NEXT-GENERATION AEROTHERMODYNAMIC MODELING FOR SPACE DEBRIS USING DEEP LEARNING

Nicholas Sia[(1)] **and Piyush M. Mehta**[(2)]

[(1)] *West Virginia University, Morgantown, USA, Email: nisia@mix.wvu.edu*
[(2)] *West Virginia University, Morgantown, USA, Email: piyush.mehta@mail.wvu.edu*

ABSTRACT

The number of resident space objects re-entering the atmosphere is expected to rise with increased space activity over recent years and future projections. Predicting the probable survival and impact location of the medium to large sized re-entering objects becomes important as they can cause on ground casualties and damage to property. Uncertainties associated with the re-entry makes necessary an expensive probabilistic approach. To date, object-oriented analysis is the dominant tool used for atmospheric re-entry modeling and simulation, where aerothermodynamic coefficients are used to determine the risk a re-entering object poses to the ground through the use of analytical formulations. Close form solutions are limited to convex objects in the free molecular and continuum flow regime and stagnation point estimates. In the transition regime (75-150 km), a combination of bridging and shape functions are used for the different primitive objects. In this work; we leverage the power of deep learning to develop next-generation models for the aerothermodynamic modeling (drag coefficient and full body heating distributions) in the transition flow regime for both convex and concave primitive shapes (sphere, cube, and cylinder). The increasing LEO population puts more stress on NASA's recommended probability and makes this a timely contribution.

Keywords: Re-entry; Aerothermodynamics; Space Debris; Deep Learning.

1. INTRODUCTION

Rapidly growing technological demands is an important driver of a very quickly expanding interest from both public and private sources in the satellite market. With tens of thousands of satellite launches planned in the next decade combined with the existing volume of manmade objects and debris already in space, the area in Low Earth Orbit (LEO) is expected to become considerably more crowded. As these objects are launched and eventually re-enter Earth's atmosphere, they are expected to not exceed a NASA-established 1:10,000 probability of survival upon re-entry. Atmospheric re-entry analysis is a critical component to mission planning in order to ensure that the end of a spacecraft's mission is properly planned. Specifically, it is of upmost importance to ensure that any re-entering spacecraft or debris do not pose any threat to civilians, buildings, or populated areas on the ground. As the volume of spacecraft and other space assets in the Low-Earth Orbit (LEO) environment increases, it is expected that space will be considerably more crowded in the next decade and beyond. Per requirements and recommendations outlined by NASA's Orbital Debris Program Office (ODPO) to mitigate further pollution of the space environment, satellites are recommended to re-enter Earth's atmosphere within 25 years of the end of its life. Upon re-entry, spacecraft must not pose a greater than 1:10,000 risk of striking any significant objects or causing any civilian casualties on the ground [1].

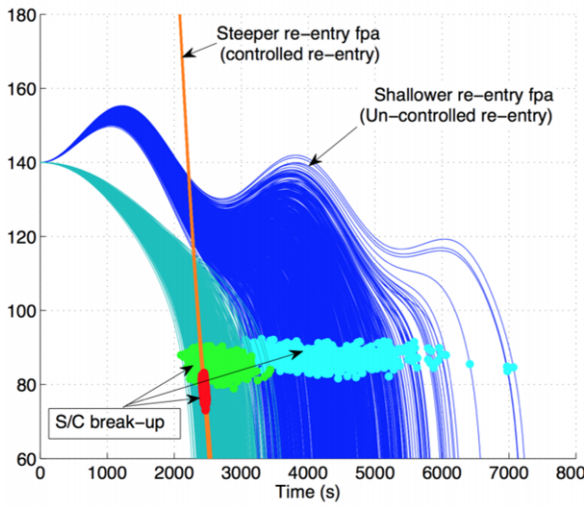


Figure 1. Monte Carlo analysis and the concept behind an object-oriented tool [3]. fpa = flight-path angle

To date, the predominant analysis tools for re-entry assessment are object-oriented codes such as NASA's ORSAT, CNES' DEBRISK, and ESA's SARA. These tools assume a trajectory in which a spacecraft will fragment into multiple shape primitives at a certain altitude, and the following risk analysis is performed on these simple objects. Conversely, spacecraft-oriented tools such as ESA's

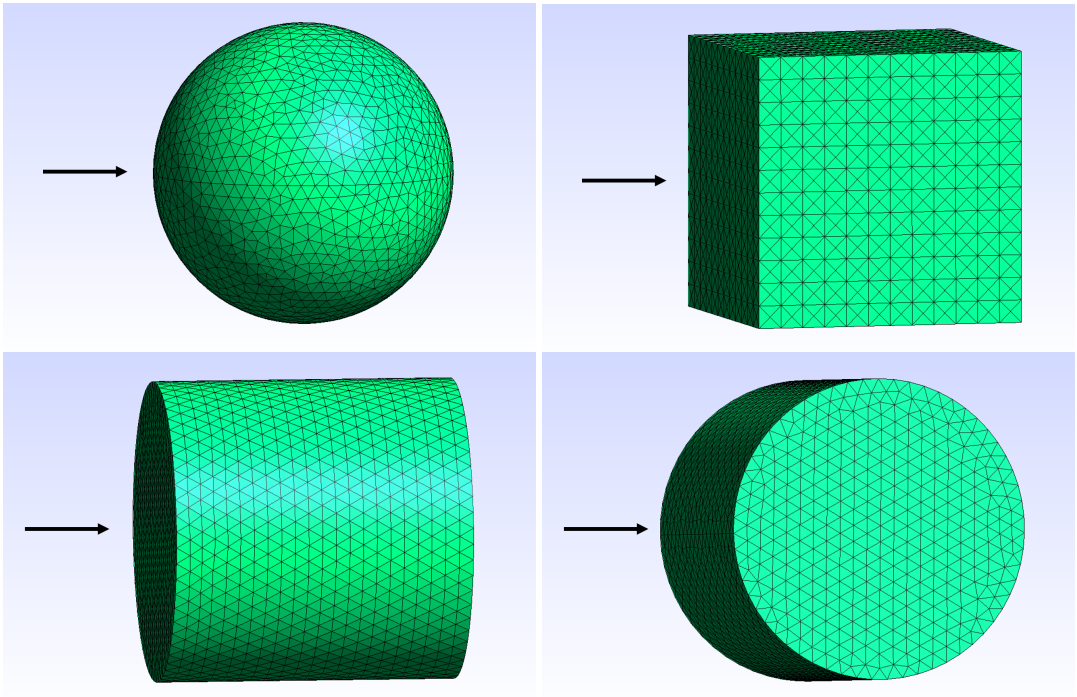


Figure 2. Flow Directions for Shape Primitives

SCARAB, simulate the entire spacecraft until demise to determine its risk. Both types of tools determine drag and heating coefficients for risk assessment through the use of semi-empirical and analytical formulations. For finite nose radius objects such as a sphere, closed form solutions exist for stagnation point heating. The opposite is true for nonconvex shapes such cubes, cylinders, and other objects with sharp edges. For these geometries, some sort of estimation is achieved with the use of shape factors or bridging functions. These models are quickly evaluated, but have compounding uncertainties with increased evaluations that are not well-defined. Significant efforts have led to noteworthy improvements in our ability to modeling heating distributions; however a lot remains to be achieved, especially for non-convex objects such as cubes and cylinders [2].

High-fidelity modeling tools such as Direct Simulation Monte-Carlo (DSMC) codes can provide numerical estimates for heating and drag distributions over shape primitives given certain flow conditions. The main drawback to these tools is computational expense (where analytical formulations excel): The computational resources needed to numerically solve particle motion and surface interactions scales exponentially with increasing atmospheric density.

Re-entry flow is typically classified into one of three flow regimes: Continuum, transition, and free molecular. The bounds of each flow regime are characterized by a specific Knudsen Number, which defines the kinetic distance between gas particles compared to some characteristic length. In free molecular flow, the Knudsen Number is sufficiently high such that intermolecular collisions are improbable. In this flow regime, where heating is not much of a concern, it is possible to use high-fidelity techniques such as DSMC with full-scale spacecraft geometries without much computational expense. While transition flow generally corresponds to a denser accumulation of gas particles, DSMC is still valid here, albeit requiring more computational resources. However, in the continuum flow regime, intermolecular collisions are constantly happening due to the density of particles in any flow field. For this reason, DSMC starts to become impractical due to the sheer amount of computational resources required. At altitudes where flow can be classified in the continuum regime, heating is at its highest and the use of CFD is required. For intensive cases such as those relating to the flows in the ionosphere (Approximately 90km), simulation times vary from a few days to a few weeks, making uncertainty quantification and establishing a probabilistic casualty risk area in a practical timeframe highly unlikely.

However, a tradeoff can be reached through deep learning techniques. Given a set sample size of heat flux distributions, a deep learning architecture, such as a fully-connected or convolutional neural network, can be trained to produce full-detail solutions given a set of inputs such as atmospheric state and object attitude. This enables accurate emulation of the numerical solutions for given a set of flow conditions that can be produced almost instantly. With this robust framework, re-entry computations can be completed quickly while maintaining the same fidelity as numerical methods, thereby making possible probabilistic analyses through large number of ensemble runs.

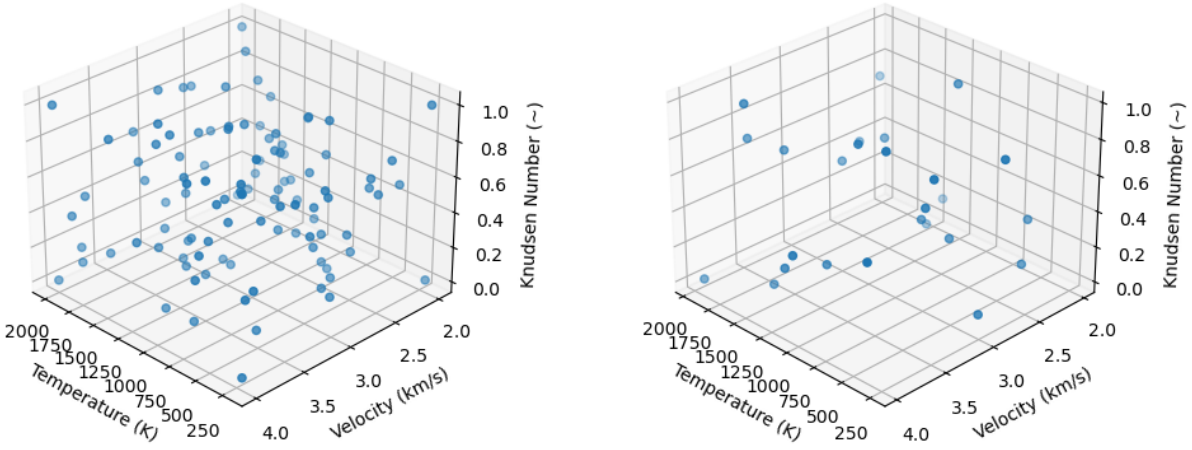


Figure 3. Distribution of DSMC Training (left) and Validation (right) Cases

2. METHODOLOGY

2.1. Direct Simulation Monte Carlo (DSMC)

DSMC is a numerical technique that is commonly used to model rarefied gas flows for finite Knudsen Numbers. Developed by G. Bird, the technique uses a stochastic simulation to probabilistically solve the Boltzmann Equation and model particle motion and interactions within a given simulation domain. The particles in the simulation are representative of a group of "real" particles (most often defined as a ratio of real-to-simulated particles), and their motions with respective intermolecular, surface, and boundary reactions are modeled as a function of time. Flow properties such as temperature, density, and velocity are sampled from the particle state on a macroscopic scale over a set amount of timesteps [4].

The DSMC package used in this study is the Stochastic PArallel Rarefied-gas Time-accurate Analyzer (SPARTA), an open source DSMC code developed by Sandia National Laboratories. SPARTA benefits from being highly adaptable and efficient, with options to add on to existing features within the code with the efficiency of C++. While it is designed to be able to be used in either traditional desktop computers or parallel computing environments, the handling of parallel communication is generally more efficient. The modular design of the package as well as ease of use makes SPARTA [5] a popular choice that is akin to NASA's internal DSMC software, DAC [6].

2.2. Geometries and 3D Meshing

The subject of this study is three shape primitives: A sphere, cube, and cylinder, for which to different flow cases exist: One with the flow facing the circular face, and the other with the flow facing the rounded face. Since Knudsen Number is a ratio of the distance between molecules in the flow and the characteristic length of the object, the characteristic length of each shape except the cube is one for the sake of simplicity. For this study, the cube has a characteristic length of two. The Knudsen Number for the cube was then scaled appropriately to make direct comparison to each shape appropriate.

Part of ensuring that accurate flow phenomena is captured for each geometry in DSMC is ensuring that the 3D object created for the simulation is properly meshed. For all of computational fluid dynamics, proper object meshing is imperative to capture important features of interest during the simulation. As such, unstructured meshing, where mesh elements are not indexed and placed systematically is generally better for CFD/DSMC simulations [7]. With unstructured meshes, the elements are usually triangular as compared to quadrilaterals for structured meshes as the detail of element placing is usually more flexible with triangular elements.

For structured meshes, problems can occur at the vertices of each element, causing skewed or inaccurate results. For unstructured meshes, greater control of mesh sizing can be employed, allowing for a better mesh defined by the user. However, there is a trade-off that must be reached: The amount of mesh elements must be sufficiently high that the geometry and any important features are accurately captured, but must not be so detailed that it over-complicates the deep neural network. Since each mesh element is an independent output of the neural network, increasing the amount of mesh elements would exponentially increase the amount connections between nodes, which would eventually make the model intractable. Furthermore, it is important to note that once an object mesh is created, the mesh cannot change as this would require the deep learning model to be retrained. To extend the concept to shape primitives of different sizes, a mesh scaling technique can be adopted as preliminary

demonstrated in the results section.

2.3. Simulation Database

As mentioned above, the ultimate goal of deep learning is to produce a model that generalizes well given a set of input-output pairs (training pairs). To this end, the data provided to the deep learning model for training must be wholly representative of the phenomena at hand. In this case of atmospheric re-entry, this data must reflect a diverse set of atmospheric flow conditions corresponding to their respective heating distributions. More specifically, the dataset should span both the transition and free-molecular regimes (where DSMC is valid), and covers typical re-entry speeds and freestream conditions seen in the altitudes corresponding to transition and free molecular flow. For this study, the three flow conditions chosen to characterize the flow were the Knudsen Number, freestream temperature, and re-entry velocity. To determine suitable values for the Knudsen Number, a range of altitudes that spans both transition and free molecular regimes is determined. The altitude range is dependent on the atmospheric conditions and the size of the object:

$$Alt \in \{85, 120\} \text{ km}$$

It is within this altitude range that density and temperature can extracted from NASA's MSIS-E-90 model to determine the mean free path of the molecules in the atmosphere, and subsequently, the Knudsen Number at a given altitude:

$$\lambda = \frac{RT}{\sqrt{2}\pi d^2 N_A P} \tag{1}$$

$$K_n = \frac{\lambda}{L} \tag{2}$$

The range of Knudsen Number, temperature, and velocity determined for the flow condition dataset are as follows:

$$Kn \in \{0.01, 1\}$$
$$T \in \{200, 2000\} \text{ K}$$
$$V \in \{2, 4\} \text{ km/s}$$

With these ranges, specific flow cases were generated via Latin hypercube sampling, which aims to statistically generate a perfectly-random sample given a number of dimensions and the minimum and maximum values that each dimension can take. This ensures that each case has practically identical weighting to one another and that no flow cases are favored in training [8]. For both the sphere and the cube, the same randomly generated flow conditions were used to initialize 100 DSMC simulations for each training case and 25 simulations for each validation case.

2.4. Gaussian Process Regression Modeling

The aim of GPR is to create a distribution of potential functions that fit the given dataset according to a Gaussian Normal Distribution. Then, the function that has the highest likelihood of fitting the data (maximum likelihood estimation) is chosen as the representative model [9]. For python, the scikit-learn library offers a premade gaussian process regressor with modular functions to include different kernels and noise estimation tools which can automatically output standard deviation bounds, which is useful for uncertainty quantification [10]. Here, the inherently non-parametric approach only requires the definition of a kernel function. For this study, the Matern covariance function (kernel) is used as it generally provides a better approximation of nonlinear functions with the appropriate definition of the smoothness parameter, ν [11]. The matern kernel is defined as

$$\begin{aligned} k(x_i, x_j) = &\frac{1}{\Gamma(\nu)2^{\nu-1}}(\frac{\sqrt{2\nu}}{l}d(x_i, x_j)^{\nu} \\ &K_{\nu}(\frac{\sqrt{2\nu}}{l}d(x_i, x_j)) \end{aligned} \tag{3}$$

Where $\nu = 1.5$, indicating a once-differentiable function.

2.5. Deep Learning

Deep Learning has become a very popular tool for approximating input-output relationships for highly complex systems. Deep Learning primarily makes use of different neural network architectures to learn and predict input-output relationships. Ultimately, the goal of any supervised learning problem is to train on a set of given inputs and outputs, and formulate a functional relationship that generalizes well enough so that given a new set of inputs, the model is able to generate an accurate prediction. In this study, a fully-connected neural network is used to model a full heating distribution over the primitive surface.

A key challenge for deep learning is the identification of the optimal set of network architecture and hyperparameters due to the large amount of controllable parameters in a neural network. A solution to this is based in Neural Architecture Search (NAS) and Hyperparameters optimization algorithms. In recent years, various NAS libraries (AutoKeras [13], Keras Tuner [13], AutoML [12], etc.) have been developed that provide a easy-to-use tool for identifying the optimal set of hyperparameters for a given dataset. In this study, we use Keras Turner. In Tables 1, 2, 3, 4, the architectures obtained from Keras Tuner are presented.

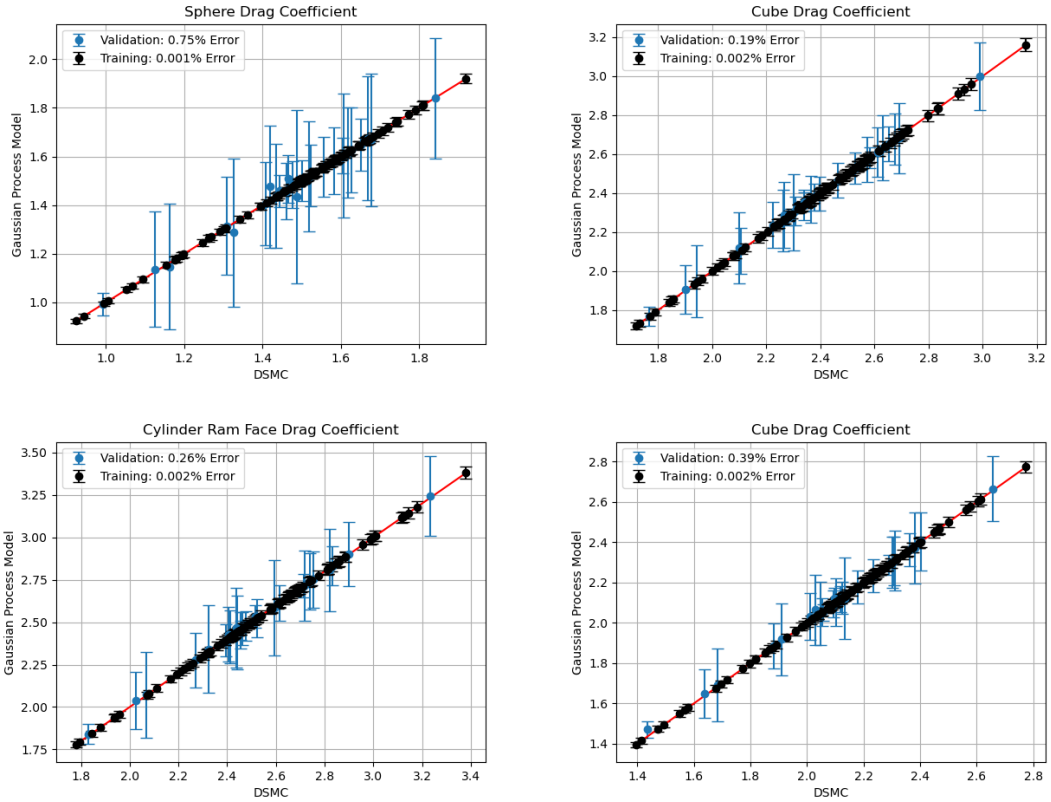


Figure 4. Drag Coefficient Results for Each Shape

3. DATA PROCESSING

There are many search algorithms used for NAS libraries, but a popular one (also used for this study) is Bayesian Optimization. This algorithm is popular compared to others because it takes a more elegant approach to converging on the best deep learning model parameters. Specifically, Bayesian Optimization leverages results from previous search trials to determine better parameters (prior information) for future trials. With enough trials specified, the algorithm will eventually converge on the best parameters for the problem at hand [15]. This generally will make more efficient use of time as other methods (random or grid search) are more "brute force" methods that do not account for past trials, making convergence in a practical amount of trials less likely.

To improve neural network performance, input and output normalization is imperative for increased learning and faster convergence. For the input data, the data was min/max normalized, defined as:

$$z = \frac{x - \min(X)}{\max(X) - \min(X)} \tag{4}$$

For the output data, the values were standard normalized, which is defined as:

$$z = \frac{x - \mu}{\sigma} \tag{5}$$

4. RESULTS

4.1. Drag Coefficient

First, the results of the Drag Coefficient predictions from the Gaussian Process Regression Model is shown in Figure 4. As expected, the training data performs extremely well with very low uncertainty bounds. However, much of the same can be said for the validation data as well. Here, a good replication of DSMC data is seen for all shapes as the mean error for each primitive is under 1%. The uncertainty bounds are larger compared to the training data due to the location of the validation inputs within the input space compared to the training inputs. Cases where predictions have higher uncertainty bounds generally correspond to input values that lie close to one of the training inputs, where as predictions with low uncertainties indicate that the inputs for that specific case lie very close or directly coincide with one of the training inputs. Nonetheless, Gaussian Process Regression Modeling offers a good prediction tool for drag coefficients in re-entry flow. A table summarizing the performance of the GPR is shown in Table 5

Table 1. Sphere Model Parameters

Layer #	# of Nodes	Activation Function
1	928	elu
2	372	elu

Table 2. Cube Model Parameters

Layer #	# of Nodes	Activation Function
1	236	tanh
2	344	sigmoid

Table 3. Cylinder Ram Face Model Parameters

Layer #	# of Nodes	Activation Function
1	302	relu
2	302	softmax
3	112	elu

Table 4. Cylinder Side Face Model Parameters

Layer #	# of Nodes	Activation Function
1	552	softsign
2	128	elu
3	122	sigmoid

4.2. Heating Distributions

For the full heating distributions, much of the same can be seen as compared to the drag coefficient. For all shapes, the deep learning models offer a good replication for of DSMC data in both training and validation data, shown in Figure 5. For the sphere and the cube specfically, there is very little over- or underfitting to speak of, as most of the deep learning predictions coinciding nearly perfectly with the DSMC data. With the cylinder cases, the side face validation indicates there is some overprediction at higher heating values which is due to a bias to lower Knudsen Number cases. However, the training data shows a good replication of DSMC results. For the ram face, there seems to be a less tight fit around the 45 degree line than in previous cases. However, the model still performs well nonetheless.

The visualizations for the lowest and highest heating cases predicted by deep learning is shown in Figure 6. For the sphere, closed form solutions for the whole distribution is modeled as a function of the stagnation point heat flux. Furthermore, it is assumed that regardless of flow regime, the distribution would be identically mapped across the entire object. However, this is seen to not be true. It is evident that for lower values of heat flux, the highest heating is much more distributed to the face exposed to the flow. For the lowest Knudsen Number case, the heating is much more concentrated in the center, which indicates that as Knudsen Number increases, the stagnation point heat flux increases exponentially compared to the remainder of the distribution. For the cube and cylinder, the most important feature to be captured that has largely been impossible to model as of date is the areas of higher heating around edges and corners due to flow expansion around these edges. For these three shapes, the deep learning model is able to capture this phenomena quite well. Similarly to the sphere, the model is able to predict more distributed values of high heating at the higher Knudsen Number cases and more concentrated points of heat flux at lower Knudsen Numbers.

Table 5. Drag Coefficient Errors (%)

Shape	Training	Validation
Sphere	0.75%	0.001%
Cube	0.19%	0.002%
Cyl. Side Face	0.26%	0.002%
Cyl. Ram Face	0.39%	0.002%

4.3. Scaling Demonstration

As previously stated, once the object is meshed, the distribution of mesh elements must not change. To allow the model to make valid predictions for objects of different characteristic lengths, the mesh can be scaled as long as the input Knudsen Number is scaled appropriately as well. Shown in Figure 7 is a demonstration of this ability, where the characteristic length is halved from its original value to one. The deep learning model, like the original cube validation cases, is still able to detect the areas of concentrated heating due to flow expansion. For the overall distribution, the model is able to replicate the DSMC data reasonably well. At lower, non-critical points of heating, there is slight over-prediction. Conversely, at higher values, there is slight underprediction, but the model still produces reasonable results.

5. CONCLUSIONS

The feasibility of using deep learning to replicating full, high-fidelity heating distributions and drag coefficients on shape primitives in re-entry flow has been demonstrated. Using Gaussian Process Regression Modeling, drag coefficient is accurately modeled, with a mean error across all shape primitives of under 1%. For the full heating distribution, fully-connected neural networks are able to detect features such as concentrated values of high heating at the edges and corners for non-convex objects such as a cube and cylinder.For all shapes, there is a very good replication of DSMC data, thus demonstrating the

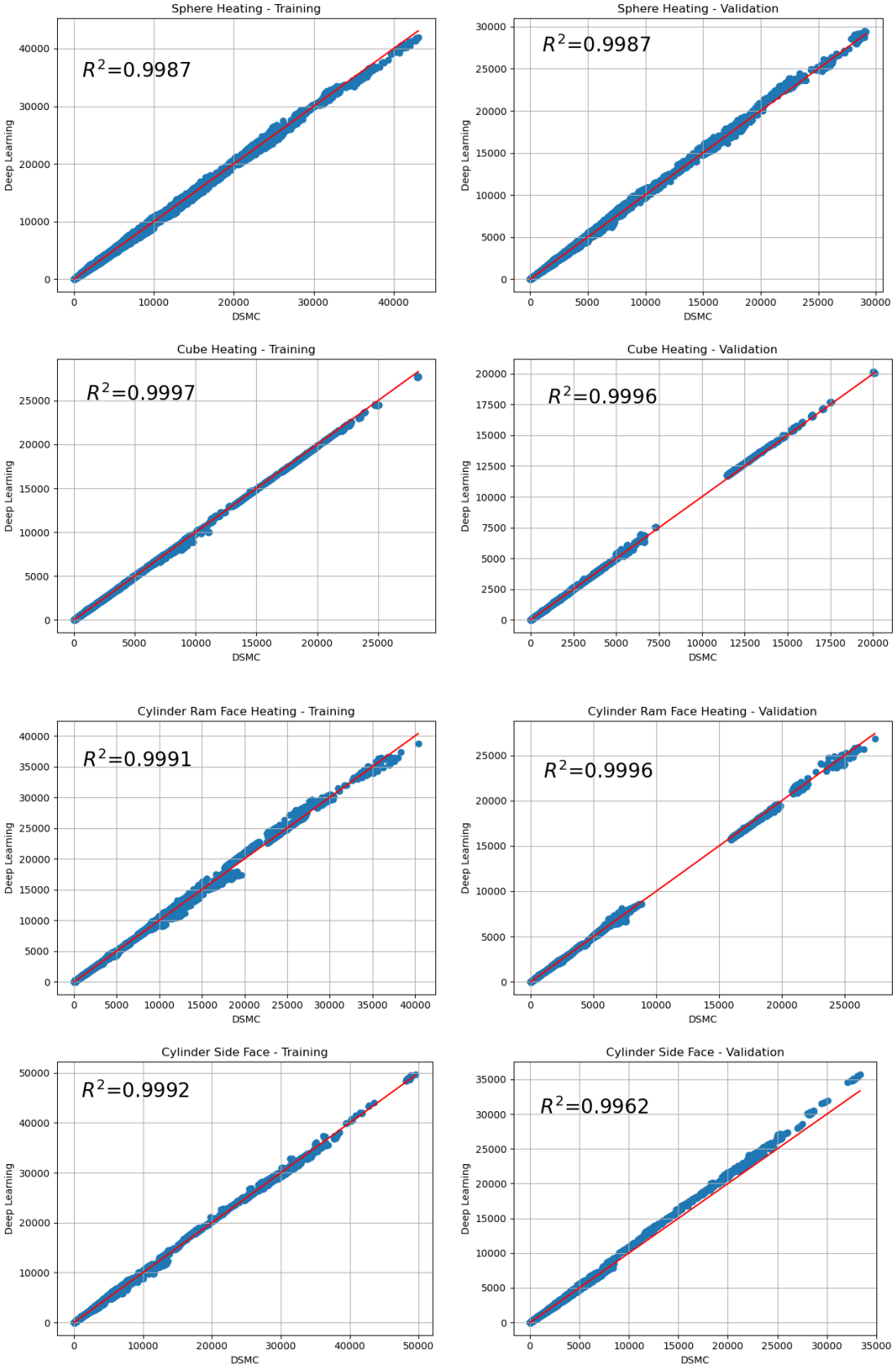
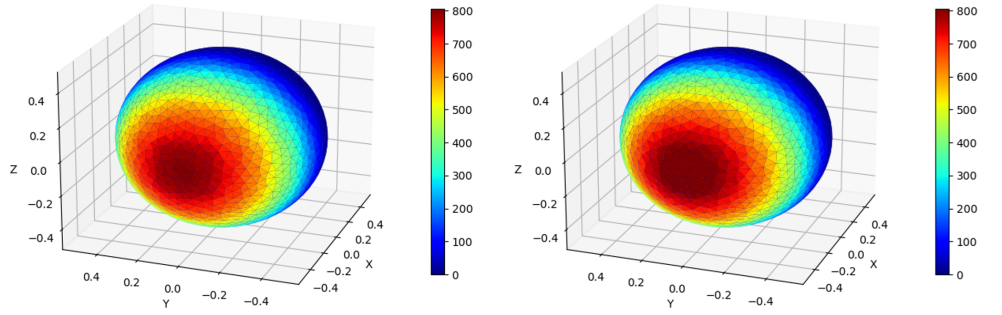
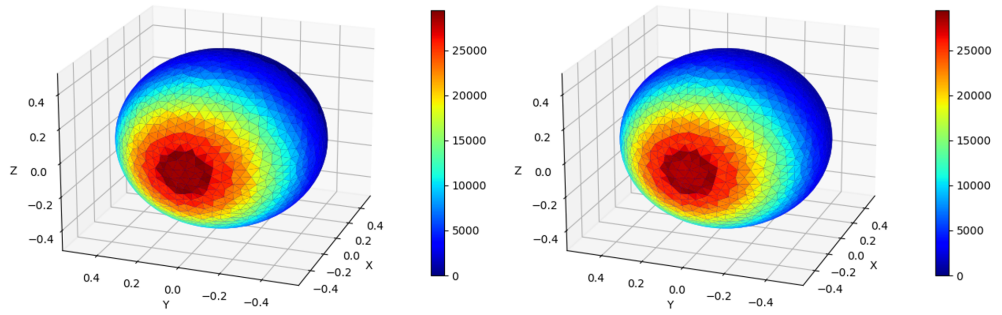


Figure 5. Heating Results for Each Shape

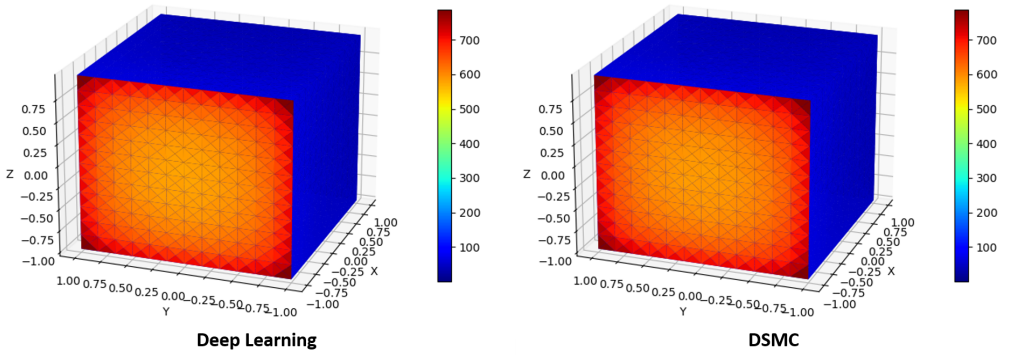
Kn = 0.84, T = 286.1 K, V = 2709.6 m/s



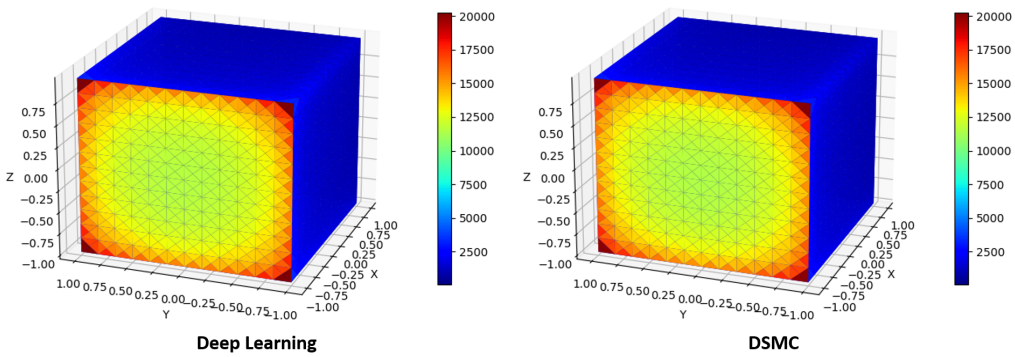
Kn = 0.04, T = 1514.9 K, V = 3693.1 m/s



Kn = 0.84, T = 286.1 K, V = 2709.6 m/s



Kn = 0.04, T = 1514.9 K, V = 3693.1 m/s



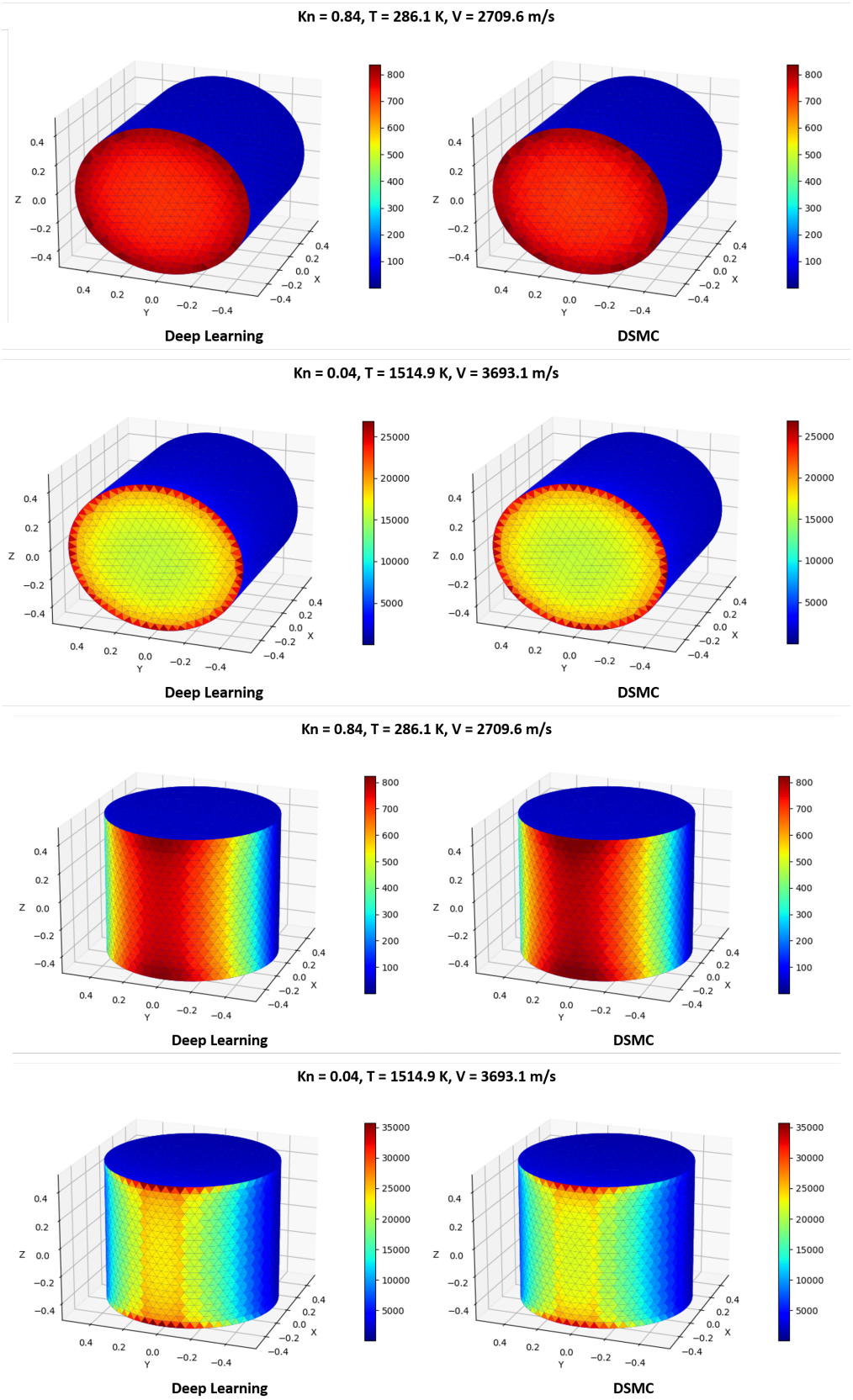


Figure 6. Heating Visualizations for Each Shape

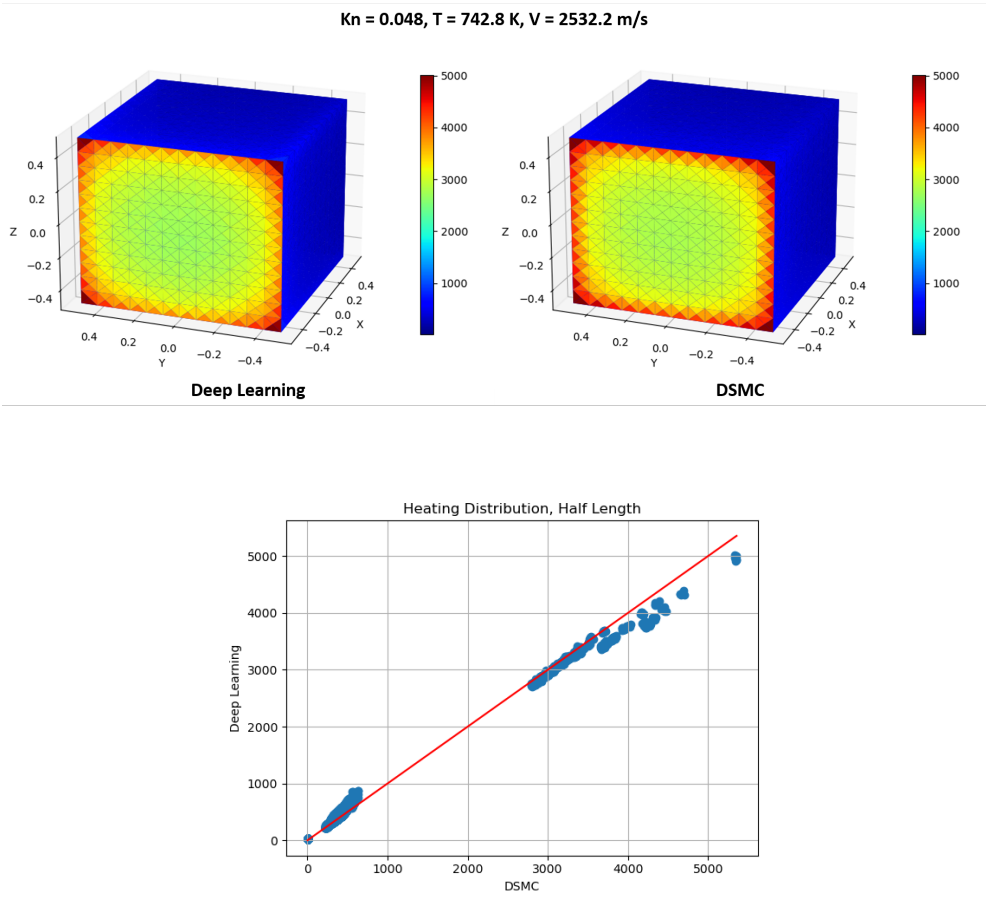


Figure 7. Scaling Results

ability of deep learning to replicate high-fidelity data in a fraction of a second. To extend the concept to shape primitives of all sizes, a scaling demonstration is shown. Here, the models provided are still able to make high-fidelity predictions despite not originally training on an object of that size.

6. FUTURE WORK

Future studies to further advance the development presented in this paper include:

- Implement attitude variations for tumbling objects
- Implement pixelator from Computer Graphics for Space Debris [16]
- Uncertainty Quantification for predictions
- Eventually extend concept to hollow shape primitives

7. ACKNOWLEDGEMENTS

This research was made possible by NASA West Virginia Space Grant Consortium, NASA Agreement #80NSSC20M0055. The authors would like to acknowledge the use of the High Performance Computing Resources Thorny Flat at West Virginia University housed at the Pittsburgh Supercomputing Center.

REFERENCES

1. NASA, (2019). Process for Limiting Orbital Debris
2. Piyush M. Mehta, Edmondo Minisci, Massimiliano Vasile, Andrew Walker and Melrose Brown, "Sensitivity Analysis towards Probabilistic Re-Entry Modeling of Spacecraft and Space Debris," AIAA Modeling and Simulation Technologies Conference, Aviation 2015, Dallas, Texas, June 22-26, 2015.
3. Cristina Parigini, Irene Pontijas Fuentes, Rodrigo Haya Ramos, and Stefania Cornara, "Debris tool and its use in mission analysis activities," in 8th ESA symposium on aerothermodynamics of space vehicles, Lisbon, Portugal, March 2015.
4. Alexander F., Garcia A., (1997). The Direct Simulation Monte Carlo Method *Computers in Physics*, **11**(588)
5. S. J. Plimpton, S. G. Moore, et.al., (2019) Direct Simulation Monte Carlo on petaflop supercomputers and beyond *Physics of Fluids*, **31**(086101)

6. “DSMC Code Simulates Rarefied Gas Dynamic Environments.” NASA, www.nasa.gov/centers/johnson/techtransfer/technology/MSC-23445-1-dsmc-dac.html.

7. Tu, J., Liu,C., et.al., (2013) Practical Guidelines for CFD Simulation and Analysis *Computational Fluid Dynamics*

8. Helton, J., Davis, F., (2002). Latin Hypercube Sampling and the Propagation of Uncertainty in Analyses of Complex Systems.

9. Sit, Hilarie. “Quick Start to Gaussian Process Regression - Towards Data Science.” Medium, 12 June 2020, towardsdatascience.com/quick-start-to-gaussian-process-regression-36d838810319.

10. “Gaussian Processes — Scikit-Learn 0.24.1 Documentation.” Scikit-Learn.Org, scikit-learn.org/stable/modules/gaussianprocess.html. Accessed 7 Aug. 2020.

11. “Matern kernel — Scikit-Learn 0.24.1 Documentation.” Scikit-Learn.Org, scikit-learn.org/stable/modules/generated/sklearn. gaussianprocess.kernels.Matern.html. Accessed 7 Aug. 2020.

12. “AutoML.Org.” AutoML, www.automl.org. Accessed 8 Mar. 2021.

13. “AutoKeras.” AutoKeras, autokeras.com. Accessed 8 Mar. 2021.

14. “Keras Tuner.” Keras Tuner, keras-team.github.io/keras-tuner/documentation/. Accessed 7 Dec. 2020.

15. “Tuners - Keras Tuner.” Keras Tuner, keras-team.github.io/keras-tuner/documentation/tuners. Accessed 7 Dec. 2020.

16. Piyush M. Mehta, Gonzalo Blanco-Arnao, Davide Bonetti, Edmondo Minisci, Massimiliano Vasile, “Computer Graphics for Space Debris”, 6th International Conference on Astrodynamics Tools and Techniques, Darmstadt, Germany, March 14-17, 2016.