review of random forest classification techniques to resolve data imbalance. In Proceedings of the 2017 1st International Conference on Intelligent Systems and Information Management (ICISIM), Aurangabad, India, 5–6 October 2017; pp. 72–78. [CrossRef]
71. Kam Ho, T. Random decision forests. In Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, Canada, 14–16 August 1995; Volume 1, pp. 278–282. [CrossRef]
72. Yang, M.; Xie, B.; Dou, Y.; Xue, G. Cascade Forward Artificial Neural Network based Behavioral Predicting Approach for the Integrated Satellite-terrestrial Networks. *Mob. Netw. Appl.* **2022**, *27*, 1569–1577. [CrossRef]
73. Zhang, H.; Zhang, L.; Jiang, Y. Overfitting and Underfitting Analysis for Deep Learning Based End-to-end Communication Systems. In Proceedings of the 2019 11th International Conference on Wireless Communications and Signal Processing (WCSP), Xi'an, China, 23–25 October 2019; pp. 1–6. [CrossRef]