Project Report
TIP-196

Rapid Aerodynamic Analysis through Deep Learning: FY23 Engineering Research Technical Investment Program

M. C. Jones
S. H. Spreizer
C. Cubra
J. Crouse

24 January 2024

Lincoln Laboratory
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LEXINGTON, MASSACHUSETTS

This material is based upon work supported by the Under Secretary of Defense for Research and Engineering under Air Force Contract No. FA8702-15-D-0001.

This report is the result of studies performed at Lincoln Laboratory, a federally funded research and development center operated by Massachusetts Institute of Technology. This material is based upon work supported by the Department of the Air Force under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of the Air Force.

Massachusetts Institute of Technology
Lincoln Laboratory

Rapid Aerodynamic Analysis through Deep Learning: FY23 Engineering Research Technical Investment Program

M.C. Jones
S. H. Spreizer
J. Crouse
Group 74

C. Cubra
DEC

Project Report TIP-196

24 January 2024

This material is based upon work supported by the Department of the Air Force under Air Force Contract No. FA8702-15-D-0001.

Lexington Massachusetts

This page intentionally left blank.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS
(Continued)

LIST OF TABLES

This page intentionally left blank.

1. OVERVIEW

1.1 PROBLEM STATEMENT

MIT Lincoln Laboratory designs and fabricates airborne sensors to solve problems across a range of disciplines. These sensors are flown on Lincoln Laboratory Flight Test Facility aircraft. In order to certify airworthiness of an aircraft modification before it is flown, substantial analysis is performed. Of interest for this work is the aerodynamic performance of modifications made to the aircraft.

Aerodynamic simulation is performed in order to estimate quantities of interest for a given aircraft configuration. Some of these quantities include lift, drag, moments, wake properties, local flow directions, and vibrational profiles. Drag is a major contributor to fuel burn, maximum flight duration, and flight performance limitations. Other listed properties all contribute as well to the airworthiness determination of a particular aircraft configuration.

Traditional analysis methodologies can be broken down into three major steps which are outlined in Figure 1. First, a high-fidelity model of the aircraft is required. Obtaining this model may involve taking a laser scan of an aircraft and any modifications which may be made to it and then simplifying this geometry by hand to a point where it can be meshed, a process which takes many hours of hands-on effort. Next, a high-quality mesh with fully resolved boundary layers is constructed. This process requires hours of hands-on work by the analyst plus additional time for the generation of the mesh after sizing has been prescribed. Finally, the aerodynamics simulation can be run on a supercomputer which may take days to weeks, especially if time-dependent phenomena like vortex shedding or vibration need to be modeled.

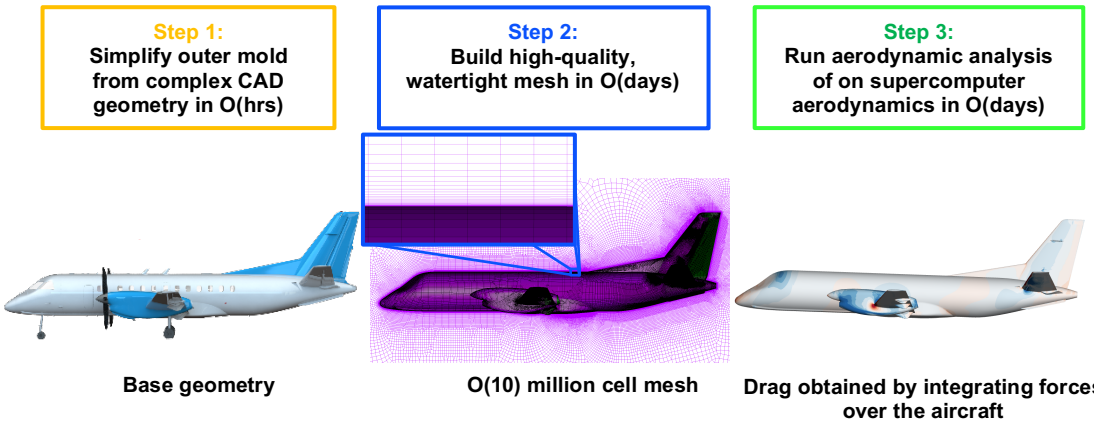

Figure 1. Conventional workflow for aerodynamic analysis is time and labor intensive.

This entire process is expensive both in terms of computational resources and in terms of the hands-on time from the expert analyst. The turnaround for such simulations is thus slow and can hinder the speed of the conceptual design phase for an aircraft modification where many different configurations are sought to be rapidly explored. The goal of this work is to apply modern machine learning techniques to this analysis workflow in order to drastically increase the rate at which design configurations can be explored.

1.2 TECHNICAL APPROACH

The Rapid Aerodynamic Analysis through Deep Learning (RAADL) Technical Initiative employs *deep learning*, a sub-category of machine learning, in the prediction of aerodynamic quantities, specifically the aerodynamic drag, for this study. Arbitrary geometries are discretized into machine-readable tensors of volume fractions. A Convolutional Neural Network is trained to recognize spatial features of the geometries and relate those features to their resulting aerodynamics for a given flight condition, as illustrated in Figure 2. The tool estimates aerodynamic drag for arbitrary geometries in a matter of seconds, providing the designer with the ability to assess aerodynamic performance for numerous concepts. Even so, as a proof of concept, the RAADL tool is limited based on the breadth of the aircraft training data, and uncertainties are large for aircraft that are dissimilar from the training data.

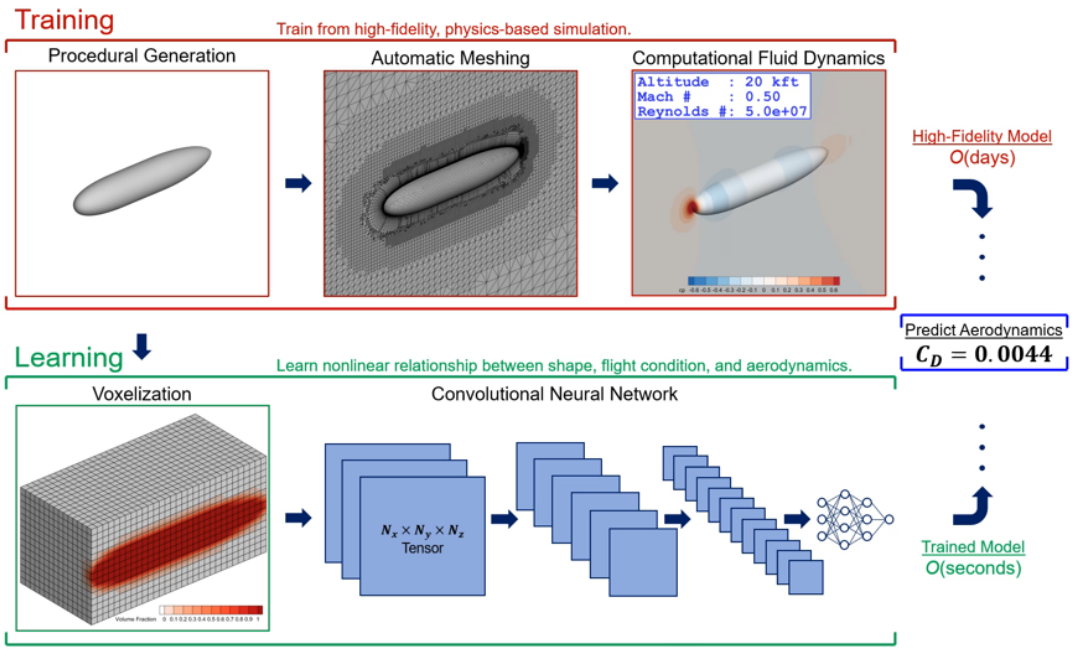

Figure 2. Illustration of how high-fidelity, fluid dynamics data trains the RAADL network in the prediciton of aerodynamic drag.

1.3 MISSION IMPACT

This tool enhances Lincoln Laboratory's capacity to evaluate airborne sensors, aircraft modifications, and other aircraft concepts. Notable mission impacts are summarized in Table 1. The enabling contribution is allowing for high-fidelity predictions in a very short amount of time. Application to the realm of multidisciplinary design optimization allows for more-efficient designs to be identified, thereby improving operational performance. Lower-drag designs will improve an aircraft's time on station, fuel savings, and maximum speed. This particular tool is just one piece in studying broader, multi-disciplinary problems. This approach can be expanded to evaluate additional quantities of interest such as lift, moment, stability derivatives, and vibration spectra, or even other disciplines, such as structural and thermal analysis. The RAADL tool is easy to install from the Lincoln Laboratory GitHub with a single command.

Table 1. Mission Impact of the Work

Impact	Explanation
Improved airborne rapid prototyping	• Evaluate numerous concepts with high-fidelity results in conceptual design phase. • Hook into multidisciplinary design optimization routine.
Airworthiness tool	Make rapid and accurate assessment of airworthiness for future modifications.
Fuel savings	Design more-efficient aircraft modifications and airborne sensors.
Future multidisciplinary capabilities	The methodology can be extended to stability, structures, thermal analysis, and more.

This page intentionally left blank.

2. DEEP LEARNING MODEL

2.1 NETWORK DIAGRAM

In this work we apply deep learning, a sub-category of machine learning, to predict the aerodynamics of an aircraft. The deep learning model is trained on a set of data so that it can learn the complex patterns and nonlinear relationships within that data. This approach is widely applicable to general engineering and practical real-world problems.

The deep learning network is composed of Convolutional Neural Networks (CNNs), which inherently have the capacity to recognize spatial features such as the curvature of a tail-cone or the shape of a fin. The overall approach is based on a previous implementation of deep learning in the analysis of fluid flow over marine vehicles [1]. Two separate approaches were implemented: (1) direct prediction of aerodynamics and (2) prediction of the fluid flow itself. A more detailed explanation of the direct-prediction model is provided in a previous report [2]. In both cases, the network code was written using the popular Python machine learning framework known as PyTorch [3].

2.1.1 Direct-Prediction Model

The network diagram for the direct-prediction model is shown in Figure 3. The aircraft geometry is input as a standard CAD file and then "voxelized" into a 3D tensor of volume fractions. This tensor is passed through a series of convolutions that reshape the tensor into a 1D array of latent variables that define the shape. The flight condition is also input concurrently in terms of altitude, Mach number, and Reynolds number. This flight condition is then concatenated with the encoded latent variables of the aircraft, and subsequently passed through a traditional series of linear neural network layers. The linear layers output the aerodynamic quantities of interest where we focus specifically on aerodynamic drag coefficient C_D, although other quantities of interest can be trained for.

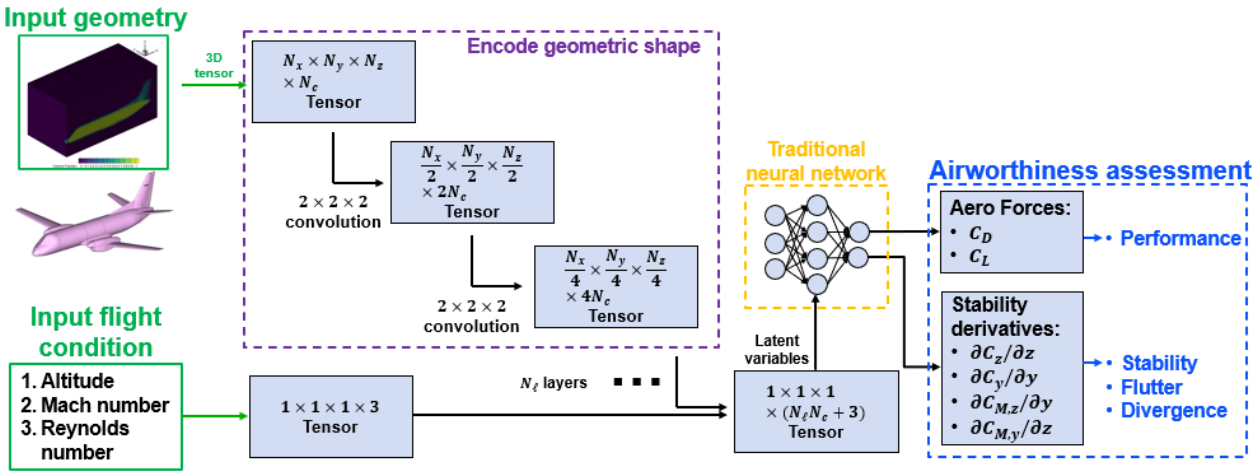

Figure 3. Direct-prediction network diagram: aerodynamic quantities are predicted based on a 3D tensor of volume fractions representing the geometry and the flight condition.

Each layer in the network contains a collection of weights and biases. When training the network, these weights are iteratively calibrated such that predictions from the network can fit the training data. The network *learns* to associate geometric features with their corresponding aerodynamic drag for a given flight condition. The number of convolutional layers, N_ℓ, depends on the size of the input tensor of voxelized volume fractions. Example convolutional layers are listed in Table 2 and Table 3 for input volume fraction tensor dimensions of 16x32x8, and 32x64x16, respectively, although the number of layers and shape of the convolutions is automatically configured to match with the input geometry. Channel size depends on the N_c parameter which is set to a value of 16 for this study. The Parametric Rectified Linear Unit (PReLU) activation function is used for all convolutions.

Table 2. Convolutional Layer Details for Input Volume Fraction Tensor of Dimension 16x32x8

Convolutional Layer #	Channels In	Channels Out	Kernel Size	Stride	Activation Function
1	1	N_c	(2, 2, 2)	(2, 2, 2)	PReLU
2	N_c	$2N_c$	(2, 2, 2)	(2, 2, 2)	PReLU
3	$2N_c$	$4N_c$	(2, 2, 2)	(2, 2, 2)	PReLU
4	$4N_c$	$8N_c$	(2, 4, 1)	(2, 2, 2)	PReLU

Table 3. Convolutional Layer Details for Input Volume Fraction Tensor of Dimension 32x64x16

Convolutional Layer #	Channels In	Channels Out	Kernel Size	Stride	Activation Function
1	1	N_c	(2, 2, 2)	(2, 2, 2)	PReLU
2	N_c	$2N_c$	(2, 2, 2)	(2, 2, 2)	PReLU
3	$2N_c$	$4N_c$	(2, 2, 2)	(2, 2, 2)	PReLU
4	$4N_c$	$8N_c$	(2, 2, 2)	(2, 2, 2)	PReLU
5	$8N_c$	$16N_c$	(2, 4, 1)	(2, 2, 2)	PReLU

The latent variables output from the convolutional layers are concatenated with the input flight condition. This 1D, concatenated array is then fed into a series of linear layers, typical to a traditional neural network. For this study, we employ three layers as detailed in Table 4. N_{in} represents the size of the concatenated array. $N_{quantities}$ represents the number of output quantities of interest. For this study, $N_{quantities} \equiv 1$ because we are only predicting aerodynamic drag, C_D. The Parametric Rectified Linear Unit (PReLU) activation function is used for all layers except the last, which is the Hyperbolic Tangent (tanh) activation function. Because tanh varies between -1 and 1, a final post-processing step scales the output to its real magnitude.

Table 4. Linear Layer Details

Linear Layer #	Features In	Features Out	Activation Function
1	N_{in}	$N_{in}/2$	PReLU
2	$N_{in}/2$	$N_{in}/4$	PReLU
3	$N_{in}/4$	$N_{quantities}$	Tanh

In the direct-prediction model, we optionally apply Bayesian machine learning using the "Bayesian-torch" Python package [4]. By training a Bayesian-based network, we no longer make single, deterministic predictions, but instead a distribution of predictions. This concept is illustrated in Figure 4. In doing so, we can estimate uncertainty in the predictions. This uncertainty estimation is helpful in understanding the level of confidence in predictions, particularly on aircraft that are dissimilar in shape from the training data.

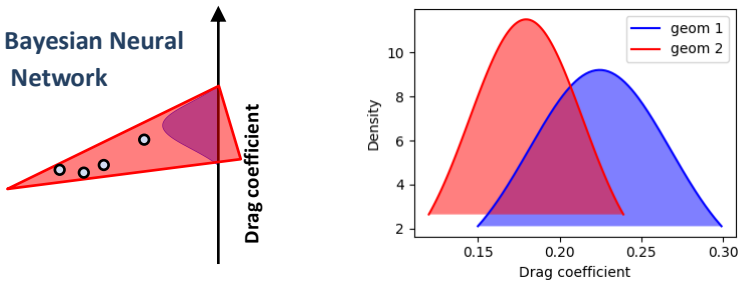

Figure 4. Illustration of predictions using the Bayesian Neural Network.

2.1.2 Flowfield-Prediction Model

The second model is referred to as the flowfield prediction model because it attempts to predict the entire fluid flowfield, which can then be post-processed for quantities of interest. This approach is advantageous for two reasons. First, predictions are more generalized and widely useful. Any arbitrary quantity can be processed from the flowfield, rather than having to retrain the model each time there is a new quantity of interest. Second, we can apply concepts from physics-informed machine learning [5]. We train the model not just to fit well with the training data but also to satisfy the physical governing equations defining the fluid dynamics around the aircraft.

The overall structure is based on the U-Net architecture [6] and is illustrated in Figure 5. The structure can be broken down into two primary sections: the encoder layer and the decoder layer. Each layer is composed of a number of encoder/decoder blocks. The encoder layer, like in the drag prediction model, calculates latent variables which describes the geometry. The decoder then uses these latent variables along with the input flight condition in order to reconstruct the flow field around the aircraft. A number of skip connections are included from the encoder to the decoder layers in order to provide more information to the decoder blocks to better recreate the flow field. Physical constraints are optionally applied to the output of the decoder blocks in order to promote reconstruction of physically meaningful flow fields.

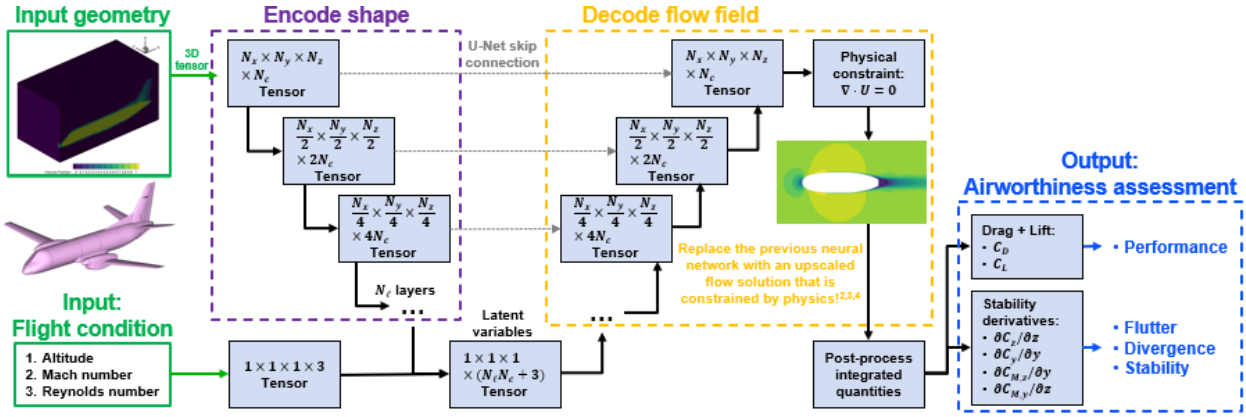

Figure 5. Flowfield-prediction network diagram: aerodynamic quantities are computed from the 3D predicted flowfield around the aircraft.

2.2 NETWORK ITERATIVE SETTINGS

Through the training process, weights are solved for iteratively using the Adam optimizer [7] with default settings. Xavier initialization decides the initial weights and biases. The loss function is the Mean Square Error. The learning rate is initially set to 10^{-3}, but is automatically adjusted using the PyTorch "ReduceLROnPlateau" learning rate scheduler with a threshold value of 10^{-5}. The code was written in such that the network weights are written to a file regularly, and the learning may be continued if it were inadvertently interrupted. Additionally, the iteration and loss values are output such that they can be continuously plotted and examined in real time.

2.3 VOXELIZATION

A fundamental hurdle in the deep-learning prediction of aerodynamics for arbitrary shapes is the characterization of those shapes in a machine-readable format. For our approach, we built an automated algorithm to discretize an input shape into a collection of voxels (i.e., cubes). Within each voxel, we compute the volume fraction of the occupying aircraft geometry. For example, if 40 percent of a given voxel was occupied by the aircraft, that voxel tensor index would hold a value of 0.4. Figure 6 illustrates the voxelization of the Saab 340B aircraft using three different $I \times J \times K$ resolutions: 8x16x4, 16x32x8, and 32x64x16. Here, I is the number cells in the axial direction, J in the transverse, and K in the vertical. Note that the input shape is assumed to be symmetric wing tip to wing tip to save computational resources, so $J = 0$ is the center of the aircraft.

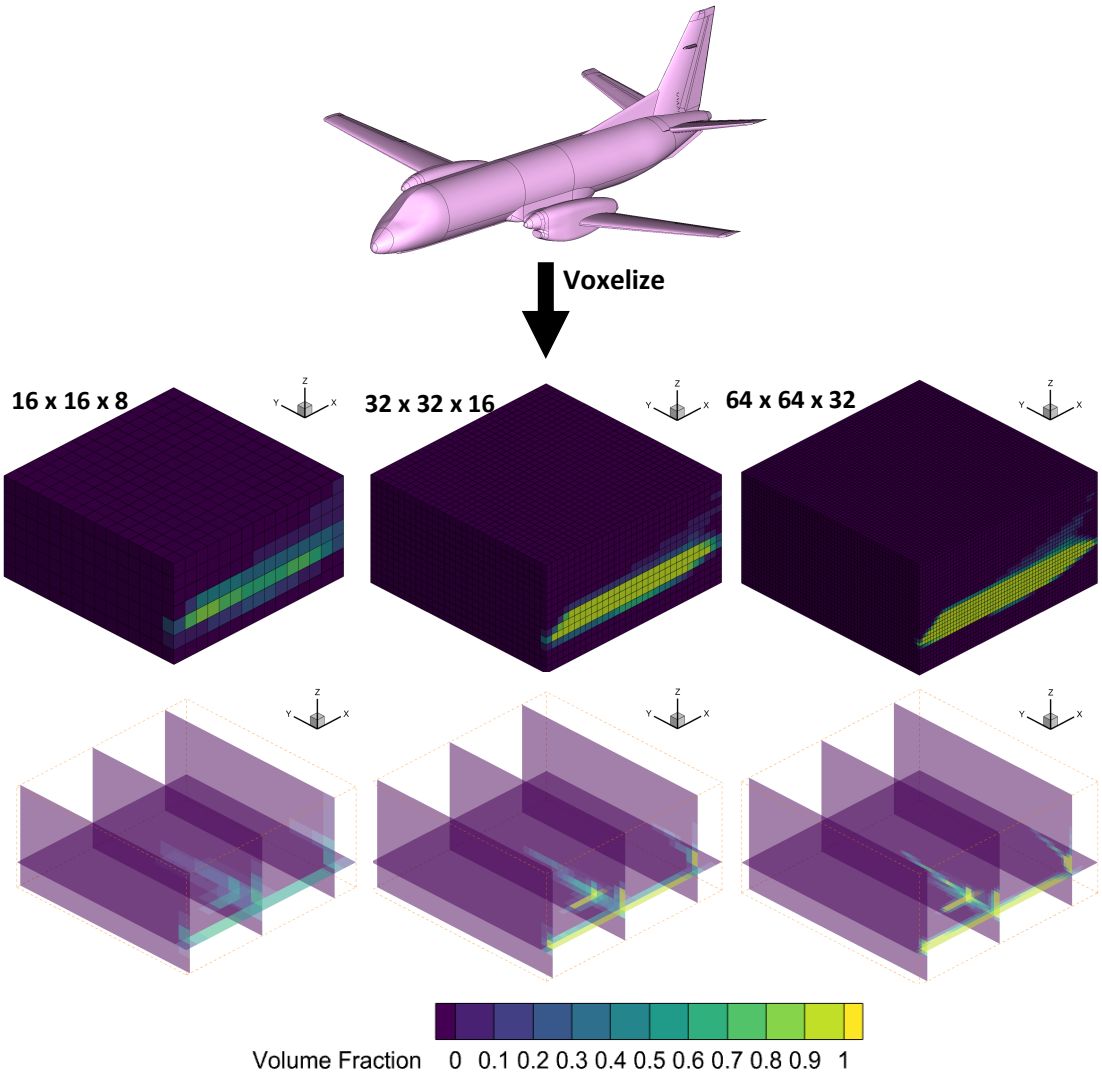

Figure 6. Voxelization example of the Saab 340B aircraft using three different resolutions: 16x16x8, 32x32x16, and 64x64x32.

Adequate sizing of $I \times J \times K$ depends on the desired level of feature resolution. For aircraft, the voxelization should be able to capture key features of the plane such the fuselage, wings, and fail. The exact resolution requirement for the desired level of detail on an arbitrary geometry is uncertain, and requires further study.

2.4 SOFTWARE DEVELOPMENT AND BEST PRACTICES

We follow several best practices in software development. First, we follow the PEP8 Python style guide to ensure that the code is readable and follows a standardized convention across the Python community. Second, we employ version control through Lincoln Laboratory's GitHub, which helps to rigorously track changes. If an unintended software bug is introduced, it is easy to revert back to a previous version of the code and correct the issue. Third, we use the "pytest" unit testing Python package [8]. Before committing a new version of the code, we run a series of tests to verify that the individual subroutines of the code are running correctly and the overall code is still running correctly. This helps to identify software bugs and maintain functionality of the code.

This page intentionally left blank.

3. TRAINING THE MODEL

3.1 STEPS TO TRAIN THE MODEL

The model must be trained to build a functional relationship between aircraft shape and its associated aerodynamic properties. This is a fundamental challenge because aircraft shapes are widely varied, and small changes in curvature can lead to significant changes in the flowfield around the surface of the aircraft. Three additional degrees of freedom are introduced from the flight condition. In other disciplines of machine learning there are large, publicly available databases that are available for research. This is not the case for aircraft and their associated flowfields. So, in this study, we begin to build this database by parameterizing an aircraft shape and conducting thousands of simulations to find associated properties that a machine learning model can learn.

Steps for generating the training database and then training the model are listed in Figure 7. This entire process is automated to efficiently generate a large training dataset. First, we generate numerous, varied aircraft geometries based on a parametric model. Second, we build computational meshes for each of those geometries. The automated meshing process creates a surface mesh on the aircraft and then performs a volume meshing to create the fluid domain. Third, we generate Computational Fluid Dynamics (CFD) case files for each of the geometries over a variety of flight conditions. Fourth, we run the CFD simulation cases using available high-performance computing resources. Concurrently, we voxelize the geometries, as described in Section 2.3. We then extract aerodynamic results and aggregate all data that associates voxelized geometries and an input flight conditions with their expected aerodynamics. Finally, we train the model from this data. After the model is trained, it is ready to run and predict the aerodynamics of any arbitrary geometric shape.

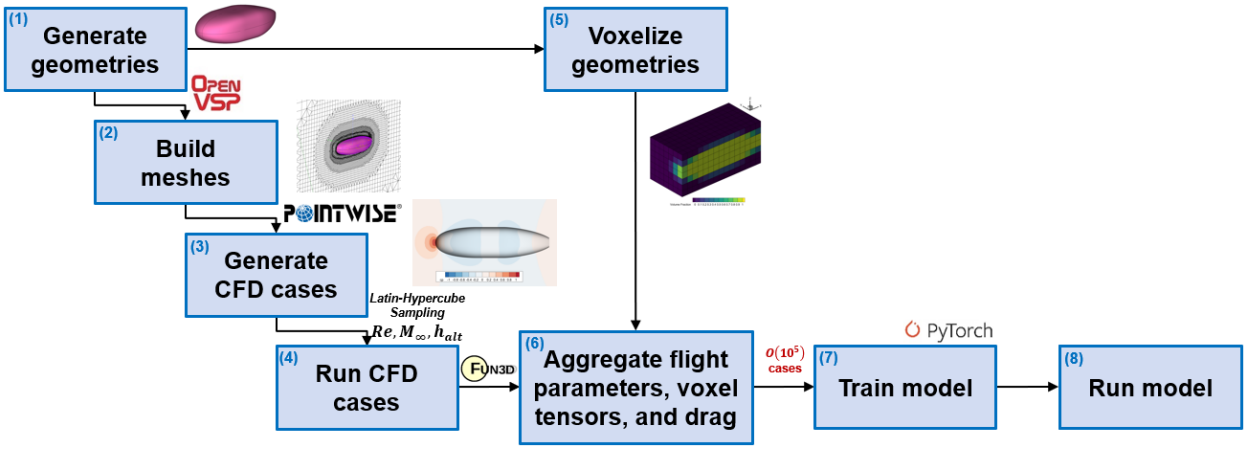

Figure 7. Steps to train the model.

3.2 GEOMETRY GENERATION

A diverse set of aircraft geometries needed to be generated in order to effectively develop a model which can predict on actual aircraft. To develop this dataset, a parameterized representative glider aircraft was created in Open Vehicle Sketch Pad (OpenVSP), an open-source parametric geometry tool released by NASA [9]. The geometry is composed of three major sections: the fuselage, the wing, and a v-tail. The baseline parameterized model can be seen in Figure 8.

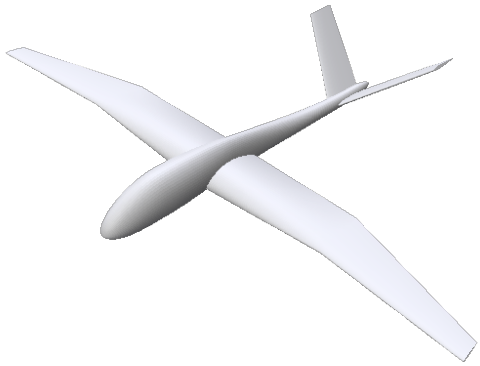

Figure 8. Parameterized aircraft geometry.

The glider is parameterized by eight values: fuselage length-diameter ratio, fuselage shape, wing aspect ratio, wing taper ratio, wing sweep angle, wing dihedral angle, wind twist, and airfoil thickness. To standardize the geometries for analysis, all fuselages are set to a length of one. The cross-section of the fuselage can be varied between a circle and rectangle, with the fuselage shape parameter controlling a blending between the two. Full descriptions of all parameters and their ranges can be found in Table 5.

Table 5. Parameters to Define Glider Aircraft Shape

Parameter	Description	Range
Fuselage length-diameter ratio	Controls diameter of fuselage (length fixed at 1).	[7, 11]
Fuselage shape	Control fuselage cross-section blending between circular and rectangular.	[2, 5]
Wing aspect ratio	Wing span squared divided by wing planform area.	[2, 6.5]
Wing taper ratio	Ratio of tip chord to root chord.	[0.5, 1]
Wing sweep angle [deg]	Angle of wing sweep aft.	[0, 5]
Wing dihedral angle [deg]	Angle of wing elevation.	[0, 10]
Wing twist [deg]	Local pitch of tip of wing relative to root (unused for this study)	[0, 0]
Airfoil thickness	Thickness ratio of airfoil	[0.08, 0.15]

To generate the training data, Latin Hypercube Sampling (LHS) is employed to perform a methodical variation across the ranges of the eight parameters. LHS possesses the advantage that a good representation of the entire parameter ranges will always be selected, a guarantee that purely random sampling cannot provide. Sampling in this manner, a number of training and testing aircraft geometries were generated. Figure 9 contains examples of some of the generated geometries with the variation in the parameters being evident.

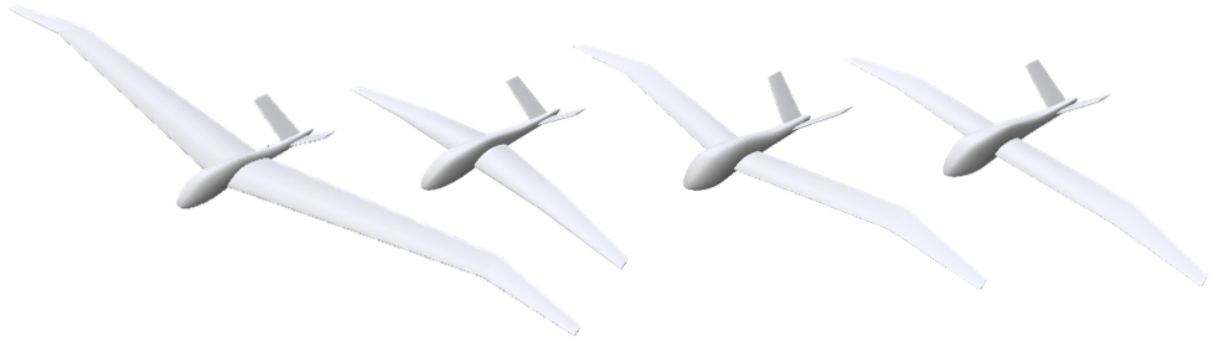

Figure 9. Example generated aircraft with parameter variation.

A Python tool was developed in order to efficiently generate the varied airplane models. It is executable both as a command line or graphical tool. A screenshot of the graphical user interface (GUI) for aircraft generation can be seen in Figure 10. The tool calls the Python API for OpenVSP in order to generate the various parameterized models. The ranges for each of the parameters are input via a text file or as a percentage variation. Parameters are selected using Latin Hypercube Sampling. The surface mesh is saved as an STL file for each of the generated aircraft. Screenshots of different views of the aircraft are also produced.

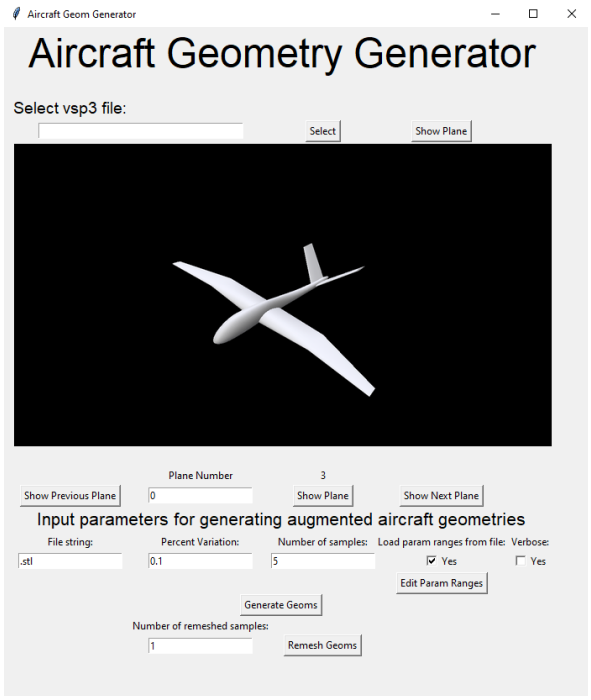

Figure 10. Aircraft geometry generator graphical user interface.

3.3 AUTOMATED MESHING

3.3.1 Mesh Generation Algorithm

In order to simulate the flow around the aircraft models, a computational mesh must be constructed on which to calculate the flow solution. Given the volume of geometries to be simulated (more than a thousand), this process needs to be fully automated. We developed an automated meshing code that utilizes the Pointwise software [10]. This code may either be run from a graphical user interface or directly from a system call. The meshing process is illustrated in Figure 11.

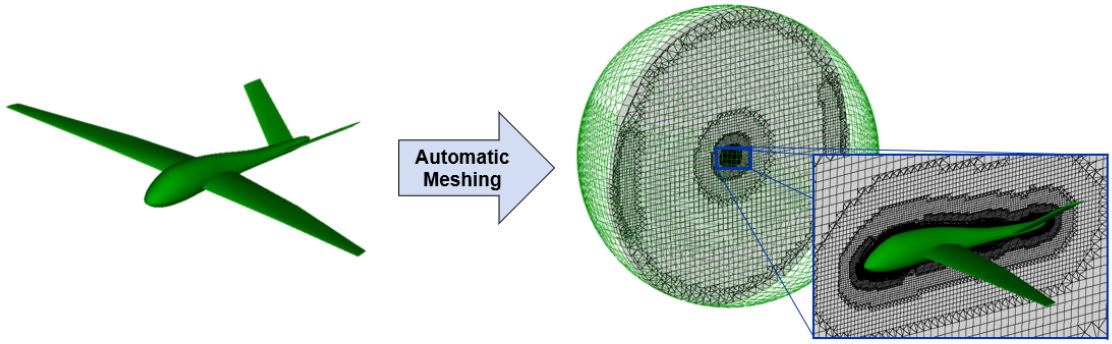

Figure 11. Automatic mesh generation process.

The meshing code builds a high-quality computational domain for an arbitrary, watertight CAD geometry. A large sphere constitutes the far field of this "O-mesh." Cell refinements are made around the features of interest of the aircraft, centered in the O-mesh. The level of refinement was calibrated through a grid independence study such that further refinement to the mesh did not significantly alter predicted aerodynamics.

3.3.2 Resolving the Boundary Layer

Special attention is given to the refinement of cells within the aircraft geometry's boundary layer, which is the thin region where velocity grows from zero on the surface to the freestream speed of the surrounding environment. Proper capturing of the boundary layer is necessary in order to accurately calculate skin friction drag on the surface of the aircraft. We select the height of the first cell in the boundary layer such that the nondimensional wall distance y^+, is less than one for our most conservative flight condition. Having $y^+ < 1$ ensures that the viscous sublayer is resolved. Beyond the first cell, 50 to 100 cells resolve the rest of the boundary layer.

3.4 GENERATING COMPUTATIONAL FLUID DYNAMICS CASES

Like for the geometries, the flight conditions for each aircraft simulation are selected using Latin Hypercube Sampling across altitude, Mach number, and Reynold's number. The ranges for each of these parameters can be found in Table 6. The upper bounds are based on the typical high-speed flight condition experienced by a business jet such as a Gulfstream IV. Any number of flight conditions can be simulated

for each generated geometry. In previous work, diversity in geometry was determined to be more important than diversity in flight conditions, so, we build the training data set with 10 flight conditions per geometry.

Table 6. Flight Condition Bounds

	Lower Bound	Upper Bound
Altitude [kft]	0	50
Mach Number	0.05	0.9
Reynolds Number	10^6	5×10^8

3.5 FLUID DYNAMICS SIMULATION

3.5.1 Numerical Approach

The CFD simulations are run to generate the "ground truth" data used to train and evaluate the machine learning model. The NASA FUN3D solver is used to perform the simulations [11]. A two-stage run process is used for each of the cases. First, a steady state simulation is run using the Spalart-Allmaras (SA) turbulence model [12] with a freestream turbulence intensity of three percent. The solution from the steady run is then continued as a transient simulation using the Delayed Detached Eddy Simulation (DDES) turbulence model which provides a better resolved and more accurate flow solution to use as the ground truth. For both the steady and transient portions of the run, the van Albada flux limiter with the low-diffusion flux splitting "LDFSS" flux construction method is used [13]. All simulations were conducted using Lincoln Laboratory Super Computing resources. To illustrate the variability in aircraft shapes, the first 25 simulated test-case geometries are shown in Figure 12.

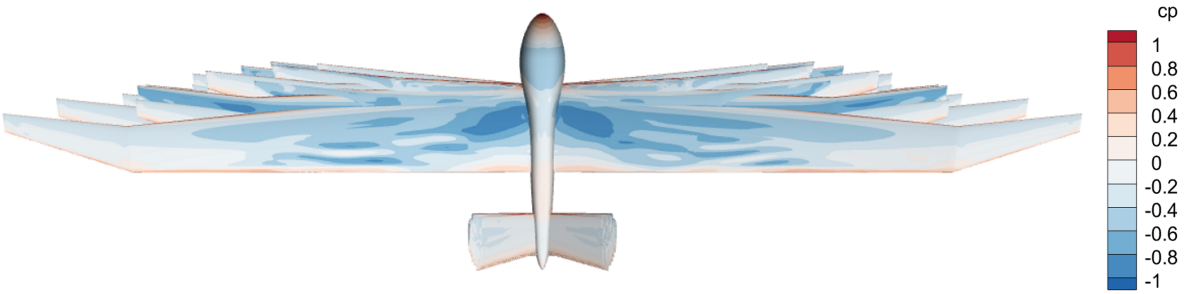

Figure 12. First 25 simulated test-case geometries superimposed on one another and colored by coefficient of pressure.

3.5.2 Iterative Convergence

The simulations are run until they are well converged, that is, the forces and moments converge to singular values. Normalized simulation residuals driven below 10^{-5}. Any cases that do not converged well are discarded from the data. An example flow solution for the flow over a glider aircraft can be seen in Figure 13.

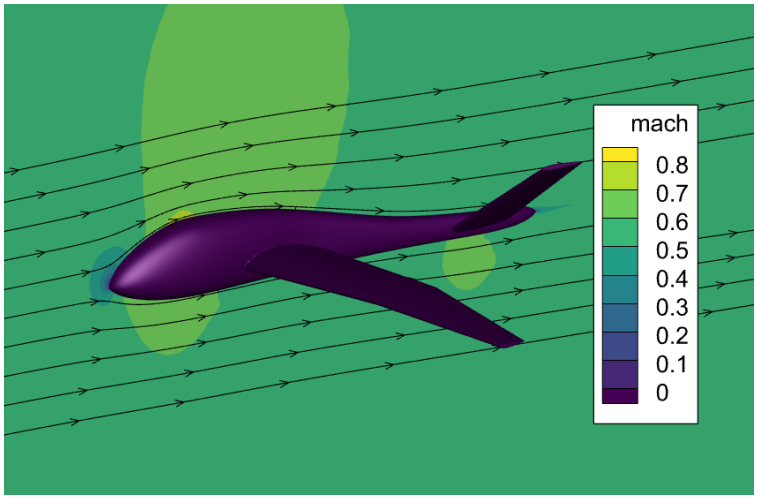

Figure 13. Example Mach number solution and streamlines of flow over an aircraft.

From the simulations, aerodynamic loads on the entire aircraft (e.g., lift, drag, moments) are saved for each of the flight conditions on which a geometry is simulated. Additionally, for the flow field prediction, the flow state (velocity, density, and pressure) is saved at each of the voxel centers that do not contain the aircraft to allow for training of the flow field model.

3.6 TRAINING THE MODEL

3.6.1 Initialization and Optimization

Initial model weights and biases are computed using Xavier initialization. Optimization of the network parameters is performed used the Adam optimizer [7] with an adaptive learning rate. Using an adaptive learning rate is beneficial in training to a useful solution. The PyTorch "ReduceLROnPlateau" learning rate scheduler implements this functionality and is provided bounds on the learning rate. Selection of the initial learning rate has a large impact on the convergence of the model. Too high, and an optimal

solution cannot be reached. Too low, and training will take longer than is feasible. Figure 14 shows an example of the loss during training for the direct prediction model on the 1000 geometry dataset. The sharp changes in slope are indicative of the adaptive learning rate adjusting as the training progresses.

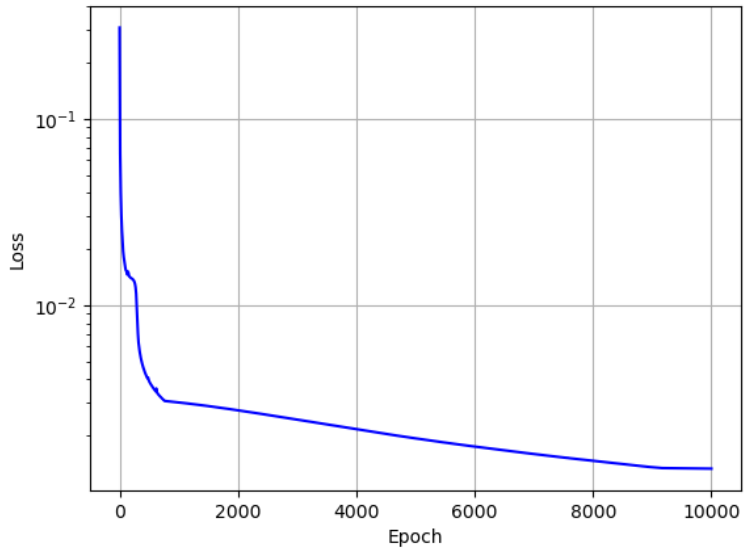

Figure 14. Example training loss history.

During the training process, model state is regularly saved in order to be able to restart training if the code is interrupted or to continue with new user input. Per iteration loss is saved to allow for plotting and to monitor real-time convergence of the network.

3.6.2 Loss Formulation

For both the direct prediction and flow field models, mean square error is used in the loss function. For the direct prediction model, error in the predicted drag coefficient is calculated with respect to the "ground truth" training data. The flow field model computes the error in the flow properties at each of the voxel centers not occupied by the aircraft body.

An addition loss term is applied in the case of the flow field prediction model. The flow field around an aircraft is required to satisfy the equations governing fluid flow, in our case the compressible Navier-Stokes equations. Thus, when a prediction of the flow field is made, we would like it to also satisfy these governing equations. To promote this in the learned model, training is performed with an additional loss function evaluating satisfaction of the governing equations (i.e., a physical constraint on the model output).

For this work, residual of the compressible continuity equation,

$$\vec{\nabla} \cdot \left(\rho \, \vec{V}\right) = \ell$$

is calculated via a finite difference approximation on the voxels using 2nd-order, central differences. This value is added to the mean square error on the data with the goal of forcing the model to only learn flow fields which satisfy this constraint and are thus physically meaningful.

A weighting factor is included in order to balance the impact of each of the two loss terms for the flow field model. Too much weight on the training data loss may lead to a solution that doesn't well satisfy the governing equations of the flow. Too much weight on the continuity equation loss has the potential to drown out the training data and lead to a trivial and non-meaningful flow prediction.

This page intentionally left blank.

4. MODEL PERFORMANCE

4.1 AIRCRAFT DRAG PREDICTIONS

The performance of the direct drag prediction model was evaluated using simulation data for the SAAB 340B aircraft. A trained model for the 8x16x4 voxel case was used. Table 7 shows a summary of the model performance on this data set. Predictions of drag are fairly consistent, demonstrated by the lower standard deviation. All the model predictions underpredicted drag as well which is likely due to fundamental differences in the testing SAAB aircraft versus the training set gliders (e.g., traditional tail vs. v-tail). The reasonably low mean error supports performance of the model trained on the representative gliders even when applied to real aircraft.

Table 7. Summary of Model Prediction on SAAB 340B Aircraft when Trained on Gliders

Metric	Mean Abs. Error	Max Abs. Error	Min Abs. Error	Standard Deviation of Error
Value	13.7%	23.6%	5.6%	6.1%

Another study was performed evaluating the performance of this model when training on an aircraft geometry and then testing the same geometry at new flight conditions. This analysis was performed using the Gulfstream G-IV aircraft. Using an approximately 60/40 split of the aircraft simulation data for the training and testing, results of this study can be seen summarized in Table 8. Drag prediction of the direct model performs reasonably well in this case with good mean error but higher maximum observed error.

Table 8. Summary of Model Prediction on G-IV Aircraft when Trained on G-IV Data

Metric	Mean Abs. Error	Max Abs. Error	Min Abs. Error	Standard Deviation of Error
Training Value	8.5%	41.4%	0.1%	12.2%
Testing Value	10.0%	41.4%	0.1%	13.6%

4.2 FLOWFIELD PREDICTIONS

The trained model is capable of predicting the fluid flowfield around an aircraft in terms of velocity, pressure, and density. An example flowfield prediction for a glider aircraft is shown in Figure 15 with a voxelization resolution of 16x32x8. This example is representative of broader results for other aircraft shapes. The flowfield prediction is generally accurate, but there are still small regions of error. We

investigated varying the weight the physics-informed, continuity equation constraint to reduce error. With the present formulation, highly-weighting the physics-informed constraint drives the solution to zero velocity, resulting in large inaccuracies. It is speculated that the continuity constraint is not helpful for such coarse resolutions of the flowfield, because solutions will ignore more complicated features in the true flowfield. So, we only include a small weight to the continuity constraint, typically less than ten percent.

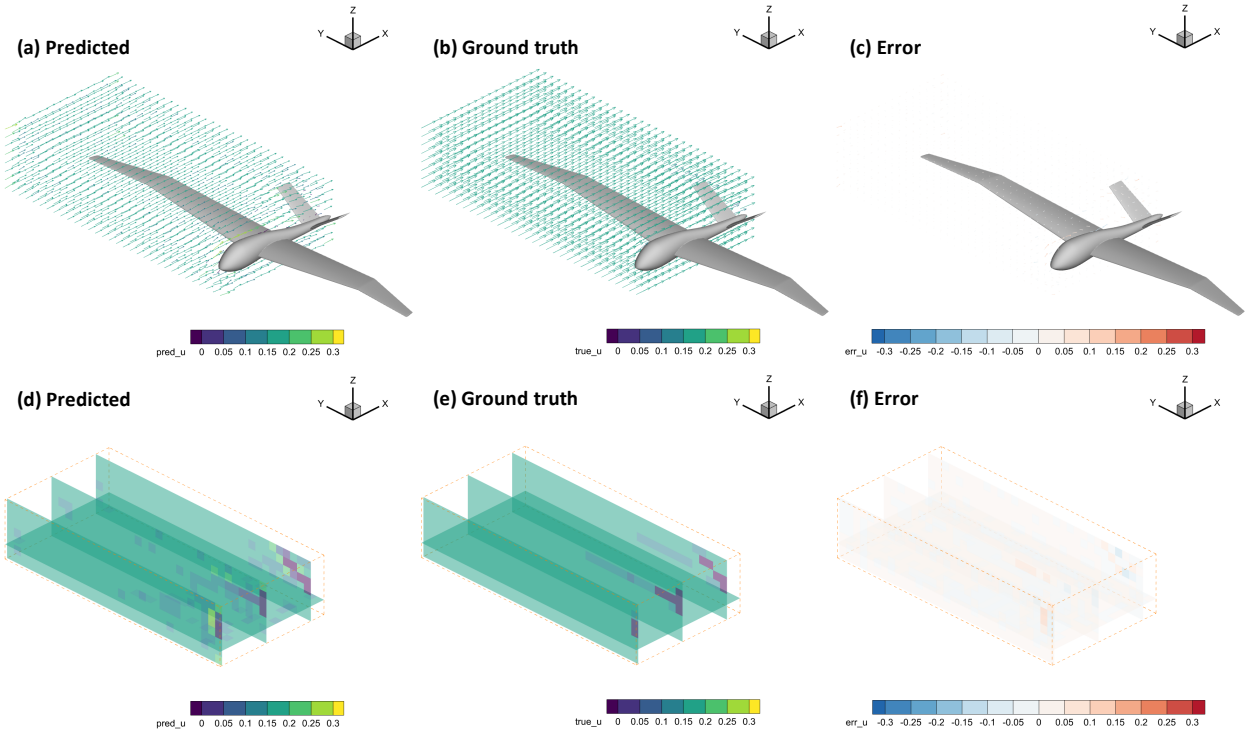

Figure 15. Example flowfield prediction over a glider aircraft visualized as velocity vectors and normalized by the speed of sound with (a) the predicted flowfield, (b) the true flowfield found directly from simulation, and (c) the error. Cutting planes show an analogous comparison in (d), (e), and (f).

The general flow-field predictions are reasonable, but they are limited in resolution. Existing methods used for sampling flow data from CFD simulations were limited to a maximum sample size of 10,000, so without improving these processing techniques, the output flowfield was limited to resolutions that multiplied to less than 10,000, such as 16x32x8, or 4096 points. Attempting to sample 32x64x16, or 32768 points, for example, resulted in errors in the algorithm. With such coarseness, it is not possible to resolve some important features of the flow such as the boundary layer.

These limitations motivate future work to improve the utility of these predictions. Future development in the sampling algorithm is necessary to increase the resolution of the current model. Alternatively, reworking the overall approach to employ graph neural networks shows promise. Additional development is also required to integrate aero loads over the geometry to estimate lift, drag, and moments.

4.3 RUN TIME PERFORMANCE

Once the model is trained, predictions can be made in a matter of seconds. The total run time is directly related to the voxelization resolution and the file size (i.e., detail), of the input aircraft geometry. Run time performance for the direct-prediction model is shown in Figure 16. Run time performance for the flowfield-prediction model is the same order of magnitude and scales similarly; however, there may be a need for higher resolutions to adequately resolve the flowfield. In this case, it is desirable to make improvements to the speed of voxelization, perhaps by rewriting it to run on GPUs or through parallel processing.

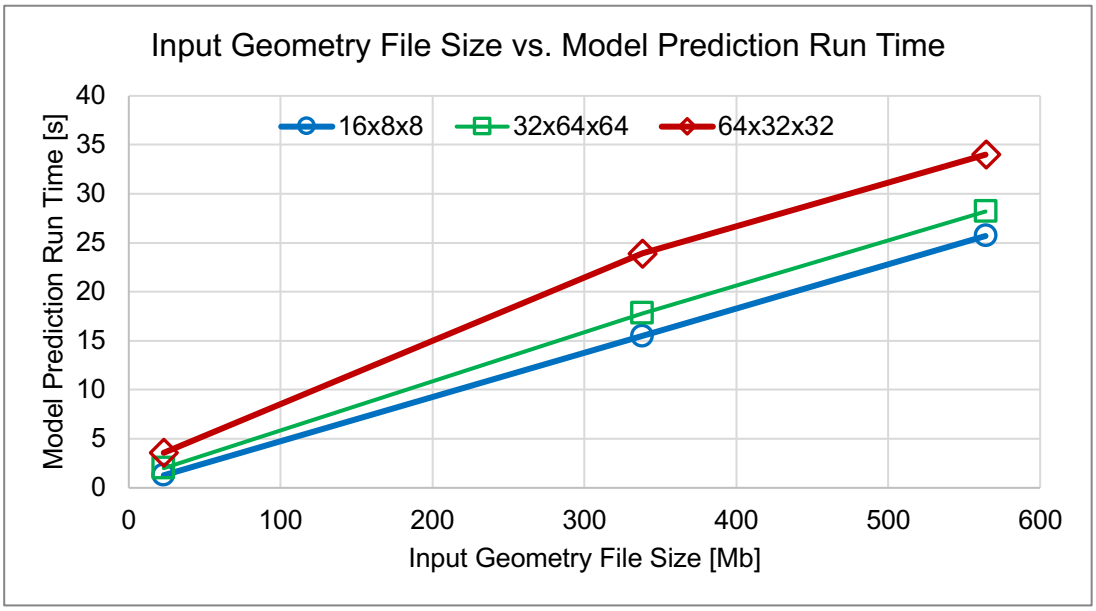

Figure 16. Model prediction time vs. voxelization and input geometry file size.

4.4 TRAINING TIME REQUIREMENTS

We present training time requirements to demonstrate practical limitations in the training database size, and, consequently, the diversity. Figure 17 shows the required time to mesh, simulate ground truth data, and train both of the deep learning models. In terms of computer hardware, in each step we use Xeon-g6 computational nodes. We employ GPU-acceleration in the fluid dynamics simulation using 40

concurrent Xeon-g6 nodes to run 40 separate simulations simultaneously. GPU-acceleration was not implemented for the deep learning training, and this type of speedup is desired in future work.

Figure 17. Meshing, fluids simulation, and model training run time requirements vs. total number of training cases.

Figure 17 shows that when using 1000 training cases, the total time required is on the order of a day, however, the time requirements increase exponentially with training dataset size. When using 100,000 training cases, the time requirement for meshing is approximately 46 days, for simulation: 30 days, and for the deep learning training of the direct-prediction model: 24 days. The total time requirement in this scenario is approximately three months to complete, which is prohibitively expensive in many situations. Training dataset sizes on the order of 10,000 are more justifiable and take approximately ten days to complete for the direct-prediction model. Training for flow prediction is more intensive, however, as the time requirement grows much more quickly with the number of training cases. In this case, training dataset sizes are limited to O(1000) cases until GPU acceleration can be implemented in the deep learning training.

4.5 SOURCES OF UNCERTAINTY

Given the moderate uncertainty in model predictions, it is important to understand sources of error, so as to reduce them in the future. Sources of uncertainty are listed in Table 9. The quality of the training data is most-important because all predictions are fundamentally derived from that data. In generating this data, errors arise in the computational modeling of the fluid flow and discretization of the fluid domains. Discretization error also appears in the voxelization representation of geometries, and can filter out important features of the aero-shape if the resolution is too coarse. Typical machine learning sources of uncertainty are listed as well. We propose several thrusts as future work to address these uncertainties and improve the model.

Table 9. Sources of Uncertainty in Overall Approach

Source of Uncertainty	Explanation
Training Data Error	The "ground truth" aerodynamics of training data geometries comes from fluid dynamics simulations, and the predicted aerodynamics can only be as accurate as these results. Sources of error include: discretization error, modeling error, iterative error, and roundoff error.
Voxelization Representation	The discrete voxelization imposes a level of resolution to capture features of the aircraft. Aircraft features smaller than the voxels are not spatially resolved, and are only captured by the model as a scalar volume fraction. There is also error in the computation of volume fractions.
Network Modeling Error	The network model has a limited capacity to learn spatial features and correlate them to realistic predictions.
Over/Under Fitting	The network model may over or under-fit the training data.

4.6 FUTURE WORK

We seek to improve some of the limitations in the present model. The most substantial proposed change is changing from a U-Net architecture to that of a Graph Neural Network (GNN) [14], as shown in Figure 18. In doing so, our model can learn from computational grids explicitly, rather than uniform, sampled points in the current approach. This will allow the model to learn intricate, detailed flow behavior at regions requiring finer detail. Data in these regions is dense, because we build our mesh to concentrate cells in them. We still seek to apply concepts from physics-informed machine learning [15, 16, 17] but applied to GNN structure.

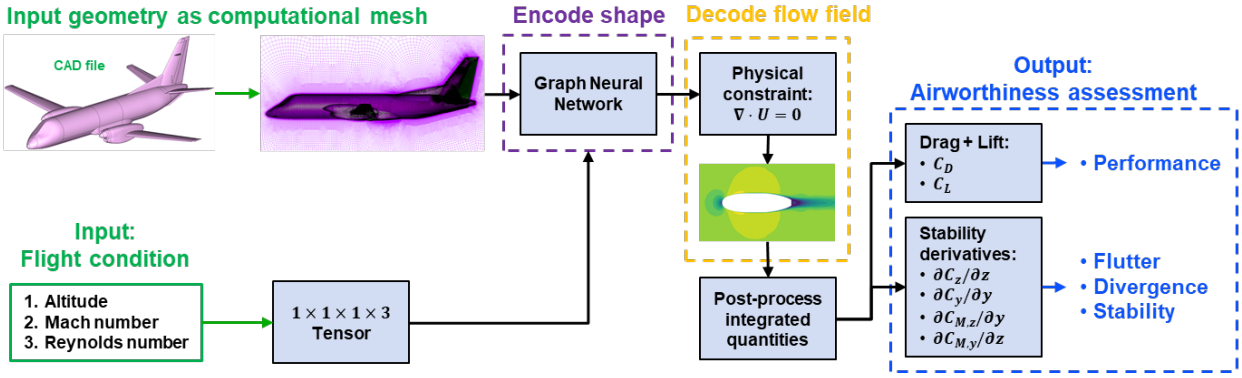

Figure 18. Proposed, future network diagram: we apply a Graph Neural Network to train for flowfield predictions directly from the computational mesh.

Another future task is improving the utility and effectiveness of the training data. We hope to accomplish this in two ways. First, we want to expand the database of aircraft to a greater diversity in shapes beyond those of gliders to other categories. We also hope to include additional real-world aircraft in the training. Second, to make effective use of this training data, we would like to apply concepts in *continual learning*, [18] without catastrophic forgetting. So, the model can maintain accuracy on older data, while effectively learning the parameter space of new data—and not bias one or the other unfairly.

The final goal is to expand the scope of the predictive capabilities in both aerodynamics and other disciplines. In the short term we hope to predict additional airworthiness criteria, such as aerodynamic flutter, for business jets. In the long term, we hope to extend these techniques to other disciplines such as thermal and structural analysis. The future work roadmap is shown in Figure 19.

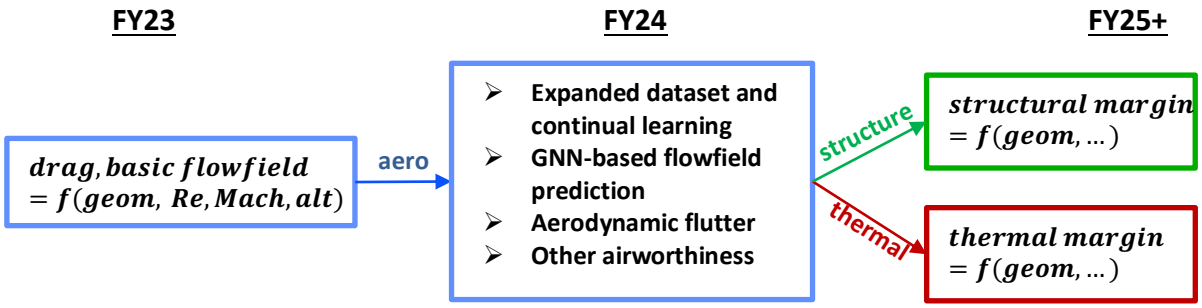

Figure 19. Future work roadmap.

5. CONCLUSIONS

We developed a tool that predicts aerodynamic properties of arbitrary geometries either directly or by first generating a flowfield that can be post-processed for quantities of interest. After training the model, these predictions are conducted in a matter of seconds with the direct-prediction model demonstrating 10% to 14% mean absolute error for real-world aircraft that were not included in the training data. Predictions are fundamentally limited by the diversity of the training data and the quality of predictions is inherently bound to the quality of that data. We sought to improve realism in predictions through the incorporation of a physics-informed neural network; however, the present implementation does not show major improvement. We believe that the formulation of the network should be rewritten as a Graph Neural Network, so that the model can better learn the intricacies of wall-bounded shear flows by having more detail in the regions of interest. This in turn can help to provide better predictions of the flowfield and, subsequently, better airworthiness assessments. As future work we also hope to expand the training data to more diverse aircraft shapes and incorporate concepts of continual learning. Furthermore, we hope to apply these techniques beyond fluid dynamics to structural and thermal mechanics.

This page intentionally left blank.

REFERENCES

[1] M. C. Jones, Accelerating Conceptual Design Analysis of Marine Vehicles through Deep Learning, Blacksburg: Virginia Polytechnic Institute and State University, 2019.

[2] M. C. Jones, S. Kodali and A. McCourt, "Rapid Aero Analysis through Deep Learning: FY22 Engineering Research Technical Investment Program," MIT Lincoln Laboratory, Lexington, 2023.

[3] "PyTorch," 11 2022. [Online]. Available: https://pytorch.org.

[4] R. Krishnan, P. Esposito and M. Subedar, "Bayesian-Torch: Bayesian neural network layers for uncertainty estimation," Jan 2022. [Online]. Available: https://github.com/IntelLabs/bayesian-torch.

[5] S. Cai, Z. Mao, Z. Wang, M. Yin and G. E. Karniadakis, "Physics-informed neural networks (PINNs) for fluid mechanics: A review," *Acta Mechanica Sinica,* vol. 37, no. 12, pp. 1727-1738, 2021.

[6] O. Ronneberger, P. Fischer and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention--MICCAI 2015: 18th International Conference*, Munich, 2015.

[7] D. P. Kingma and B. Jimmy, "Adam: a method for stochastic optimization," *arXiv preprint arXiv:1412.6980,* vol. 22, 2014.

[8] "pytest," 30 11 2022. [Online]. Available: https://pytest.org/.

[9] R. A. McDonald and J. R. Gloudemans, "Open vehicle sketch pad: An open source parametric geometry and analysis tool for conceptual aircraft design," in *AIAA SciTech 2022 Forum*, 2022.

[10] Cadence, "Pointwise," 11 2022. [Online]. Available: https://www.pointwise.com/.

[11] FUN3D, 11 2022. [Online]. Available: https://fun3d.larc.nasa.gov/.

[12] S. R. Allmaras and F. T. Johnson, "Modifications and clarifications for the implementation of the Spalart-Allmaras turbulence model," in *Seventh international conference on computational fluid dynamics (ICCFD7)*, Big Island, Hawaii, 2012.

[13] H. Atkins, N. Alexandrov, K. Bibb, M. Carpenter, P. Gnoffo, D. Hammond, W. Jones, W. Kleb and E. Lee-Rausch, "Team software development for aerothermodynamic and aerodynamic analysis and design," NASA TM-2003-212421, 2003.

[14] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li and M. Sun, "Graph neural networks: A review of methods and applications," *AI open,* vol. 1, pp. 57-81, 2020.

[15] R. Wang, R. Walters and R. Yu, "Incorporating symmetry into deep dynamics models for improved generalization," *arXiv preprint arXiv:2002.03061,* 2020.

[16] R. Wang and R. Yu, "Physics-guided deep learning for dynamical systems: A survey," *arXiv preprint arXiv:2107.01272,* 2021.

[17] J. C. Wong, C. Ooi, P.-H. Chiu and M. H. Dao, "Improved Surrogate Modeling of Fluid Dynamics with Physics-Informed Neural Networks," *arXiv preprint arXiv:2105.01838,* 2021.

[18] S. Beaulieu, L. Frati, T. Miconi, J. Lehman, K. O. Stanley, J. Clune and N. Cheney, "Learning to continually learn}," *arXiv preprint arXiv:2002.09571,* 2020.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
01-24-2024	Project Report	

4. TITLE AND SUBTITLE

Rapid Aerodynamic Analysis through Deep Learning: FY23 Engineering Research Technical Investment Program

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Matthew C. Jones, Steven Spreizer, Cameron Cubra, James Crouse

5d. PROJECT NUMBER
TI95-2625

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

MIT Lincoln Laboratory
244 Wood Street
Lexington, MA 02421-6426

8. PERFORMING ORGANIZATION REPORT NUMBER

TIP-196

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Engineering Research Technical Initiative (MIT LL Technology Office /Lab funded)

10. SPONSOR/MONITOR'S ACRONYM(S)
MIT LL

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.
This material is based upon work supported by the Under Secretary of Defense for Research and Engineering under Air Force Contract No. FA8702-15-D-0001.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

MIT Lincoln Laboratory designs and fabricates airborne sensors to solve problems across a range of disciplines. These sensors are flown on Lincoln Laboratory Flight Test Facility aircraft. In order to certify airworthiness of an aircraft modification before it is flown, substantial analysis is performed. Of interest for this work is the aerodynamic performance of modifications made to the aircraft.
Aerodynamic simulation is performed in order to estimate quantities of interest for a given aircraft configuration. Some of these quantities include lift, drag, moments, wake properties, local flow directions, and vibrational profiles. Drag is a major contributor to fuel burn, maximum flight duration, and flight performance limitations. Other listed properties all contribute as well to the airworthiness determination of a particular aircraft configuration.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**	None	42	**19b. TELEPHONE NUMBER** *(include area code)*
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED			