Project Report
TIP-188

Rapid Aero Analysis through Deep Learning: FY22 Engineering Research Technical Investment Program

M.C. Jones
S. Kodali
A. McCourt

23 February 2023

Lincoln Laboratory
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LEXINGTON, MASSACHUSETTS

This material is based upon work supported by the Department of the Air Force 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 Aero Analysis through Deep Learning: FY22 Engineering Systems Technical Investment Program

M.C. Jones
S. Kodali
A. McCourt
Group 74

Project Report TIP-188

23 February 2023

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

This page intentionally left blank.

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS
(Continued)

LIST OF TABLES

This page intentionally left blank.

1. OVERVIEW

1.1 PROBLEM STATEMENT

MIT Lincoln Laboratory frequently designs and fabricates state-of-the-art airborne sensors to be flown on Flight Test Facility aircraft. During the design process of these sensors, aerodynamic performance must be evaluated to determine the best design and to establish the airworthiness of the aircraft modification. Quantities of interest often derive from in-flight aerodynamic forces such as: lift, drag, moments, shedding frequencies, and vibrational profiles. Of these quantities, aerodynamic drag is a commonly-evaluated design metric because of its impact on fuel burn, time on station, and flight speed and its correlation to undesirable vortex shedding. With present-day methodologies, high-fidelity analysis requires three major steps as shown in Figure 1: simplifying the geometric model to an approximate outer-mold shape, building the computational mesh, and running the high-fidelity fluid-flow simulation to extract the flow quantity of interest. The entire process requires days or weeks of work and careful expertise from the analyst. These time and resource commitments are a significant hurdle during the conceptual design phase when design parameters are rapidly changing and the opportunity for design impact is largest. The goal is to apply modern advances in machine learning in the prediction of aerodynamics to accelerate the burdensome analysis process.

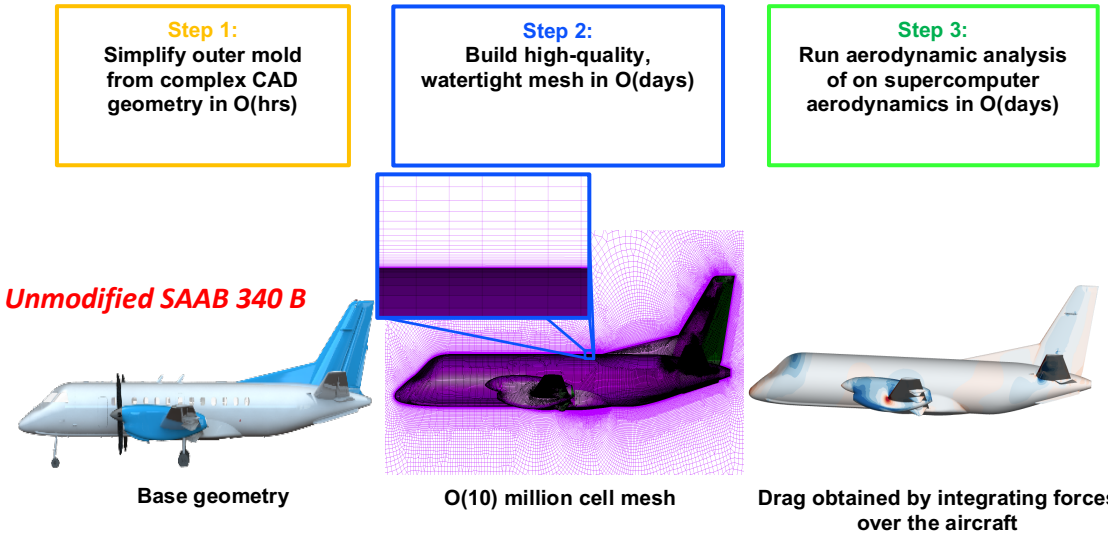


Figure 1. Conventional workflow for aerodynamic analysis is time and labor intensive.

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 as an initial proof of concept. 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 to drag prediction of pod-like shapes, and uncertainty in the estimates is moderate, so future work is planned to expand the scope of the tool to address those limitations.

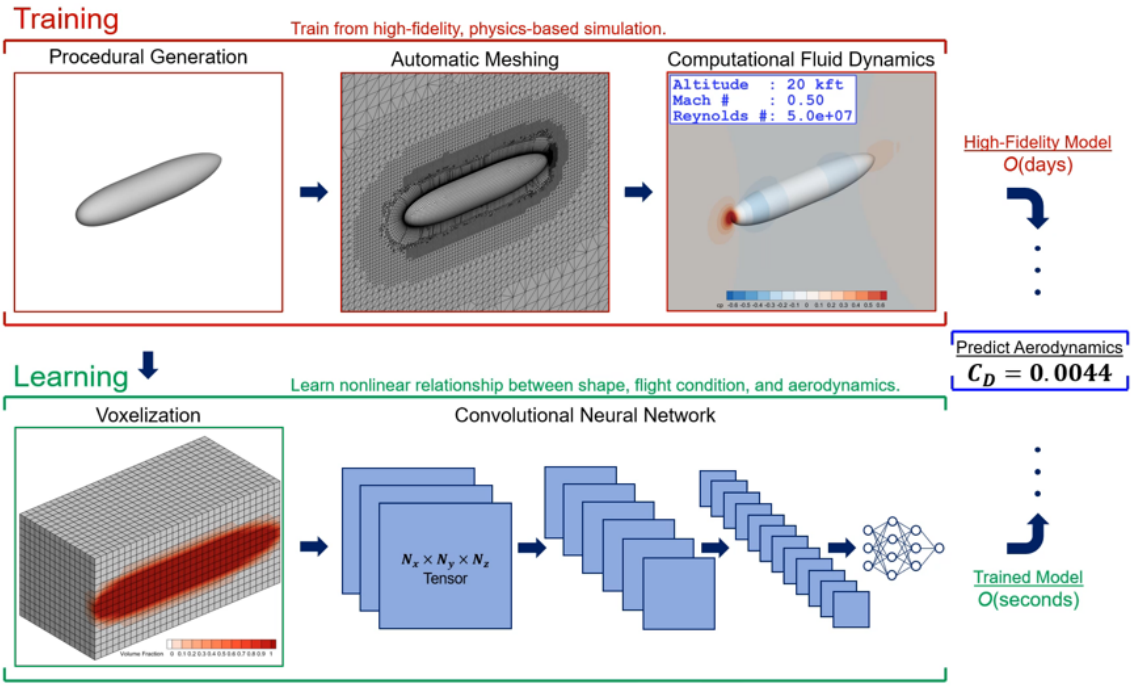


Figure 2. Illustration of how high-fidelity, fluid dynamics data trains the RAADL network in the prediction of aerodynamic drag for an arbitrary, pod-like shape.

1.3 MISSION IMPACT

This tool enhances Lincoln Laboratory's capacity to evaluate airborne sensors, aircraft modifications, and other aircraft concepts. The mission impact is summarized in Table 1. By expanding high-fidelity analytical capabilities to the realm of multidisciplinary design optimization, more-efficient designs can be identified and constructed, improving operational performance. Low-drag designs improve a sensor platform's time on station, fuel savings, and maximum speed, while generally minimizing undesirable vortex shedding. Additionally, this particular tool is just one piece of a larger puzzle. This approach can be expanded to evaluate additional aerodynamic quantities such as lift, moment, stability derivatives, and vibration spectra, or even additional 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

The algorithm employs a sub-category of machine learning, known as deep learning. The deep learning network is composed of Convolutional Neural Networks, 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]. The network diagram is shown in Figure 3. An arbitrary geometry is input as a "voxelized" 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 input concurrently in terms of altitude, Mach number, and Reynolds number. The flight condition is then concatenated with the latent variables, and then passed through a traditional series of linear neural network layers. The traditional linear layers output the aerodynamic quantities of interest, where here, we focus specifically on aerodynamic drag coefficient C_D. The network code was written using the popular Python machine learning framework known as PyTorch [2].

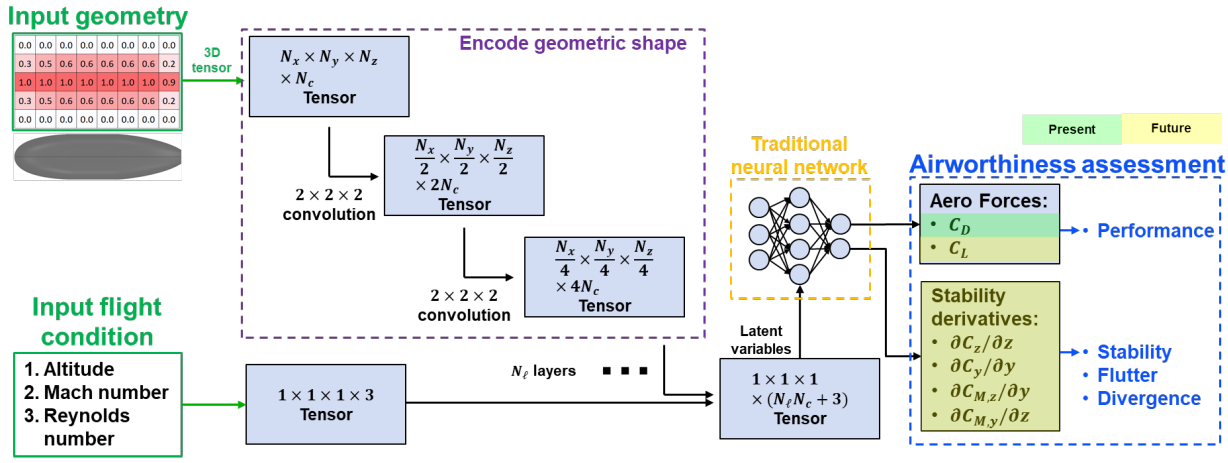


Figure 3. Network diagram: aerodynamic drag is predicted based on a 3D tensor of volume fractions representing the geometry and the flight condition.

Each layer in the network contains a collection of weights. When training the network, these weights are iteratively calibrated such that predictions from the network match with the training data. So, 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. Details of the convolutional layers are listed in Table 2, Table 3, and

Table 4 for input volume fraction tensor dimensions of 16x8x8, 32x16x16, and 64x32x32, respectively. 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 16x8x8

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	$2N_c$	$4N_c$	(4, 2, 2)	(2, 2, 2)	PReLU

Table 3

Convolutional Layer Details for Input Volume Fraction Tensor of Dimension 32x16x16

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	$4N_c$	$8N_c$	(4, 2, 2)	(2, 2, 2)	PReLU

Table 4

Convolutional Layer Details for Input Volume Fraction Tensor of Dimension 64x32x32

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	$8N_c$	$16N_c$	(4, 2, 2)	(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 5. 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 5

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

2.2 NETWORK ITERATIVE SETTINGS

Through the training process, weights are solved for iteratively using the Adam optimizer [3] 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% of a given voxel was occupied by the aircraft, that voxel tensor index would hold a value of 0.4. Figure 4 illustrates the voxelization of the Saab 340B aircraft using three different $I \times J \times K$ resolutions: 16x8x8, 32x16x16, and 64x32x32. 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.

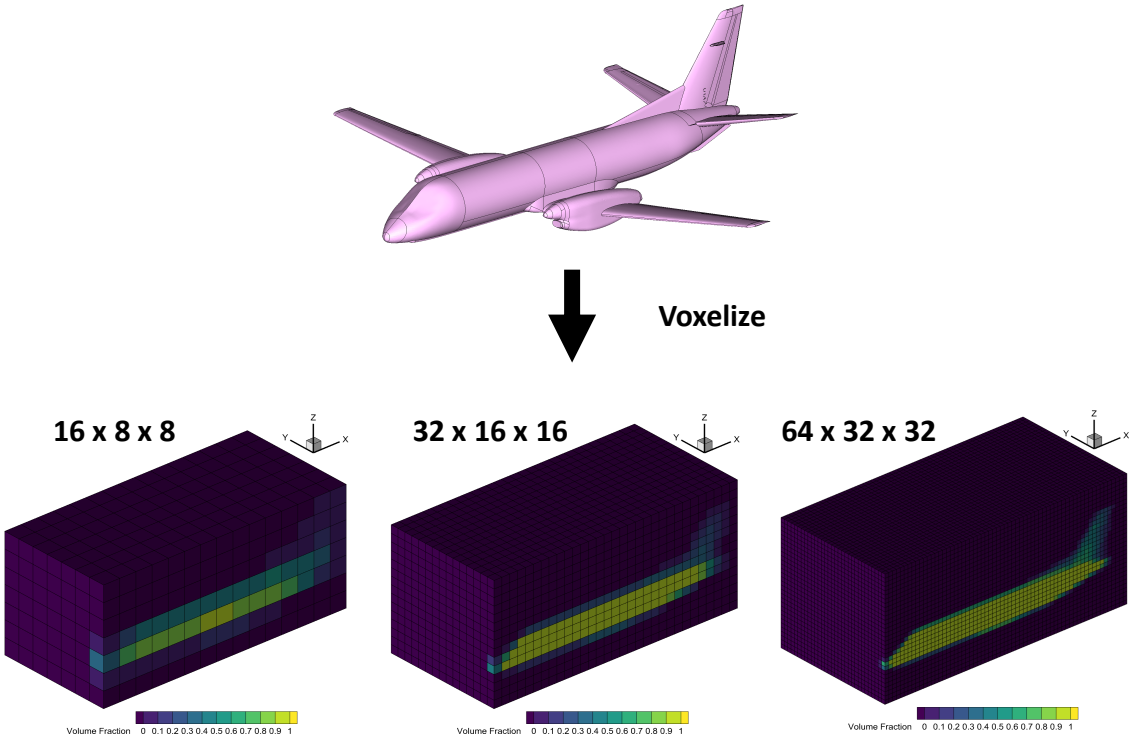


Figure 4. Voxelization of Saab 340B aircraft using three different resolutions: 16x8x8, 32x16x16, and 64x32x32.

For this study, we focus on "pod-like" shapes, that is, geometries that represent wing stores on an airplane. An illustration of the voxelization of a pod-like shape is shown in Figure 5. Adequate sizing of $I \times J \times K$ depends on the desired level of resolution. Since these geometries are much simpler than a complete aircraft, relatively coarser voxelizations can capture the overall geometric features. The exact resolution requirement for the desired level of detail on an arbitrary geometry is uncertain, and requires further study.

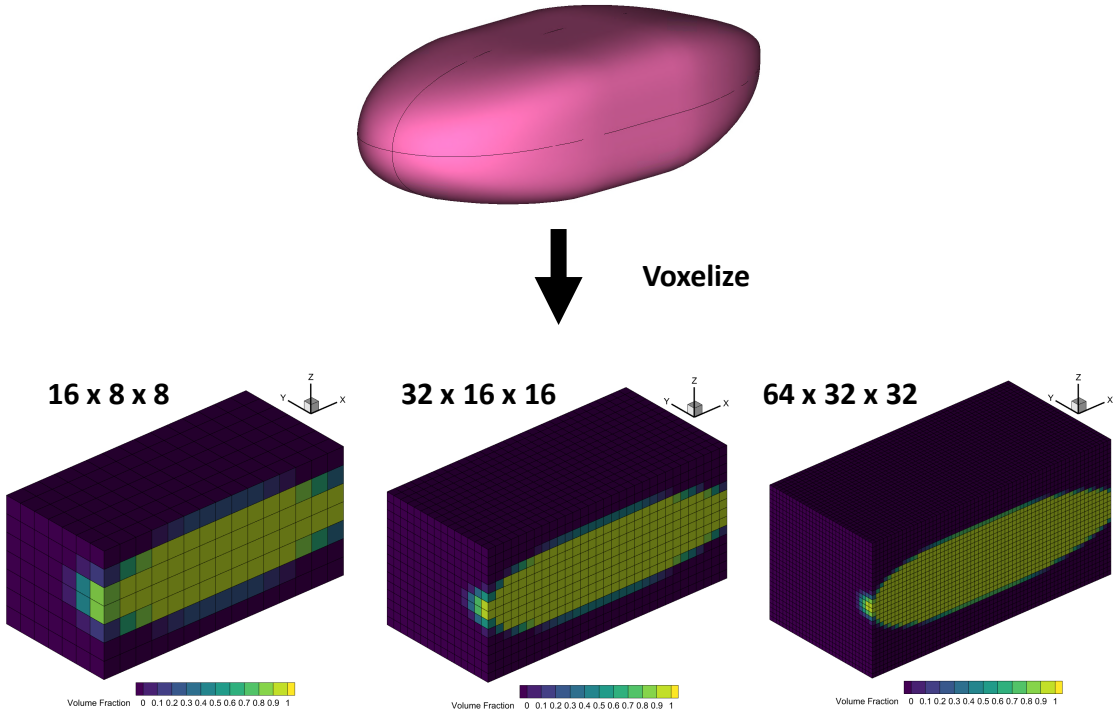


Figure 5. Voxelization of example pod-like shape for three different resolutions: 16x8x8, 32x16x16, and 64x32x32.

2.4 SOFTWARE DEVELOPMENT AND BEST PRACTICES

We employ a number of best practices in developing the software. First, we also follow the PEP8 Python style guide. This style guide helps to keep the code readable and follow a standardized convention across the Python community. Second, the code is version controlled using Lincoln Laboratory's Github. Version history is maintained and changes can be reviewed and merged in a rigorous manner. If unintended software bugs are introduced, it is easy to compare to previous, working version to identify and correct the issue. Third, we employ unit testing using the "pytest" framework [4]. Before each major code commit, we run a series of tests that verify that each piece of the code is running correctly. In doing so, we can identify bugs and correct them before major contributions are pushed to the central repository.

3. TRAINING THE MODEL

3.1 STEPS TO TRAIN THE MODEL

The model must be trained such that the underlying deep learning network learns the relationship between geometric shapes and their predicted aerodynamic properties. One fundamental challenge is finding a dataset that captures the diversity of possible geometric shapes with their associated aerodynamic properties over a wide range of flight conditions. Since such a dataset does not exist for general aircraft shapes, we must generate one ourselves. The primary steps involved in generating the training dataset and then training the model before running are listed in Figure 6. Each of the steps is automated, so that we can efficiently generate a large training dataset, train the model, evaluate, and easily tweak the workflow to improve the overall process. First, we generate numerous, varied geometries. Second, we build computational meshes for each of those geometries. Third, we generate Computational Fluid Dynamics (CFD) cases for each of the geometries over a variety of flight conditions. Fourth, we run the CFD fluid dynamics 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.

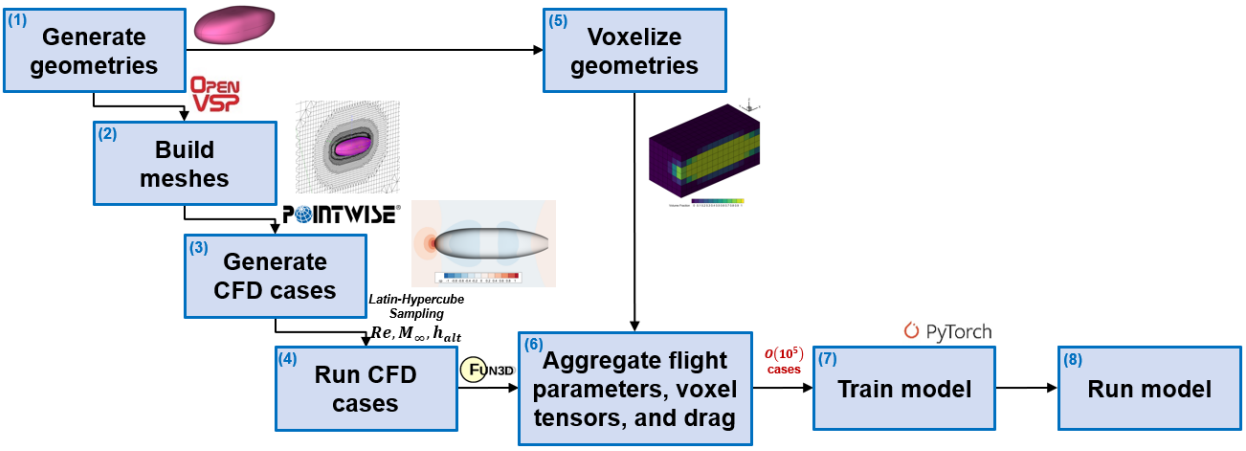


Figure 6. Steps to train the model.

3.2 GEOMETRY GENERATION

To generate a variety of geometric shapes for the model to train off of, we defined a parametric model, as shown in Figure 7 and randomly sampled the parameters when generating new shapes. Parameters were selected using Latin Hypercube Sampling. The scope of this study is limited to "pod-like" shapes, however, we intend to include additional aircraft lifting surfaces in future implementations. The geometric shape is divided into three sections: a forebody, a midbody, and an afterbody. The forebody has variable bluntness and a leading-edge point. The midbody maintains uniform cross-section. The afterbody has variable bluntness with a trailing edge that is either a point or a flat cross-section whose shape is of variable proportion to the center-body cross-section.

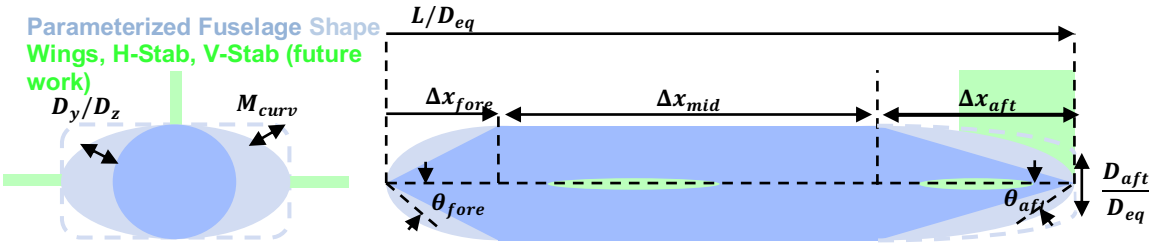


Figure 7. Parameterized aircraft geometry.

The geometric shape is fully defined by 8 parameters, as defined in Table 6. Practical bounds for each parameter are enforced in the generation algorithm. If an input parameter lies beyond the bounds, the parameter is changed to the bounding value. Sample range sometimes extends beyond these practical bounds with the understanding that the bounding value will be selected more often. This decision is intentional and helps to bias specific parameters that are more common in reality such as: a circular cross-section, an ellipsoidal cross-section, or a pointed trailing edge. For example, the sample range for M_{curve} lies between 0 and 5, however M_{curve} is bounded between 2 (circle) and 5 (rectangle). So, if a new geometry is generated for which M_{curve} falls somewhere between 0 and 2, then M_{curve} will be reset to $M_{curve} \equiv 2$, which will produce a circular cross section.

Table 6

Parameters to Define "Pod-Like" Geometric Shape

#	Symbol	Description	Enforced Bounds	Sample Range
1	L/D_{eq}	Ratio of length to equivalent diameter.	[2, 10]	"
2	$\Delta x_{mid}/L$	Ratio of midbody length to overall length.	[0.05, 0.5]	[0, 0.5]
3	$\Delta x_{fore}/\Delta x_{aft}$	Ratio of forebody length to afterbody length.	[0.3, 1]	"
4	$(\theta_{fore} - \theta_{min})/(\theta_{max} - \theta_{min})$	Forebody bluntness.	[0, 1]	"
5	$(\theta_{aft} - \theta_{min})/(\theta_{max} - \theta_{min})$	Afterbody bluntness.	[0, 1]	"
6	D_y/D_z	Ratio of transverse body dimension to vertical body dimension.	[1, 3]	[0, 3]
7	M_{curv}	Cross-sectional curvature: $M \equiv 2$ for a circle and $M \equiv 5$ for a rectangle.	[2, 5]	[0, 5]
8	D_{aft}/D_{mean}	Ratio of afterbody equivalent diameter to full body equivalent diameter.	[0.05, 5]	[-0.5, 5]

In addition to generating geometries based on random sampling of a parametric model, we examined realistic pod geometries. The realistic pod geometries are displayed in Figure 8 and labeled by alias names. These shapes include details beyond the scope of the parametric model such as bump-outs, scoops, cavities, strong-backs, and sharp transitions along the outer mold line. By evaluating RAADL predictions for these geometries, we can consider model effectiveness for real shapes, given that the predictions are made for geometries *not included in the training dataset*. Alternatively, if we choose to include a subset of these shapes in the training dataset, they allow the model to learn more-complex latent variables and make more-accurate predictions.

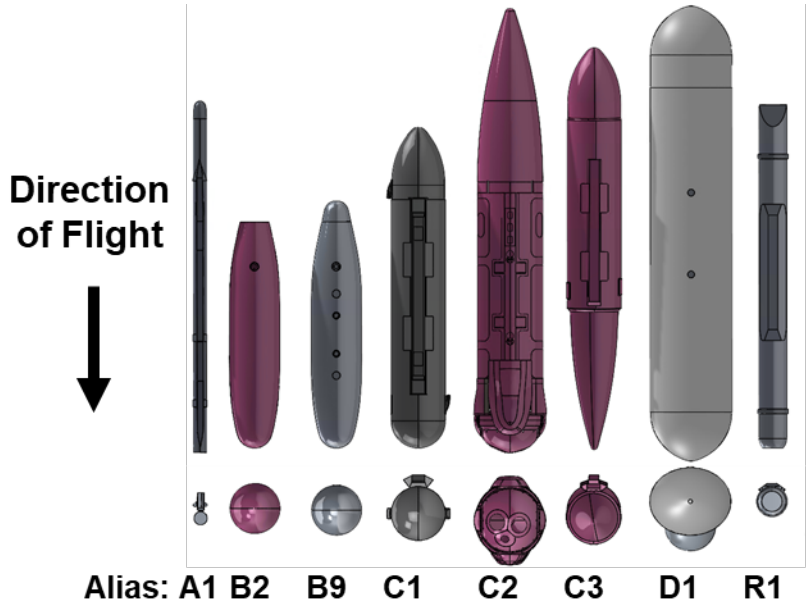


Figure 8. Realistic pod shapes.

3.3 AUTOMATED MESHING

3.3.1 Mesh Generation Algorithm

In generating datasets from the parametric geometry and the realistic pod shapes, a computational mesh must be built as an input to the fluid dynamics simulation. Since we are training off of potentially tens of thousands of geometries, the meshing process must be fully automated. We developed an automated meshing code that utilizes the Pointwise software [5], which is available on the Lincoln Laboratory GitHub. This code may either be run from a graphical user interface or directly from a system call. The meshing process is illustrated in Figure 9. The code builds a high-quality computational mesh for an arbitrary watertight CAD geometry. A large sphere forms the far field of the "O-mesh." Cells are refined around the features of interest of the geometry in the center of the O-mesh. The level of refinement was calibrated such that further refinement to the mesh did not significantly change predicted aerodynamics.

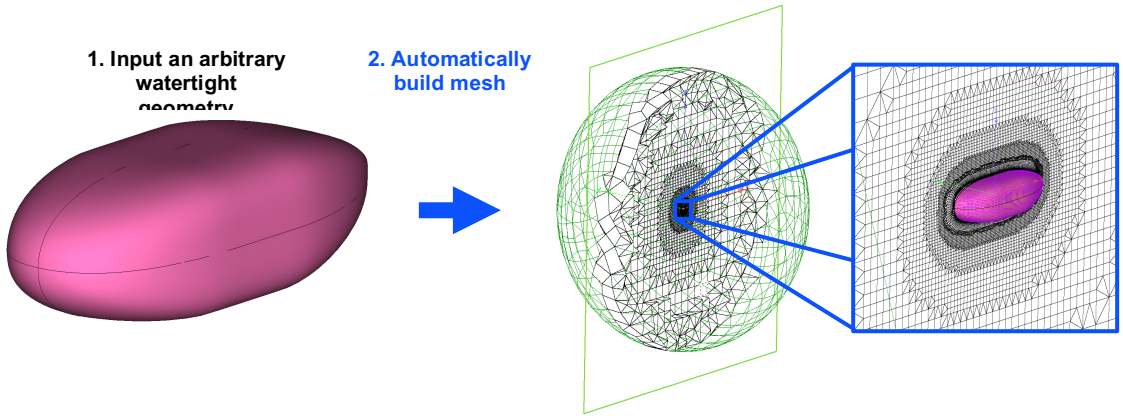


Figure 9. Automatic mesh generation process.

3.3.2 Resolving the Boundary Layer

Special attention is given to the refinement of cells within the airborne geometry's boundary layer, that is, the thin region where velocity grows from zero on the surface to the freestream speed of the surrounding environment. Proper resolution of the boundary layer is essential to accurately predicting skin friction drag. 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.

To verify this requirement, we surveyed 100 randomly generated geometries in the bounding high-altitude, high-Mach number, and high-Reynolds number flight condition: 50 kft, $M = 0.9$, and $Re_L = 10^8$. The y^+ quantity was measured at five locations along the body. Table 7 lists the results. Of all of the geometries, #85, #68, #48, and #57 demonstrated the largest y^+ in the survey. For all of these maxima, $y^+ < 1$, showing that the viscous sublayer is resolved.

Table 7

Sampled y+ Maxima over a Survey of 100 Randomly Generated Geometries for the Bounding Flight Condition

Sampling Location $[x/L]$	Maximum y^+	Case #
0.1	0.84	85
0.3	0.87	68
0.5	0.84	48
0.7	0.82	57
0.9	0.79	57

A visualization of y^+ for Case #57 is shown in Figure 10. Through inspection, we see that $y^+ < 1$ over the entire shape. Further visualization of Case #57 is shown in Figure 11. Mach number colors a vertical slice through the computational domain. Zooming in to the smallest features of the boundary layer, we can see adequate cell resolution to capture the velocity gradient coming off of the skin.

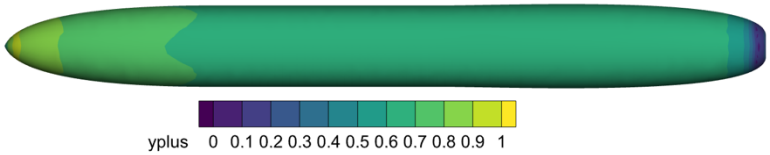


Figure 10. Visualization of y+ for Diagnostic Case #57, the least-favorable geometry, shows that boundary layer is resolved down to the viscous sublayer even for the worst-case, bounding flight condition.

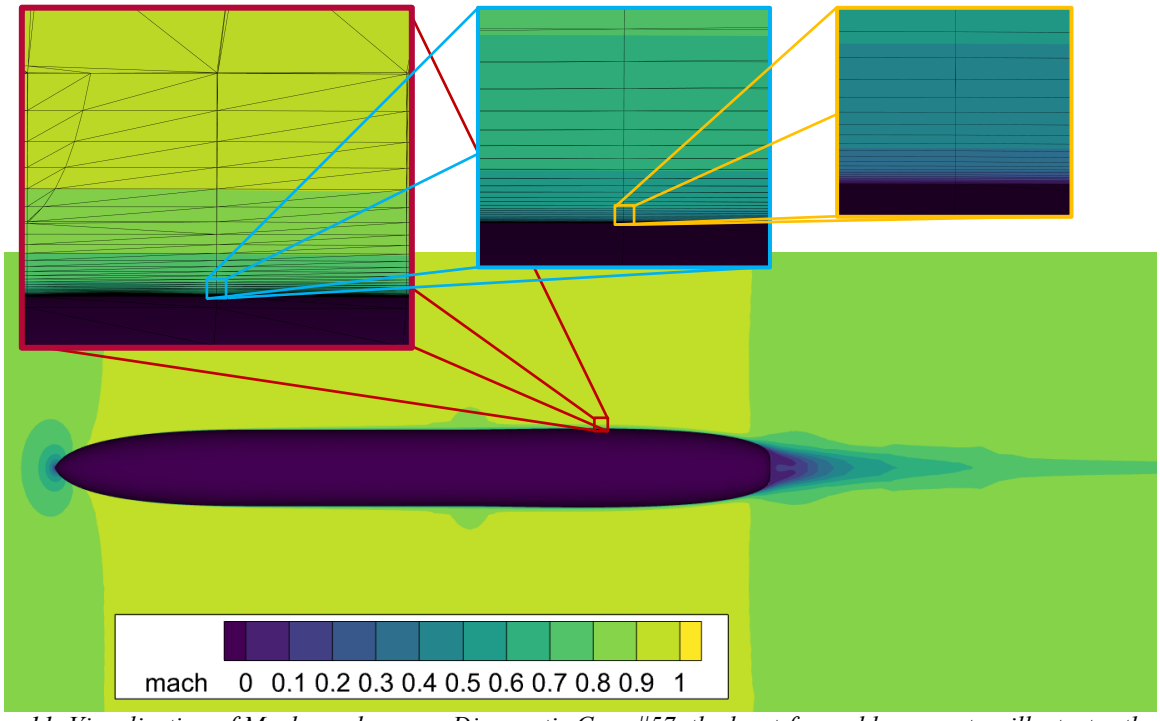


Figure 11. Visualization of Mach number over Diagnostic Case #57, the least-favorable geometry, illustrates that a sufficient number of cells are present to resolve the boundary layer even for the worst-case, bounding flight condition.

3.4 GENERATING COMPUTATIONAL FLUID DYNAMICS CASES

Similar to the geometry generation, we employ Latin Hypercube Sampling to sample flight conditions in the generation of Computational Fluid Dynamics (CFD) cases. For a given generated geometry, any number of CFD cases can be generated. The desired number of CFD cases for a given geometry is an area of study. We bound the flight conditions based on practical limits and the scope of this effort. The flight condition is defined by altitude, Mach number, and Reynolds number. Table 8 lists the imposed lower and upper bounds. Here, the upper bounds are based on the limiting flight conditions experienced by wing stores flown on a Gulfstream IV aircraft.

Table 8

Flight Condition Bounds

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

3.5 FLUID DYNAMICS SIMULATION

3.5.1 Numerical Approach

Details of the Computational Fluid Dynamics (CFD) simulation are important because the simulated results represent the "ground truth" for the deep learning model. Based on the input flight condition and computational grid for a given geometry, we solve the Reynolds-Averaged Navier-Stokes (RANS) mass, momentum and energy equations through the NASA FUN3D simulation framework [6]. Turbulence is modeled using the Spalart-Allmaras model [7] with a freestream turbulence intensity of 3%. We employ the van Albada flux limiter with the low-diffusion flux splitting "LDFSS" flux construction method [8]. All simulations were conducted using Lincoln Laboratory Super Computing resources.

3.5.2 Iterative Convergence

Simulations are run until the governing mass, momentum, energy, and turbulence equations are well converged, that is, when relative residuals fall below 10^{-8}. Poorly converged simulations are discarded from the training data. Figure 12 shows residual convergence for two example cases. As shown, residuals are eventually driven below 10^{-8}. At iteration 5000, flux limiters are frozen to aid in convergence.

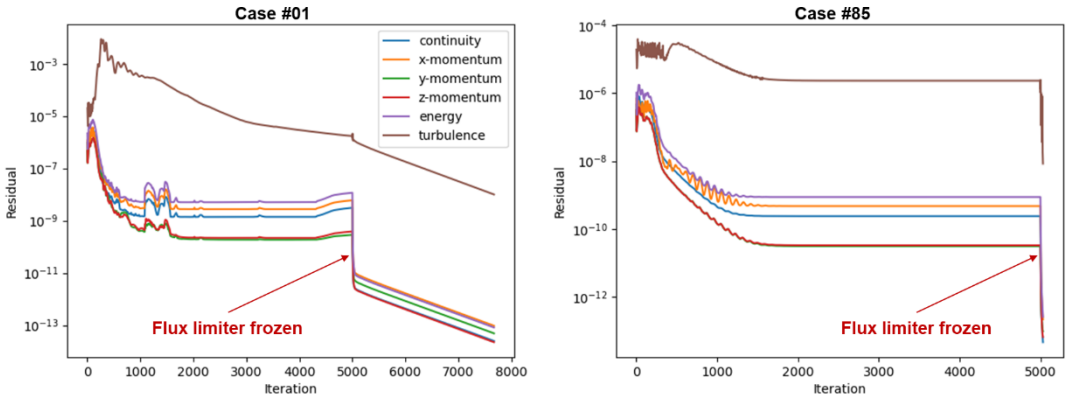


Figure 12. Residual convergence for example cases #01 and #85.

3.5.3 Example Fluid Flow Visualization

Figure 13 visualizes the flow field around an example geometry. The outer mold line of the geometry and a vertical cutting plane are colored by pressure coefficient, C_P. Red regions indicate relatively higher pressure, such as regions of stagnation. Blue regions indicate relatively lower pressure. The pressure gradient near the leading edge of the body indicates flow acceleration. An adverse pressure gradient is visible near the trailing edge of the body. Integrating pressure and skin friction forces around the body yields the aerodynamic loads on the body, such as drag.

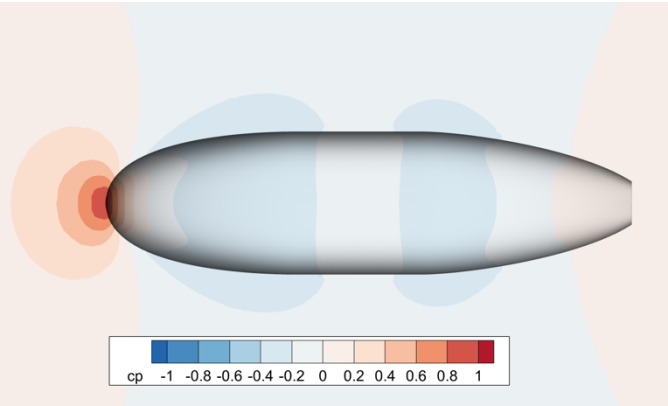


Figure 13. Example visualization of pressure coefficient over a pod-shape training data case.

3.6 TRAINING THE MODEL

After the CFD simulations are conducted and aerodynamic data is aggregated with the voxelized representation of the respective geometry, the dataset can be used to train the deep learning model. This is an iterative process in each model weights in the Convolutional Neural Network and linear layers are iteratively calibrated such that model predictions match the training data. Figure 14 shows an example iterative history in the training of a case with 16x8x8 voxelization and 1000 training geometries with 1 flight case for each geometry. The loss, which is Mean Square Error, is driven down exponentially with the assistance of an automatically adjusting learning rate. Additional details of the network are listed in Sections 2.1 and 2.2.

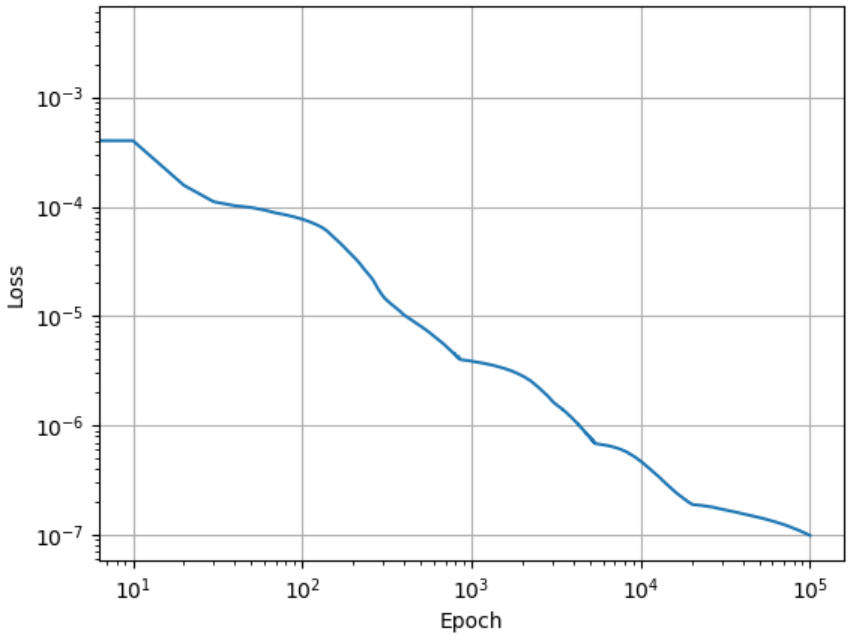


Figure 14. Example training history for 16x8x8 case with 1000 training geometries with 1 flight case each.

4. MODEL PERFORMANCE

4.1 TRAINING TIME REQUIREMENTS

Training time requirements are important to understand because they limit the overall size, and consequently the diversity, of the training dataset. Figure 15 shows the training time required for meshing, fluid dynamics simulation, and deep learning as a function of the total number of training cases assuming one simulation per geometry. Each step makes use of Xeon-g6 computational nodes. The fluid dynamics simulation makes use of GPU-acceleration and 40 concurrent Xeon-g6 nodes to run 40 separate simulations simultaneously. GPU-acceleration was not implemented for the deep learning training, but is desired as future work.

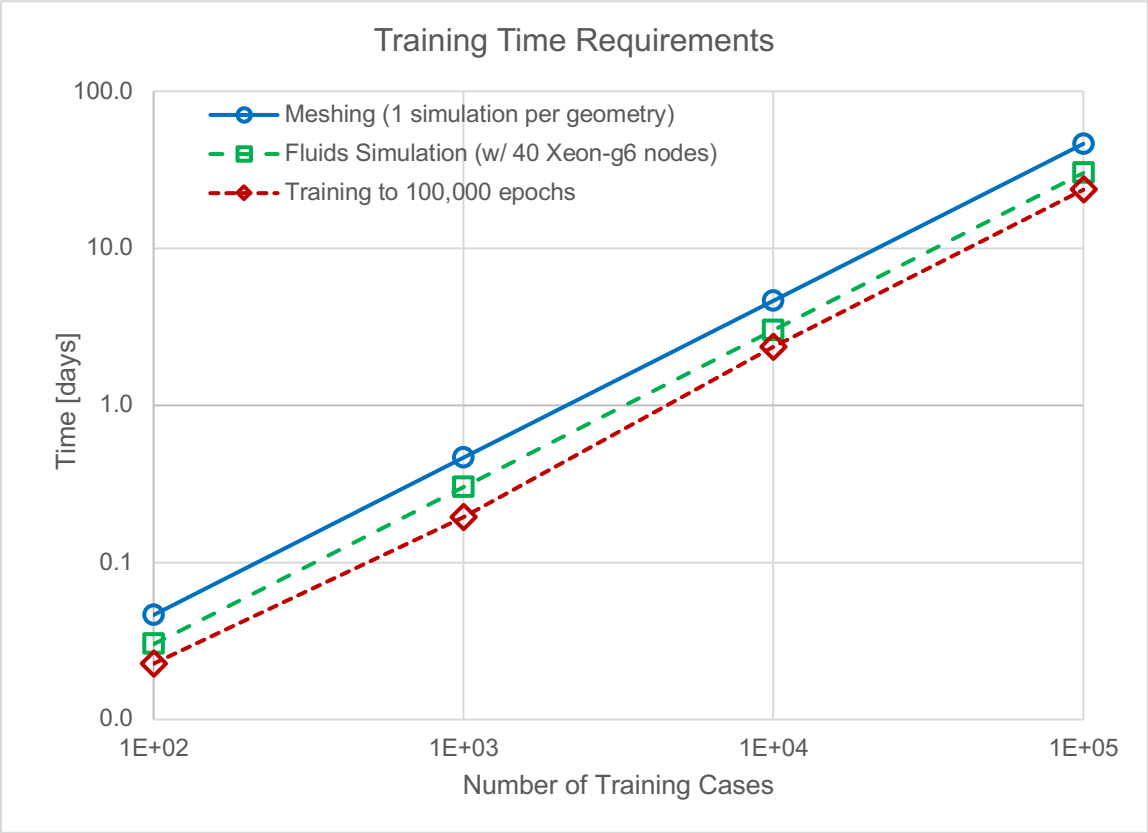


Figure 15. Meshing, fluids simulation, and model training run time requirements vs. total number of training cases.

Figure 15 shows that for just 1000 training cases, the total time required is less than a day, however, these time requirements increase exponentially. For 100,000 training cases, the meshing time requirement is approximately 46 days, the simulation is 30 days, and the deep learning training is 24 days. In total, this many cases requires approximately three months to complete, which is prohibitively expensive for model calibration exercises and quick-turnaround studies. Training dataset sizes on the order of 10,000 are much more tenable, taking approximately ten days to complete. The soft, upper limit on practical training dataset sizes is between 10,000 and 100,000 with the present implementation and hardware.

4.2 MODLE PREDICTION RUN TIME

After the model is trained, it can quickly predict the aerodynamic properties of an arbitrary geometry. Model prediction run time is presented in Figure 16 as a function of the input geometry file size in megabytes. The relationship is presented for three different voxelization resolutions: 16x8x8, 32x64x64, and 64x32x32. For file sizes as small as 23 Mb, the run time is 1.3, 2.0, and 3.6 for 16x8x8, 32x64x64, and 64x32x32, respectively. Finer voxelization resolutions take more time to compute. For file sizes as large as 564 Mb, the run time is increased to 26, 28, and 34 seconds. Even so, these run times greatly accelerate the analysis process from one that previously took days or weeks, to one that is on the order of seconds.

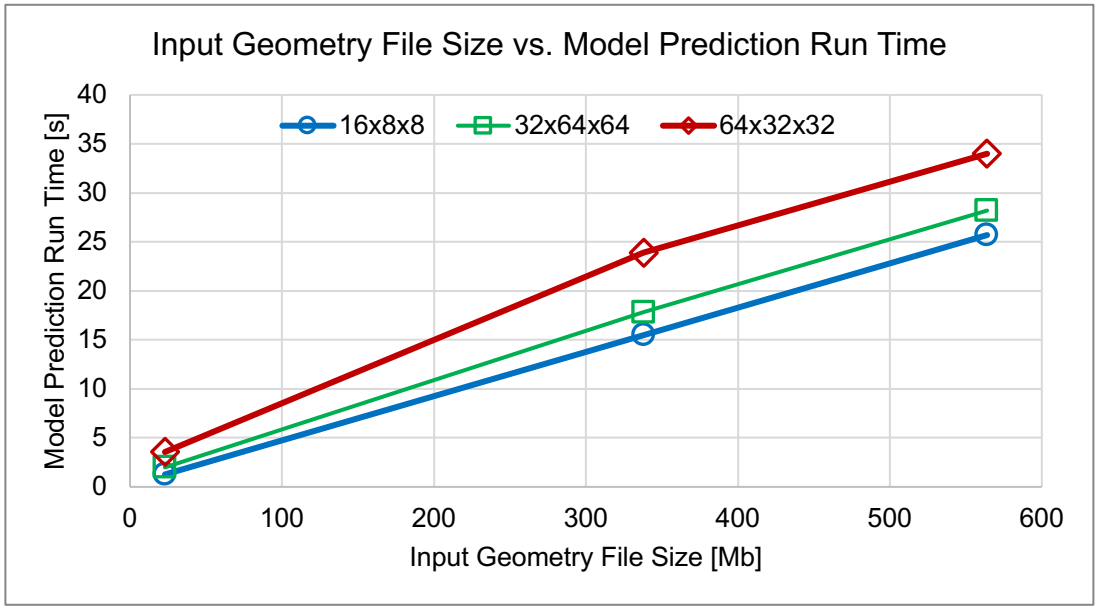


Figure 16. RAADL model prediction time vs. input geometry file size and voxelization resolution.

4.3 MODEL UNCERTAINTY

Alongside run time requirements, model predictive accuracy determines the utility of the RAADL tool. Uncertainty is evaluated, first, by predicting the aerodynamic drag for 1000 test cases that are *not included in the training data*; the model has never seen these geometries before. These drag predictions are compared to the "ground truth" Computational Fluid Dynamics (CFD) results for those cases and the Mean Absolute Error (MAE) is computed for the aggregated 1000 test cases.

Table 9 examines the relationship between MAE and the number of training geometries and the number of CFD cases. The voxelization resolution is set to 16x8x8. With only 100 geometries and 100 CFD cases (1 case per geometry), the percentage MAE is 32%. Increasing the number of CFD cases to 1000 and to 10,000 (10 and 100 cases per geometry) results in a small reduction in MAE. Increasing the number of training geometries has a stronger effect, however. With 10,000 geometries and 1 CFD case per geometry, we see a MAE of 14%. This result shows that it is more important to increase the diversity of geometries than to increase the diversity of the flight conditions.

Table 9

Model Accuracy Based on Percentage Mean Absolute Error in the Aerodynamic Drag Prediction of 1000 Test Cases That the Model Has Never Seen Before

1000 Test Cases Mean Abs. Error [%]		# Training Geometries 100	1000	10000
# Training CFD Cases	100	31.8		
	1000	31.2	21.8	
	10000	30.5	17.9	14.0

Similar results are seen when examining the standard deviation of the percentage MAE, as shown in Table 10. The standard deviation decreases as the number of training geometries and fluid dynamics simulations are increased. Note, that there is variability in the results. Initialization of training weights is random, so the quality of a training run will vary when run multiple times. For each comparison, several training runs were conducted and the best performing model was recorded while the poorer models were discarded.

Table 10

Standard Deviation of Percentage Mean Absolute Error Based on the Number of Training Geometries and the Number of Fluid Dynamics Simulations

1000 Test Cases Error Stand. Dev. [%]		# Training Geometries 100	1000	10000
# Training CFD Cases	100	55.6		
	1000	72.3	45.0	
	10000	81.1	33.0	25.9

While the model shows a predictive uncertainty of 14% for the 1000 generated parametric geometries, it is also important to consider realistic pod shapes. Table 11 shows performance for the realistic geometries previously presented in Figure 8. Predictions are made for 100 different flight conditions for each of the geometries. Error is much larger, on the order of 55% for all permutations of training geometries and CFD cases. This poor performance is explained by the lack of appropriate features in the training dataset. None of the generated pod shapes from the parametric model include the distinct features of the real pods such as bump-outs, scoops, cavities, strong-backs, and sharp transitions. The deep learning model cannot appropriately predict the aerodynamics of these features because they are not within the scope of the training dataset.

Table 11

Initial Evaluation of All Realistic Geometries, None of Which Have Been Included in the Training Data

All Real Geometries Mean Abs. Error [%]		# Training Geometries 100	1000	10000
# Training CFD Cases	100	57.6		
	1000	58.7	54.9	
	10000	54.6	57.1	54.6

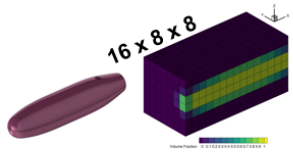
To account for the lack of additional realistic geometric features, the model is retrained using all of the real pods *except for* B2, and predictions are made for B2 in 100 flight conditions. The results are much more favorable, showing an uncertainty of 9.7% for the largest training dataset with 1 CFD case per geometry. MAE predictions for the other realistic geometries are on the order of 1–2%, which is unremarkable and expected, given that they are now present in the training dataset.

Table 12

Mean Absolute Error in the evaluation of just the B2 realistic geometry. All realistic geometries *except* B2 are included in the training data.

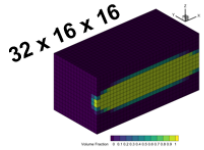
B2 Real Geometry Mean Abs. Error [%]		# Training Geometries (not including real geometries) 100	1000	10000
# Training CFD Cases*	100	18.6		
	1000	13.9	12.4	
	10000	11.2	12.0	9.7

A sensitivity study is conducted to evaluate desired voxelization resolution. Recall that the "voxel" tensor of volume fractions represents input geometries. Results are examined for three different levels of refinement: 16x8x8, 32x16x16, and 64x32x32, as shown in Figure 17. As before, the trained networks are evaluated over 1000 test cases to determine MAE. Once again, the results are "noisy" due to the inherent randomness derived from the initialization of the training process. Over all, there is not a strong correlation between resolution and performance. The coarsest resolution provides similar performance to the finest and seemingly has sufficient potential to capture and learn the features of interest in the training dataset.



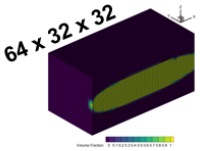
16x8x8, Test Cases Mean Abs. Error [%]		# Training Geometries* 100	1000	10000
# Training CFD Cases*	100	31.7		
	1000	28.9	21.9	
	10000	25.0	20.1	13.8

16x8x8, B2 Geometry Mean Abs. Error [%]		# Training Geometries* 100	1000	10000
# Training CFD Cases*	100	18.6		
	1000	13.9	12.4	
	10000	11.2	12.0	9.7



32x16x16, Test Cases Mean Abs. Error [%]		# Training Geometries* 100	1000	10000
# Training CFD Cases*	100	30.2		
	1000	25.7	22.5	
	10000	30.5	18.4	13.2

32x16x16, B2 Geometry Mean Abs. Error [%]		# Training Geometries* 100	1000	10000
# Training CFD Cases*	100	18.3		
	1000	19.8	17.9	
	10000	11.2	7.6	12.9



64x32x32, Test Cases Mean Abs. Error [%]		# Training Geometries* 100	1000	10000
# Training CFD Cases	100	32.5		
	1000	32.2	20.4	
	10000	22.8	21.3	13.2

64x32x32, B2 Geometry Mean Abs. Error [%]		# Training Geometries* 100	1000	10000
# Training CFD Cases	100	16.4		
	1000	15.6	12.2	
	10000	13.2	10.7	14.3

*Also trained on each of the realistic geometries *except* for B2, 100 CFD cases each.

Figure 17. Voxelization refinement sensitivity study. Mean Absolute Error is shown for the 1000 test cases and also for the B2 geometry for three levels of refinement: 16x8x8, 32x16x16, and 64x32x32.

Although predictions for a full aircraft with wing and tail surfaces are beyond the scope of the training data, accurate prediction of these aircraft is an eventual goal for future implementations of the model. Here, we evaluate the prediction for a full Saab 340B aircraft, presented previously in Figure 4, in order to gauge

the level of uncertainty for *geometries completely beyond the scope of the training data*. Table 13 shows the percentage error in the prediction of aerodynamic drag for a flight condition of 150 knots-calibrated airspeed at 24 kft. Drag is underestimated by a factor of up to, approximately, 50%. In order to improve performance here, it is necessary to include full aircraft in the training data.

Table 13

Percentage error is examined for the full Saab 340B aircraft, even though no aircraft shapes were included in the training data. Error is large as expected. Real aircraft shapes must be trained on to make accurate predictions.

Voxelization Resolution, IxJxK	% Error
16x8x8	-32
32x16x16	-24
64x32x32	-48

4.4 SOURCES OF UNCERTAINTY

Given the moderate to large uncertainty of model predictions, it is important to understand sources of uncertainty so as to best reduce them in future work. Sources are listed in Table 14. The quality of the training data is paramount because all predictions are fundamentally derived from that data. Using the voxelization representation of geometries could filter out important features of the aeroshape, so adequate resolution should be reevaluated when the scope of the aircraft geometries is expanded. Finally, traditional machine learning sources of uncertainty pose an ever-present challenge. We propose several future work thrusts to address and reduce these uncertainties and improve model accuracy.

Table 14

Sources of Uncertainty in Overall Modeling Approach

Source of Uncertainty	Explanation
Training Data Error	Fluid dynamics simulations provide the "ground truth" aerodynamics of training data geometries. Predicted aerodynamics can only be as accurate as these results. Errors include: discretization error, modeling error, iterative error, and roundoff error.
Voxelization Representation	The voxelized representation divides the 3D geometry into finite voxel cells – each with a corresponding volume fraction. Aircraft features smaller than the voxels are not uniquely resolved. Furthermore, the actual estimation of volume fraction has a small error.
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.

4.5 FUTURE WORK

To address uncertainty in the prediction of aerodynamics, we seek to bound aerodynamic predictions by the physical governing mass, momentum, and energy equations. These bounds will enforce realistic predictions. This addition to the framework, however, is ambitious. The model will be reworked based on advances in physics-informed learning [9–11]. Instead of predicting post-processed aerodynamic quantities, we seek to predict the fluid flow itself, and then derive the desired aerodynamic properties from that flow field. Figure 18 shows the future model extended to predict a full fluid flow for a given geometry. After the flow is predicted, traditional post-processing is used to determine quantities for establishing airworthiness.

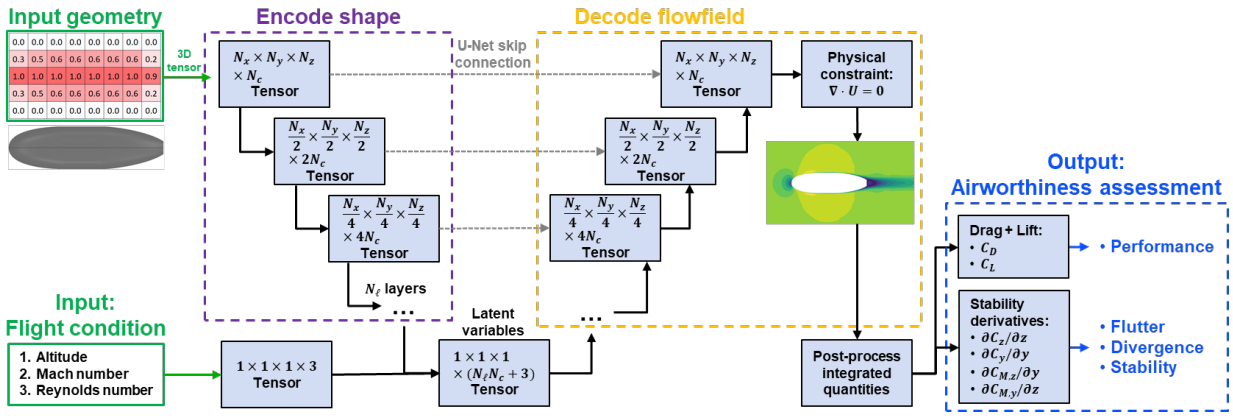


Figure 18. Proposed, future network diagram: the model predicts the flow, itself, and is bounded by physical constraints.

Another future desire is a method to quantify how similar the input geometry is to the geometries of the training data. The goal is to quantify the uncertainty derived from the model's unfamiliarity with the input geometry. We will use statistical methods to develop a confidence score for a given geometry.

The final goal is to continually expand the scope of the predictive capabilities. In the short term, we hope to predict additional airworthiness criteria for a wider variety of aeroshapes. In the long term, we hope to extend these deep learning techniques to additional disciplines such as structural and thermal analysis. The tentative future work roadmap is shown in Figure 19.

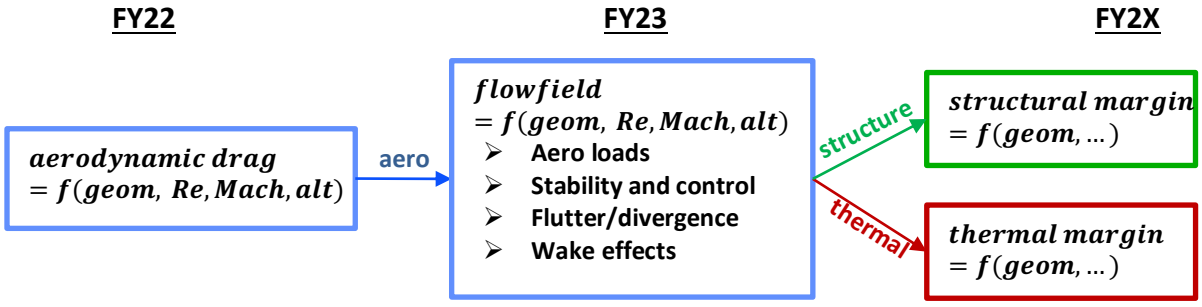


Figure 19. Future work roadmap.

5. CONCLUSIONS

We developed a tool that can predict aerodynamic properties of arbitrary geometries. As a first pass, the model is bounded to the prediction of aerodynamic drag of pod-like shapes. We developed fully-automated methods to generate a large training dataset of tens of thousands of training cases. The methods include an automated meshing routine that has been used extensively even for outside efforts. We trained the model off of various sizes and resolutions of training data to evaluate performance. In the future, we will extend predictive capabilities to additional aeroshapes and additional flow quantities important to establishing airworthiness. We hope to predict the fluid flow itself and bound predictions by the fluid-flow governing equations to improve realism.

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

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

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

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

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

[7] 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.

[8] 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.

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

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

[11] 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.

This page intentionally left blank.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
02-23-2023	Project Report	

4. TITLE AND SUBTITLE

Rapid Aero Analysis through Deep Learning:
FY22 Engineering Research Technical Investment Program

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Matthew Jones, Suhas Kodali, Anthony McCourt

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-188

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

MIT Lincoln Laboratory's Engineering Research Technical Initiative
MIT Lincoln Laboratory
244 Wood Street
Lexington, MA 02421-6426

U.S. Department of the Air Force
The Pentagon, Arlington County, VA

10. SPONSOR/MONITOR'S ACRONYM(S)

MIT LL and DAF

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 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.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

MIT Lincoln Laboratory frequently designs and fabricates state-of-the-art airborne sensors to be flown on Flight Test Facility aircraft. During the design process of these sensors, aerodynamic performance must be evaluated to determine the best design and to establish the airworthiness of the aircraft modification. Quantities of interest often derive from in-flight aerodynamic forces such as: lift, drag, moments, shedding frequencies, and vibrational profiles. Of these quantities, aerodynamic drag is a commonly-evaluated design metric because of its impact on fuel burn, time on station, and flight speed and its correlation to undesirable vortex shedding. With present-day methodologies, high-fidelity analysis requires three major steps as shown in Figure 1: simplifying the geometric model to an approximate outermold shape, building the computational mesh, and running the high-fidelity fluid-flow simulation to extract the flow quantity of interest. The entire process requires days or weeks of work and careful expertise from the analyst. These time and resource commitments are a significant hurdle during the conceptual design phase when design parameters are rapidly changing and the opportunity for design impact is largest. The goal is to apply modern advances in machine learning in the prediction of aerodynamics to accelerate the burdensome analysis process.

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	31	19b. TELEPHONE NUMBER *(include area code)*
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED			