# Learning Hypersonic Flow Fields With Sparse Data Using a Multi-Fidelity Neural Operator

By
Saleem Adam Ali
Sc.M., Brown University 2026

Thesis

Submitted in partial fulfillment of the requirements for the Degree of
Master of Science in the School of Engineering at Brown University

PROVIDENCE, RHODE ISLAND

MAY 2026

**AUTHORIZATION TO LEND AND REPRODUCE THE THESIS**

**As the sole author of this thesis, I authorize Brown University to lend it to other institutions or individuals for the purpose of scholarly research.**

Date_____ _____

Saleem Adam Ali, Author

**I further authorize Brown University to reproduce this thesis by photocopying or other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.**

Date_____ _____

Saleem Adam Ali, Author

This thesis by Saleem Adam Ali is accepted in its present form by the School of Engineering as satisfying the thesis requirements
for the degree of Master of Science

Date_____ _____

George Em Karniadakis, Advisor

Date_____ _____

Khemraj Shukla, Co-Advisor

Approved by the Graduate Council

Date _____ _____

David P. Lindstrom, Dean of the Graduate School

# Abstract

Accurate prediction of hypersonic aerothermodynamic flow fields is critical for atmospheric entry vehicle design, yet high-fidelity computational fluid dynamics (CFD) simulations remain prohibitively expensive due to the need of fine grids required to resolve strong shock waves, thin boundary layers, and stiff thermochemical reaction networks. This thesis investigates the use of Deep Operator Networks (DeepONets) as surrogate models for steady-state laminar hypersonic flow over a generic sphere–cone entry configuration with an 18-species thermochemical nonequilibrium reacting flow model and carbon surface ablation.

Both single-fidelity (SF) and multi-fidelity (MF) neural operator architectures are developed and evaluated. High-fidelity training data are generated using the NASA's FUN3D solver, which solves the compressible Navier–Stokes equations with two-temperature non-equilibrium chemistry, while lower-cost aerodynamic data are obtained using the engineering-level CBAero code. The multi-fidelity framework combines these heterogeneous data sources through a shared trunk network encoding a common output function space representation, with separate branch networks capturing fidelity-specific corrections.

The MF-DeepONet achieves the largest improvement for pressure, reducing the relative $L^2$ error from 9.77% to 2.04%, with consistent improvement also observed for density. Temperature fields show more modest gains due to weaker cross-fidelity correlation arising from nonequilibrium thermochemical processes not captured by the low-fidelity model. Sensitivity studies confirm that incorporating even modest amounts of low-fidelity data substantially sta-

bilizes training when high-fidelity supervision is sparse.

The influence of multi-fidelity loss weighting strategies on convergence and accuracy is also investigated. Results indicate that pressure and density benefit from stronger high-fidelity supervision, while temperature fields respond better to curriculum-style schedules. Once trained, both surrogate models provide inference times orders of magnitude faster than CFD, supporting their use in parametric studies and design-space exploration for atmospheric entry applications.

# Acknowledgement

I am deeply thankful for the experiences I have had over the past two years. The work, the challenges, the lessons, and above all the people have made this period meaningful and transformative. During my time here, I have learned a great deal and have grown both personally and professionally. I am especially grateful for the encouragement and support of my family, friends, and Professors.

I would like to express my appreciation to Professor George Karniadakis, Professor Khemraj Shukla, and the Crunch group for their guidance, mentorship, and support. It has been a privilege to work within such a dedicated research community. I am especially grateful to Professor Shukla for the generous time and expertise he has shared with me. His mentorship has shaped both this project and my development as a researcher and Army Officer in profound ways.

I am also grateful to Sean George and Arthur Huang from Draper for their guidance and for helping keep the project focused through our weekly meetings. Their engineering insight and their expertise in hypersonics and computational fluid dynamics strengthened this work and furthered my development as a researcher. I appreciate the time, perspective, and feedback they provided throughout this effort.

Finally, I would like to thank Draper for supporting my time at Brown. I am also grateful to the U.S. Army for supporting my graduate education while I served as an active duty army officer. I hope to carry forward the knowledge, perspective, and skills I have gained here as I continue a life of service.

# Contents

# List of Figures

# 1 Introduction

Modeling atmospheric re-entry is one of the central challenges in hypersonic vehicle design. Vehicles entering Earth's atmosphere at hypersonic speeds (Mach Number, $Ma > 5$) encounter extreme aerodynamic heating caused by strong shock waves, high-temperature gas effects, and complex chemical reactions. These conditions produce a highly nonlinear aerothermodynamic environment in which accurate prediction of surface temperature and heat flux is critical for assessing thermal protection system performance and ensuring structural survival of the vehicle [7, 29]. In particular, high-temperature gas chemistry and surface ablation introduce strong coupling between fluid dynamics, thermodynamics, and material response, significantly increasing the complexity of the governing models.

Traditionally, these hypersonic flow problems are studied using high-fidelity computational fluid dynamics (CFD). Modern solvers resolve the compressible Navier–Stokes equations together with multi-species chemical kinetics and thermochemical nonequilibrium models. While these approaches provide accurate predictions, they are computationally expensive. A single high-fidelity simulation of a hypersonic entry configuration may require tens of thousands of core-hours due to the need to resolve strong shocks, thin boundary layers, and stiff chemical reaction networks. As a result, tasks that require repeated evaluations—such as parametric sweeps, uncertainty quantification, shape optimization, or large-scale data generation—can quickly become computationally prohibitive [30].

The computational burden becomes particularly pronounced for the class

of problems considered in this work. Hypersonic entry flows require very fine spatio-temporal discretization to resolve thin boundary layers and steep temperature gradients even in the absence of shock. In addition, the present configuration includes multi-species thermochemical non-equilibrium with an 18-species reacting flow model and carbon ablation at the vehicle surface. These effects introduce numerous additional transport equations and stiff chemical reaction networks that significantly increase the cost of each numerical solve. Consequently, even steady-state laminar simulations of relatively simple geometries, such as the sphere–cone configuration considered here, can require substantial computational resources.

These computational challenges motivate the development of efficient surrogate models capable of approximating high-fidelity simulations at significantly reduced cost. In recent years, neural operators [21, 19] have emerged as a promising class of machine learning models for this purpose. Unlike conventional neural networks that learn mappings between finite dimensional vectors, neural operators are designed to learn mappings between functional spaces (Infinite dimensional function spaces), effectively approximating the solution operator of governing partial differential equations. Once trained offline, these models can infer the flowfields orders of magnitude faster than traditional numerical solvers while retaining much of their predictive capability and accuracy. Furthermore, multi-fidelity learning frameworks enable neural operators to combine information from sparse high-fidelity simulations with larger quantities of inexpensive low-fidelity data, improving accuracy while reducing the number of expensive CFD evaluations required [22].

In this study, neural operator methods are applied to model steady-state laminar hypersonic flow over a generic sphere–cone entry vehicle. The configuration includes multi-species thermochemical non-equilibrium and carbon ablation represented through an 18-species reacting flow model [20]. High-fidelity data are generated using the NASA FUN3D solver [2], while lower-cost aerodynamic data are obtained using the engineering-level CBAero code [17]. The goal is to construct surrogate models capable of predicting the aerodynamic and thermodynamic flow fields across a range of Mach numbers and altitudes representative of atmospheric entry conditions.

Specifically, this work implements Deep Operator Networks (DeepONets) as surrogate models for the hypersonic flowfield surrounding the sphere–cone configuration. Both single-fidelity and multi-fidelity neural operator architectures are investigated and compared in terms of predictive accuracy and data efficiency. In addition, the effectiveness of weighting strategies for combining different data fidelities and emphasizing important spatial regions of the flowfield is examined. In short, it was found that a multi-fidelity DeepONet is a cost effective surrogate model in terms of computational resources, time, and accuracy.

The remainder of this thesis is organized as follows. Section 2 introduces the computational foundations underlying neural networks. Section 3 presents neural operators and the DeepONet architecture. Section 4 describes multi-fidelity learning strategies. Section 5 introduces the residual-based attention weighting scheme and other weighting strategies used in this work. Section 6 presents the results and discussion, and Section 7 outlines directions for future

research.

# 2 Neural Networks

Neural networks have become powerful tools for approximating complex nonlinear mappings arising in scientific and engineering applications. In particular, they provide flexible parametric models capable of approximating high-dimensional functions from data. This section introduces neural networks as universal function approximators, which form the conceptual foundation for neural operators introduced in the following section.

Neural networks are machine learning models inspired by biological neural systems and are formulated as computational graphs of artificial neurons [10]. Numerical inputs are transformed through successive layers parameterized by weights and biases, with optimal parameters obtained by minimizing a loss function using gradient-based optimization. This process relies on automatic differentiation and backpropagation for efficient computation of gradients.

A fundamental neural network architecture is the Multi-Layer Perceptron (MLP), first introduced in 1958. Given an input vector $x \in \mathbb{R}^{n_x}$, each layer applies an affine transformation followed by a nonlinear activation function. The first layer is given by

$$L^1 = W_1 x + b_1, \tag{1}$$

where $W_1 \in \mathbb{R}^{L^1 \times n_x}$, $b_1 \in \mathbb{R}^{L^1 \times 1}$, and the layer width $L^1$ is a tunable hyperparameter.

Subsequent layers are defined recursively as

$$L_k = W_k L_{k-1} + b_k, \tag{2}$$

with $W_k \in \mathbb{R}^{l_k \times l_{k-1}}$ and $b_k \in \mathbb{R}^{l_k \times 1}$. The output of an MLP with $N$ layers is expressed as the composition

$$\mathcal{O}_\theta(x) = L_N \circ \sigma \circ L_{N-1} \circ \cdots \circ \sigma \circ L^1. \tag{3}$$

For regression tasks, the output layer typically employs the identity activation function, resulting in a linear mapping from the final hidden representation to the predicted output [10].

Training a neural network can be formulated as an unconstrained optimization problem:

$$\min_{\theta \in \mathbb{R}^p} \ \mathcal{L}(\theta), \tag{4}$$

where $\theta$ denotes the set of model parameters and $\mathcal{L}(\theta)$ is a task-dependent loss function. For supervised learning over a dataset $\{(x_i, y_i)\}_{i=1}^N$, a common choice is the mean squared error (MSE),

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left\| f_\theta(x_i) - y_i \right\|^2, \tag{5}$$

or alternatively the mean absolute error (MAE).

While conventional loss functions are purely data-driven, Physics-Informed Neural Networks (PINNs) augment the objective by incorporating residuals of

governing partial differential equations (PDEs):

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{data}}(\theta) + \lambda \, \mathcal{L}_{\text{PDE}}(\theta), \tag{6}$$

where $\mathcal{L}_{\text{PDE}}$ penalizes violations of the underlying physical laws and $\lambda$ is a weighting parameter. This formulation enables learning consistent with both observational data and physical constraints [26].

Model parameters are updated using gradient-based optimization methods. A standard gradient descent iteration is given by

$$\theta^{(k+1)} = \theta^{(k)} - \alpha \nabla_\theta \mathcal{L}(\theta^{(k)}), \tag{7}$$

where $\alpha > 0$ is the learning rate. In practice, adaptive methods such as the Adam optimizer are often preferred, as they scale parameter updates using estimates of the first and second moments of the gradients [15].

Gradients of the loss function are computed via backpropagation over the network's computational graph. The compositional structure of neural networks allows efficient evaluation of derivatives via the chain rule, while automatic differentiation enables accurate and scalable computation of gradients.

## 2.1 Deep Learning Frameworks

The widespread adoption of neural networks has led to the development of open-source deep learning frameworks such as TensorFlow, PyTorch, and JAX. These platforms provide tools for constructing and training neural networks, including support for defining model architectures, initializing parameters, for-

mulating loss functions, and implementing optimization algorithms.

The key components involved in building and training neural networks are summarized as follows:

1. **Architecture:** Specifies the structure of the network, including hyperparameters such as depth, width, and activation functions. Task-specific layers, such as convolutional, recurrent, or transformer modules, may also be incorporated.

2. **Initialization:** Model parameters are initialized prior to training. Initialization strategies influence training stability and convergence behavior.

3. **Loss Function:** The loss function defines the learning objective and measures the discrepancy between predictions and target values.

4. **Loss Weighting:** When multiple objectives or imbalanced outputs are present, weighting individual loss terms can guide the training process.

5. **Optimizer:** Optimization algorithms update model parameters using gradient information. Choices of optimizer and learning rate schedule affect convergence speed and final performance.

6. **Data Processing:** Preprocessing steps such as normalization and scaling improve numerical conditioning and training stability.

Neural network design is largely guided by empirical considerations, as theoretical guarantees for convergence and performance remain limited. Con-

sequently, architecture selection, initialization, loss formulation, and optimization strategies are often problem-dependent, and parametric studies are commonly used to identify effective configurations.

## 2.2 Universal Approximation and Error Sources

The universal approximation theorem states that a sufficiently wide multilayer perceptron (MLP) can approximate any continuous function to arbitrary accuracy using a single hidden layer [10]. While this result is theoretically significant, practical neural networks rarely achieve negligible error, particularly when restricted to shallow architectures. In practice, convergence behavior depends on multiple interacting factors, and general error bounds remain difficult to establish.

The total error of a neural network model is commonly decomposed into three components: approximation error, estimation error, and optimization error. Approximation error measures the discrepancy between the target function and the best representation achievable within a given model class. Estimation error quantifies the gap between this ideal representation and the best model learned from finite data. Together, these two terms define the generalization error, which characterizes how well the trained model represents the target distribution. Optimization error arises from the training process itself and is often the dominant source of error in neural networks due to the non-convexity of the loss landscape.

Formally, let $g$ be a target function and $\mathcal{H}$ a model class. Denote by $h_g$

the best approximation to $g$ within $\mathcal{H}$. The approximation error is defined as

$$E_{\text{approx}} = \|g - h_g\|. \tag{8}$$

This error reflects the inherent expressive limitation of the model class and is typically the primary concern in classical regression problems involving convex or orthogonal spaces.

Estimation error becomes significant when learning from finite or sparse data. Let $g_N = [g(x_1), g(x_2), \ldots, g(x_N)]$ denote sampled evaluations of $g$ , and let $h_{g_N}$ be the best approximation of $g_N$ within $\mathcal{H}$. The estimation error is given by

$$E_{\text{est}} = \|h_g - h_{g_N}\|. \tag{9}$$

A representative example is aliasing in frequency-domain approximations, where insufficient sampling leads to large estimation error even when the approximation error is zero.

The final contribution is optimization error, which arises from imperfect minimization of the loss function. Unlike traditional regression methods with closed-form solutions, neural networks rely on iterative optimization of non-convex objectives. Let $h_{\text{opt}}$ denote the model obtained at an optimally reached training state. The optimization error is defined as

$$E_{\text{opt}} = \|h_{g_N} - h_{\text{opt}}\|. \tag{10}$$

This term captures the discrepancy caused by convergence to suboptimal local minima or saddle points.

Despite the empirical success of neural networks, a comprehensive theoretical understanding of convergence and error behavior remains incomplete. Consequently, model design is often guided by empirical intuition and parameter sweeps. However, in many scientific computing applications, the objective is not merely to approximate a finite-dimensional function but rather to approximate the solution operator of a partial differential equation. Such operators map input functions or parameters, such as boundary conditions or physical coefficients, to solution fields. This perspective motivates the development of neural operators, which extend neural network approximation from functions to operators.

# 3 Neural Operators

Neural operators aim to approximate mappings between functions rather than finite-dimensional vectors, thereby extending neural network methodologies to infinite-dimensional function spaces. In the context of scientific computing, such mappings can be interpreted as solution operators of partial differential equations, which map problem inputs—such as boundary conditions, parameters, or forcing functions—to the corresponding solution fields.

One of the earliest and most influential approaches for learning operators with neural networks is the Deep Operator Network (DeepONet), which provides a practical and theoretically grounded framework for approximat-

ing nonlinear operators [21]. Since its introduction, several alternative neural operator architectures have been proposed, including Fourier Neural Operators (FNOs), which leverage spectral representations [19], and Laplace-based neural operators, which exploit integral transform formulations [6].

In addition, multiple variants of the original DeepONet architecture have been developed to improve accuracy, stability, and adaptability to specific classes of problems. In this work, we focus on selected DeepONet variants and introduce a novel weighting strategy aimed at improving surrogate modeling performance for the hypersonic generic sphere–cone configuration.

Before introducing the DeepONet architecture, it is important to distinguish between function approximation and operator approximation. Conventional neural networks approximate mappings between finite-dimensional spaces:

$$f_\theta : \mathbb{R}^{d_x} \to \mathbb{R}^{d_y}, \quad x \mapsto y = f_\theta(x), \tag{11}$$

where $x \in \mathbb{R}^{d_x}$ is a fixed-dimensional input vector and $y \in \mathbb{R}^{d_y}$ is the corresponding output. Such models are well-suited for standard regression or classification tasks.

In contrast, many problems in scientific computing involve mappings between infinite-dimensional function spaces. Let $\mathcal{X}$ and $\mathcal{Y}$ denote Banach (or Hilbert) spaces of functions. The goal is to approximate a nonlinear operator

$$\mathcal{G} : \mathcal{X} \to \mathcal{Y}, \quad u \mapsto \mathcal{G}(u), \tag{12}$$

where $u$ is an input function (e.g., initial conditions, boundary conditions, or

coefficients), and $\mathcal{G}(u)$ is the corresponding solution field.

A prototypical example arises from partial differential equations (PDEs). Consider a parametrized PDE of the form

$$\mathcal{N}(u;\mu) = 0 \quad \text{in } \Omega, \tag{13}$$

subject to appropriate boundary and/or initial conditions, where $\mathcal{N}$ is a (possibly nonlinear) differential operator, $u$ is the solution field, and $\mu$ denotes input parameters or forcing terms. The associated solution operator can be written as

$$\mathcal{G} : \mu \mapsto u, \tag{14}$$

or more generally as a mapping from input functions (e.g., boundary data) to solution functions.

Neural operators aim to approximate $\mathcal{G}$ directly:

$$\mathcal{G}_\theta \approx \mathcal{G}, \tag{15}$$

such that for any admissible input function $u \in \mathcal{X}$, the model produces an accurate approximation $\mathcal{G}_\theta(u) \in \mathcal{Y}$. Unlike classical neural networks, neural operators are designed to generalize across discretizations, enabling learning at the level of function spaces rather than fixed grids.

This perspective is particularly advantageous for parametric PDE problems, where the objective is to efficiently evaluate solution fields across varying inputs, such as boundary conditions, geometries, or flow parameters. In

such settings, neural operators provide a powerful framework for constructing fast and scalable surrogate models that retain the underlying structure of the governing equations.

## 3.1 DeepONet

The Deep Operator Network (DeepONet) is motivated by the universal approximation theorem for operators, which establishes that certain neural network architectures can approximate nonlinear operators between infinite-dimensional function spaces [31, 21]. Given an operator $\mathcal{G}$, the objective is to learn its action on an arbitrary input function $f \in \mathcal{X}$:

$$\mathcal{G} : f \mapsto \mathcal{G}[f]. \tag{16}$$

DeepONet achieves this by combining two neural networks with complementary roles (see Figure 1**A**). The first component, referred to as the *branch network*, encodes the input function. In accordance with the theoretical framework, the function $f$ is evaluated at a fixed set of sensor locations $\{x_i\}_{i=1}^m \subset \Omega$, producing a finite-dimensional representation

$$f_s = [f(x_1), f(x_2), \ldots, f(x_m)] \in \mathbb{R}^m. \tag{17}$$

This discretized input is passed through the branch network, yielding a latent representation

$$c(\theta; f_s) \in \mathbb{R}^{N_\ell}, \tag{18}$$

where $\theta$ denotes the trainable parameters and $N_\ell$ is the latent dimension.

The second component, known as the *trunk network*, encodes the dependence of the output on the evaluation location $x \in \Omega$. Its role is to learn a set of basis functions spanning the output function space:

$$\phi(\mu; x) \in \mathbb{R}^{N_\ell}, \tag{19}$$

where $\mu$ denotes the trunk network parameters.

The outputs of the branch and trunk networks are combined via an inner product, consistent with the structure implied by the operator approximation theorem. The resulting approximation of the operator is given by

$$\mathcal{G}_\theta[f](x) \approx \sum_{i=1}^{N_\ell} c_i(\theta; f_s)\, \phi_i(\mu; x) = \langle c(\theta; f_s), \phi(\mu; x) \rangle_{\mathbb{R}^{N_\ell}}. \tag{20}$$

Here, the latent dimension $N_\ell$ determines the expressiveness of the shared representation and can be interpreted as the number of learned basis functions.

Training the DeepONet consists of minimizing a data-driven loss over a set of input functions $\{f^{(j)}\}$ and evaluation points $\{x_k\}$:

$$\min_{\theta,\mu} \frac{1}{N} \sum_{j=1}^{N_f} \sum_{k=1}^{N_x} \left| \mathcal{G}[f^{(j)}](x_k) - \sum_{i=1}^{N_\ell} c_i(\theta; f_s^{(j)})\, \phi_i(\mu; x_k) \right|^2. \tag{21}$$

This optimization problem is inherently non-convex and is typically solved using gradient-based methods such as stochastic gradient descent or adaptive optimizers. As a result, DeepONet models are subject to approximation, estimation, and optimization errors, analogous to those encountered in standard

neural networks (see Section 2). From a modeling perspective, DeepONet can therefore be interpreted as a natural extension of classical neural networks to operator-valued learning tasks.

In this study, we employ both a standard (vanilla) DeepONet and a multi-fidelity DeepONet architecture. These models are evaluated for their ability to approximate the solution operator associated with the hypersonic flow problem described in Section 5. Furthermore, we investigate the effectiveness of loss-weighting strategies in improving surrogate model accuracy and robustness.

# 4 Multi-Fidelity Learning

Multi-fidelity learning leverages data of varying accuracy and computational cost to efficiently approximate high-fidelity solutions. In computational science and engineering, data can often be obtained from multiple sources—such as coarse-grid simulations, reduced physical models, or experimental measurements—each offering different trade-offs between fidelity and expense. The fundamental objective of multi-fidelity learning is to combine these heterogeneous sources in a way that maximizes predictive accuracy while minimizing the reliance on costly high-fidelity data.

High-fidelity simulations, such as those produced by direct numerical simulation (DNS) or high-order computational fluid dynamics (CFD) solvers, provide accurate representations of the underlying physics but are often prohibitively expensive. In contrast, low-fidelity models—including inviscid flow solvers, empirical correlations, reduced-order models, or coarse-grid simula-

tions—are computationally inexpensive but capture only partial or approximate physics. As a result, many scientific and engineering problems are characterized by an abundance of low-fidelity data and a scarcity of high-fidelity data due to computational or experimental constraints.

Long before the advent of data-driven machine learning methods, researchers sought to exploit this imbalance through physics-based multi-fidelity strategies. One prominent example is the *space mapping* technique, originally developed for electromagnetic optimization, which couples fast, low-fidelity models with a small number of expensive high-fidelity simulations through an explicit parameter-space transformation. By iteratively aligning the coarse model to the fine model, space mapping directs the bulk of the optimization effort to the inexpensive model while retaining the accuracy of the high-fidelity representation [5]. This approach can be viewed as an early precursor to modern multi-fidelity learning, albeit one based on hand-crafted mappings and problem-specific structure rather than data-driven inference.

Modern multi-fidelity learning generalizes this idea by replacing explicit analytical or heuristic mappings with learned statistical or machine-learning models that infer the relationship between fidelity levels directly from data. Rather than discarding inexpensive low-fidelity data, these methods leverage it to guide and regularize the learning of high-fidelity predictors. This paradigm has been shown to significantly improve data efficiency and predictive performance across a wide range of applications, including aerodynamics, materials science, and structural analysis [22, 11, 13, 34, 18]. In this sense, contemporary multi-fidelity machine learning can be interpreted as a data-driven extension

of earlier physics-based approaches such as space mapping, offering greater flexibility, scalability, and applicability to complex, high-dimensional systems.

Mathematically, consider two models that approximate an unknown target function $f(x)$:

$$f_H(x) = f(x) + \epsilon_H(x), \quad f_L(x) = \rho f(x) + \epsilon_L(x), \tag{22}$$

where $f_H$ and $f_L$ denote high- and low-fidelity outputs, respectively, and $\epsilon_H, \epsilon_L$ represent their respective discrepancies. A central assumption in multi-fidelity modeling is that the low-fidelity model is correlated with the high-fidelity model, allowing inexpensive low-fidelity data to inform predictions of the more accurate high-fidelity response. The goal of multi-fidelity learning is to construct a surrogate model $\hat{f}_H(x)$ that closely approximates $f_H(x)$ while exploiting the cheaper information available from $f_L(x)$.

Various multi-fidelity strategies have been proposed, ranging from co-kriging in Gaussian process models to deep neural network architectures that explicitly encode fidelity relationships. In particular, neural operator frameworks provide a natural extension to multi-fidelity learning by modeling mappings between function spaces at multiple fidelity levels.

In hypersonic flow modeling, high-fidelity simulations that include detailed thermochemistry and ablation physics are computationally expensive, while lower-fidelity aerodynamic models can be evaluated rapidly but lack physical accuracy. Multi-fidelity learning provides a mechanism to exploit the abundance of inexpensive low-fidelity simulations while incorporating the accuracy

of sparse high-fidelity CFD data. This approach is particularly attractive for surrogate modeling in design studies, where rapid prediction of flow fields across large parameter spaces is required.

## 4.1 Multi-Fidelity DeepONet

The Multi-Fidelity Deep Operator Network (MF-DeepONet) extends the standard DeepONet framework to simultaneously learn from multiple data fidelities [22]. The key idea of MF-DeepONet is to share information between fidelity levels by learning a common representation of the output function space while allowing separate components to model fidelity-specific discrepancies. While a conventional DeepONet seeks to approximate a single operator

$$\mathcal{G}_H : f_H \mapsto \mathcal{G}_H[f_H], \tag{23}$$

a multi-fidelity formulation jointly approximates operators corresponding to low- and high-fidelity data,

$$\mathcal{G}_L : f_L \mapsto \mathcal{G}_L[f_L], \qquad \mathcal{G}_H : f_H \mapsto \mathcal{G}_H[f_H]. \tag{24}$$

The MF-DeepONet architecture incorporates two sets of branch networks, one for each fidelity, which share a common trunk network to enforce a shared representation of the output function space (see Figure 1. This shared trunk encodes physical correlations across fidelities, while separate branches capture fidelity-specific discrepancies.

Let $\mathcal{B}_L(f_L)$ and $\mathcal{B}_H(f_H)$ denote the branch network outputs correspond-

ing to low- and high-fidelity inputs, respectively, and let $\mathcal{T}(x)$ denote the trunk network output evaluated at spatial or parametric location $x$. The MF-DeepONet output is then given by

$$\hat{\mathcal{G}}_L(x) = \mathcal{B}_L(f_L) \cdot \mathcal{T}(x), \tag{25}$$

$$\hat{\mathcal{G}}_H(x) = [\rho\, \mathcal{B}_L(f_L) + \delta\, \mathcal{B}_H(f_H)] \cdot \mathcal{T}(x), \tag{26}$$

where $\rho$ and $\delta$ are trainable coefficients controlling the degree of information transfer between the two fidelities. The term $\rho\, \mathcal{B}_L(f_L)$ introduces the low-fidelity correlation, while $\delta\, \mathcal{B}_H(f_H)$ captures the high-fidelity residual correction.

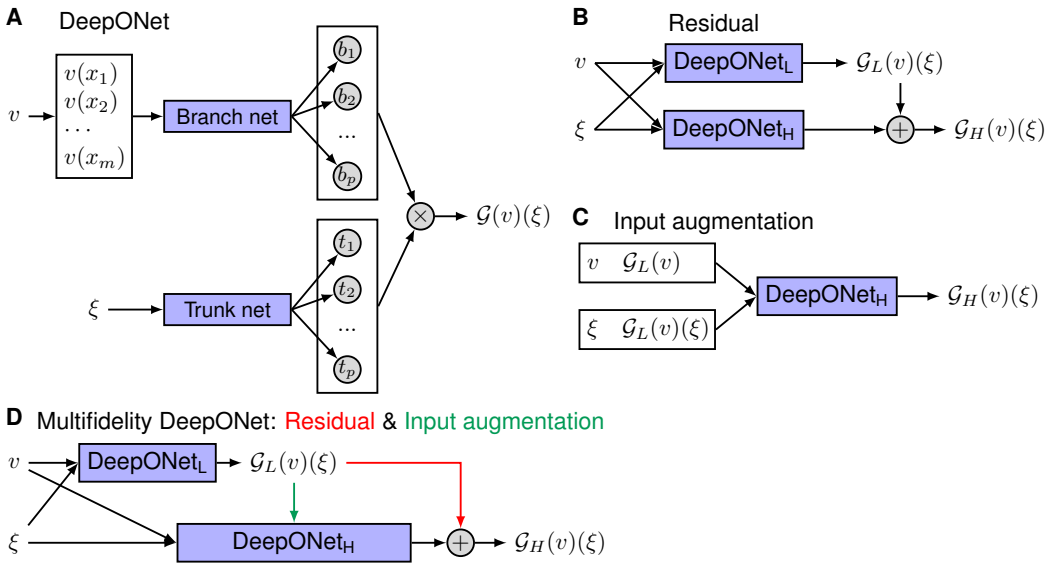



Figure 1: Architectures of DeepONet and MF-DeepONet. Original figure in [22]. **(A)** Overview of the DeepONet architecture. **(B)** Residual learning approach where DeepONet predicts the discrepancy between high and low fidelity solutions. **(C)** Input augmentation approach where the low-fidelity solution is provided as an additional input to the network. **(D)** Input-augmentation and residual learning approach, which is used in this study.

For the DeepONet surrogate models developed in this work, the branch network encodes the operating condition associated with each sample, represented by the freestream Mach number and altitude (in Figure 1, this is $v$). The trunk network encodes the spatial dependence of the solution through the coordinates of points on the sphere-cone surface mesh (in Figure 1, this is $\xi$). This model will be call the single-fidelity (SF) model. In the multi-fidelity (MF) architecture, the branch input is further augmented by the low-fidelity model output (in Figure 1, this is $\mathcal{G}_L(v)(\xi)$), enabling the high-fidelity predictor to learn a correction informed by both the freestream condition and the low-fidelity approximation. This will be called the MF model.

## 4.2 Loss Function Formulation

Training the MF-DeepONet involves jointly minimizing errors across both fidelity levels while maintaining consistency between fidelity levels. A typical multi-fidelity loss function is constructed as a weighted combination of fidelity-specific objectives:

$$\mathcal{L}_{\text{MF}} = \lambda_L \mathcal{L}_L + \lambda_H \mathcal{L}_H + \lambda_C \mathcal{L}_{\text{corr}}, \tag{27}$$

where

$$\mathcal{L}_L = \frac{1}{N_L} \sum_{i=1}^{N_L} \left\| \mathcal{G}_L[f_L^{(i)}](x) - \hat{\mathcal{G}}_L[f_L^{(i)}](x) \right\|^2, \tag{28}$$

$$\mathcal{L}_H = \frac{1}{N_H} \sum_{i=1}^{N_H} \left\| \mathcal{G}_H[f_H^{(i)}](x) - \hat{\mathcal{G}}_H[f_H^{(i)}](x) \right\|^2, \tag{29}$$

**Algorithm 1** Augmented Residual Multi-Fidelity DeepONet

**Require:** $\{\hat{U}^\ell, X^\ell, \hat{Y}^\ell\}$, $\{\hat{U}^h, X^h, \hat{Y}^h\}$, weights $w_\ell, w_h$, epochs $E$
*Inputs normalized jointly across fidelities via min-max scaling.*

1: Initialize LF branch/trunk $(\mathcal{B}^\ell, \mathcal{T}^\ell)$ and HF branch/trunk $(\mathcal{B}^h, \mathcal{T}^h)$ with Glorot init
2: **for** $e = 1$ **to** $E$ **do**
3: *LF prediction at LF and HF coordinates:*

$$\hat{Y}^\ell_{\text{pred}} = \mathcal{B}^\ell(\hat{U}^\ell)\,\mathcal{T}^\ell(X^\ell)^\top, \qquad \tilde{Y}^h = \mathcal{B}^\ell(\hat{U}^h)\,\mathcal{T}^\ell(X^h)^\top$$

4: *Augment HF input with LF correction:*

$$\hat{U}^h_{\text{aug}} = \left[\, \hat{U}^h \parallel \tilde{Y}^h \,\right]$$

5: *HF residual and final prediction:*

$$\hat{Y}^h_{\text{pred}} = \tilde{Y}^h + \mathcal{B}^h(\hat{U}^h_{\text{aug}})\,\mathcal{T}^h(X^h)^\top$$

6: *Weighted MSE loss and AdamW update:*

$$\mathcal{L} = w_\ell \left\| \hat{Y}^\ell_{\text{pred}} - \hat{Y}^\ell \right\|^2 + w_h \left\| \hat{Y}^h_{\text{pred}} - \hat{Y}^h \right\|^2$$

7: $\theta \leftarrow \theta - \nabla_\theta \mathcal{L}$ (AdamW)
8: **end for**
9: **return** $\theta^*$

and $\mathcal{L}_{\text{corr}}$ enforces correlation between fidelity levels, typically formulated as

$$\mathcal{L}_{\text{corr}} = \frac{1}{N_C} \sum_{i=1}^{N_C} \left\| \hat{\mathcal{G}}_H[f_H^{(i)}](x) - \alpha\, \hat{\mathcal{G}}_L[f_L^{(i)}](x) \right\|^2. \tag{30}$$

The weighting parameters $\lambda_L, \lambda_H, \lambda_C$ control the contribution of each fidelity to the total loss. A common approach is to increase $\lambda_H$ during later training stages, progressively emphasizing high-fidelity accuracy after the low-fidelity representation has been captured.

In practice, adaptive or residual-based weighting schemes can be employed to dynamically balance fidelity contributions, ensuring that model updates prioritize the regions and fidelities most relevant to improving generalization. This hybrid loss formulation allows the MF-DeepONet to exploit the abundant structure of low-fidelity data while refining its predictions through sparse high-fidelity samples, yielding a model that is both data-efficient and physically consistent.

## 4.3 Overview of Loss Weighting Strategies

This section reviews loss weighting strategies in both *single-fidelity* and *multi-fidelity* settings.

### 4.3.1 Single-Fidelity Loss Weighting

In the single-fidelity setting, the training objective is typically expressed as

$$\mathcal{L}(\theta) = \lambda_r \mathcal{L}_{\text{res}}(\theta) + \lambda_b \mathcal{L}_{\text{bc}}(\theta) + \lambda_d \mathcal{L}_{\text{data}}(\theta), \tag{31}$$

where $\mathcal{L}_{\text{res}}$ enforces the governing partial differential equations, $\mathcal{L}_{\text{bc}}$ enforces boundary or initial conditions, and $\mathcal{L}_{\text{data}}$ penalizes discrepancies with observed or simulated data. The coefficients $\lambda_r$, $\lambda_b$, and $\lambda_d$ determine the relative importance of each term.

Early PINN formulations typically relied on fixed, manually tuned loss weights. While simple, this approach is highly problem-dependent and often requires extensive hyperparameter tuning.

### 4.3.2 Adaptive and Residual-Based Weighting

To mitigate sensitivity to fixed loss weights, several adaptive strategies have been proposed. One prominent approach is *residual-based attention*, in which the contribution of the PDE residual is adaptively reweighted based on its magnitude during training. Regions with large residuals are emphasized, leading to improved convergence and stability, particularly for stiff or multiscale problems [1].

An alternative viewpoint frames PINN training as a multitask learning problem, where each loss term is treated as a competing task. Adaptive weighting schemes then seek to balance optimization rates or gradient norms across tasks, improving robustness and reducing training pathologies [23, 8].

More recently, residual-based weighting has been embedded into variational and risk-aware formulations, providing a unified theoretical framework applicable to both PINNs and neural operators [32].

### 4.3.3 Multi-Fidelity Loss Weighting

In multi-fidelity PINNs and neural operators, the loss function must additionally balance multiple data sources of differing fidelity, accuracy, and computational cost. A common formulation is

$$\mathcal{L}(\theta) = \lambda_r \mathcal{L}_{\text{res}}(\theta) + \sum_{k=1}^{K} \lambda_d^{(k)} \mathcal{L}_{\text{data}}^{(k)}(\theta), \tag{32}$$

where $\mathcal{L}_{\text{data}}^{(k)}$ corresponds to the $k$-th fidelity level (e.g., low- and high-fidelity simulations or measurements), and $\lambda_d^{(k)}$ controls its influence.

Several early multi-fidelity PINN approaches adopt fixed fidelity weights, chosen to reflect trust in each data source or estimated noise levels. One study explicitly introduces tunable weights between low- and high-fidelity data and study their influence in noisy and data-scarce regimes [28]. Similarly, another study emphasized cost-aware fidelity weighting, highlighting the trade-off between abundant but biased low-fidelity data and scarce but accurate high-fidelity data [24]. More recent approaches introduce adaptive fidelity weighting strategies, in which the relative influence of different fidelities evolves during training. Curriculum-style schemes typically emphasize low-fidelity data during early optimization stages and progressively increase the weight of high-fidelity data to refine the solution [27].

Across both single and multi-fidelity settings, loss weighting plays a critical role in training effectiveness. While fixed weights remain widely used, recent approaches increasingly favor adaptive and residual-based strategies that dynamically adjust loss contributions based on optimization difficulty, fidelity

level, or uncertainty. In multi-fidelity contexts, appropriate weighting between data sources is particularly important, as an imbalance can prevent the model from effectively exploiting high-fidelity information.

# 5 Multifidelity CFD Data

Two different solvers, NASA FUN3D and CBAero, were used to generate high-fidelity and low-fidelity data for steady-state flow over a sphere–cone configuration, respectively. A sphere–cone consists of a spherical forebody smoothly joined to a conical afterbody and is commonly used as an idealized geometry for atmospheric entry vehicles due to its simple axisymmetric shape and its ability to capture the dominant aerodynamic and aerothermodynamic features of blunt-body reentry flows. The datasets were generated across a range of freestream Mach numbers and altitudes representative of hypersonic atmospheric entry conditions. The sampling intervals and number of cases used for the high- and low-fidelity datasets are summarized in Table 1.

The following subsections describe the numerical tools used to generate the high and low-fidelity aerodynamic datasets, as well as the pre-processing steps applied prior to training the neural operator models.

Table 1: Mach number and altitude ranges, sampling intervals, and dataset sizes for high- and low-fidelity simulations.

| | Mach Number | | | Altitude (km) | | | |
|---|---|---|---|---|---|---|---|
| Dataset | Min | Max | $\Delta M$ | Min | Max | $\Delta h$ | # Cases |
| High-Fidelity | 15 | 30 | 5 | 30 | 50 | 10 | 12 |
| Low-Fidelity | 10 | 35 | 1 | 20 | 60 | 2 | 336 |

## 5.1 Data Generation

### 5.1.1 High-Fidelity Solver: NASA FUN3D

High-fidelity aerodynamic data for this study were generated using NASA's FUN3D computational fluid dynamics solver. FUN3D is a state-of-the-art unstructured finite-volume code developed at NASA Langley Research Center for the simulation of compressible viscous flows over complex geometries [4, 3]. The solver advances the compressible Navier–Stokes equations on mixed-element unstructured meshes using an implicit upwind discretization and has been extensively validated for hypersonic entry, descent, and landing applications [9]. For this work, simulations were performed using FUN3D version 14.02.

The generic gas path formulation of FUN3D was used, which solves the Navier–Stokes equations with optional models for turbulence and thermochemical nonequilibrium [3]. For hypersonic entry environments, the solver supports multi-species reacting flows with separate translational and vibrational temperatures. In the present study, a two-temperature nonequilibrium model was employed to represent high-temperature gas effects surrounding the vehicle. The reacting species set was selected to be consistent with a pure carbon ablator and included the species

$$\mathrm{N_2, N, O_2, O, NO, N^+, O^+, N_2^+, O_2^+, NO^+, e^-, C, C^+, CO, CO_2, C_2, C_3, CN.} \tag{33}$$

Reaction rates and thermodynamic properties were computed using the default

models provided in FUN3D [3]. Ablation of the thermal protection system was represented using an equilibrium char assumption, with the vehicle surface modeled as graphite.

The configuration considered in this work is a 7-degree half-angle sphere–cone representative of a generic atmospheric entry vehicle. Due to geometric symmetry, only a quarter of the vehicle was modeled to reduce computational cost. Simulations were conducted at zero angle of attack over a range of Mach numbers and altitudes representative of hypersonic reentry conditions. Figure **??** illustrates the vehicle geometry, computational mesh, and an example Mach number contour.

All CFD simulations were performed by Draper Laboratory using computing resources on AWS GovCloud. Each simulation used approximately 768 processor cores and required roughly 12 hours of wall-clock time to converge, highlighting the significant computational expense associated with generating high-fidelity training data for the surrogate modeling framework developed in this work.

The resulting CFD solutions serve as the high-fidelity dataset used to train and evaluate the neural operator models developed in this work. For each simulated flight condition, FUN3D provides detailed spatial fields of aerodynamic and thermodynamic quantities over the vehicle surface and within the surrounding flowfield. These solutions capture the coupled effects of compressibility and high temperature chemistry structure that arise in hypersonic atmospheric entry. Due to the significant computational expense of these simulations, the number of high-fidelity cases that can be generated is limited.

Consequently, the FUN3D dataset provides accurate but sparse supervision for the learning framework, while the lower-cost CBAero simulations described in Section 5.1.2 are used to augment the dataset with a larger number of approximate flow realizations.

### 5.1.2 Low-Fidelity Solver: CBAero

In addition to the high-fidelity CFD simulations performed with FUN3D, a lower-cost aerodynamic model is used to generate additional training data for the multi-fidelity framework. For this purpose, the engineering-level aerodynamic analysis tool CBAero is employed. CBAero was developed under NASA's Second Generation Launch Vehicle Program to support conceptual and preliminary vehicle design [33, 16]. The solver belongs to a class of rapid aerodynamic modeling approaches based on classical Newtonian impact theory, shock–expansion relations, and Prandtl–Meyer flow theory [33]. These methods estimate aerodynamic loads using local surface geometry and freestream conditions rather than resolving the full three-dimensional flow field, allowing aerodynamic trends to be evaluated rapidly across large parameter spaces.

Unlike high-fidelity computational fluid dynamics (CFD) solvers that solve the governing equations throughout a volumetric mesh, CBAero evaluates aerodynamic quantities directly on a surface representation of the vehicle [**?**]. Aerodynamic pressures are computed panel-by-panel using local surface inclination methods, which provide computational efficiency but do not capture globally coupled flow interactions or complex viscous phenomena. Consequently, CBAero is treated in this work as a low-fidelity aerodynamic model

that provides trend-correct approximations of hypersonic flow behavior at a fraction of the computational cost of CFD.

For the present study, the same outer mold line surface mesh used in the FUN3D simulations is employed in CBAero. However, the geometry is represented using triangulated surface elements to satisfy the solver's requirements. Aerodynamic cases are evaluated at the same Mach numbers and angles of attack used in the high-fidelity simulations.

A key difference between the two solvers lies in their freestream parameterization. FUN3D directly accepts Mach number, altitude, and angle of attack, with atmospheric properties internally determined from altitude. In contrast, CBAero requires Mach number, dynamic pressure, and angle of attack. To ensure consistent freestream conditions across solvers, the dynamic pressure corresponding to each Mach–altitude pair was computed prior to running CBAero. For a given altitude, the freestream density $\rho_\infty$ and speed of sound $a_\infty$ were obtained from the standard atmosphere, and the freestream velocity was evaluated as $V_\infty = Ma_\infty$. The dynamic pressure was then computed as

$$q_\infty = \tfrac{1}{2}\rho_\infty V_\infty^2. \tag{34}$$

This conversion ensures that both solvers are evaluated under physically equivalent freestream conditions despite their differing input parameterizations.

## 5.2 Data Processing

### 5.2.1 Normalization and Scaling of the Data

Prior to training the neural operator models, all input features were normalized to improve numerical conditioning during optimization. The aerodynamic and thermodynamic variables contained in the dataset span several orders of magnitude due to variations in Mach number, altitude, and local flow conditions. Without appropriate scaling, variables with large numerical ranges can dominate gradient updates and degrade training stability [12].

In this work, min–max normalization was applied to all input variables, mapping each feature to the interval $[0, 1]$. The normalized variables include the freestream input parameters (Mach number and altitude) as well as the flowfield quantities predicted by the CFD solvers: pressure, density, translational temperature, and vibrational temperature. For a given variable $x$, the normalized value $\tilde{x}$ was computed as

$$\tilde{x} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}, \tag{35}$$

where $x_{\min}$ and $x_{\max}$ denote the minimum and maximum values of that variable within the dataset. This transformation preserves the relative structure of the data while ensuring that all features contribute comparably during gradient-based optimization.

For the multi-fidelity dataset, normalization was performed jointly across the combined high-fidelity and low-fidelity data rather than independently for each fidelity level. Specifically, the global minimum and maximum values

were computed using the union of both datasets and applied consistently to all samples. This approach ensures that the two fidelity levels remain on a common numerical scale, which is essential for effective information transfer in the multi-fidelity learning framework. By enforcing consistent scaling across fidelity levels, the neural operator can learn relationships between the low- and high-fidelity solutions without introducing artificial discrepancies caused by inconsistent feature ranges.

## 5.3 Dataset Analysis

### 5.3.1 Low and High Fidelity Data Correlation

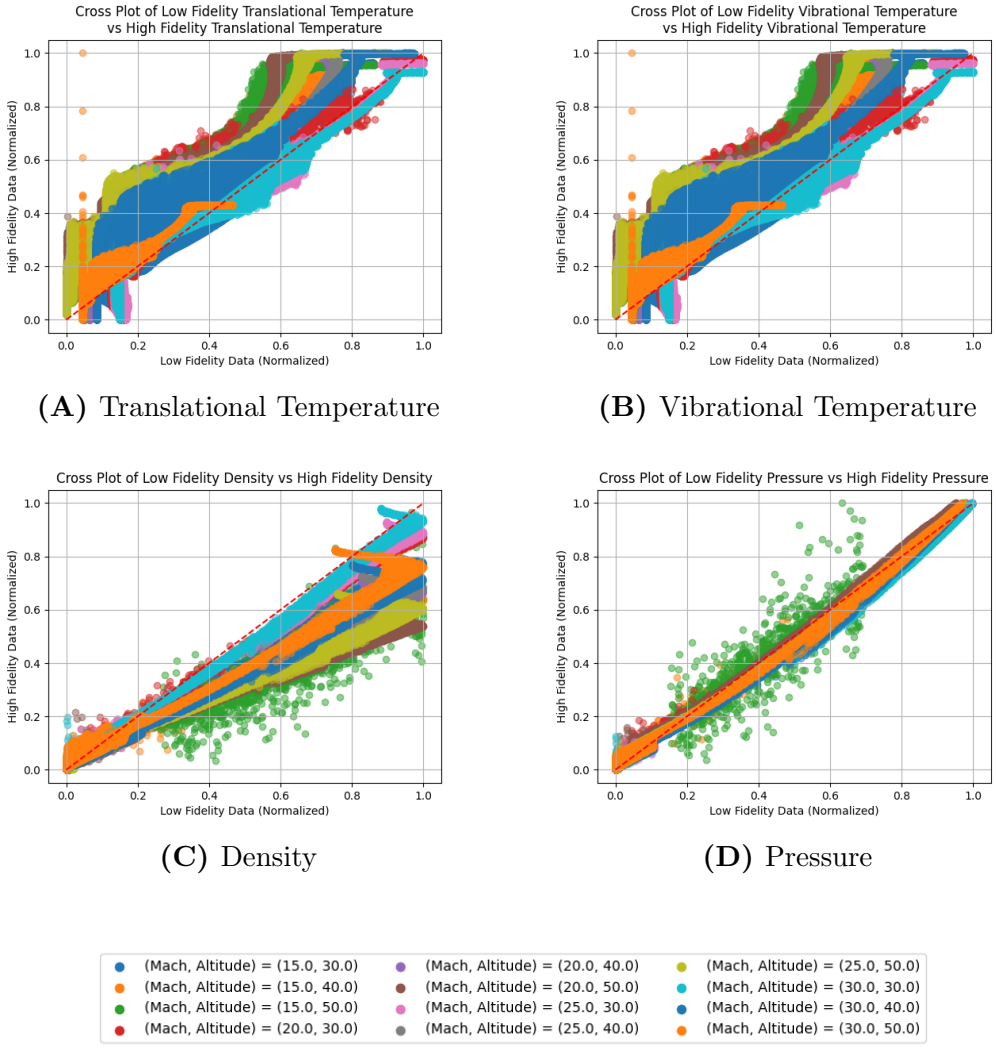
The potential benefit of multi-fidelity learning can often be assessed a priori by analyzing the statistical relationship between low- and high-fidelity datasets. Cross-plots and correlation measures provide insight into whether the low-fidelity data carries informative structure about the high-fidelity response; stronger correlation generally indicates greater potential performance gains from multi-fidelity modeling [14, 25].

In addition to statistical dependence, the relative distribution of the parameter space between fidelity levels is critical for effective information transfer. The low-fidelity dataset spans the same Mach–altitude distribution as the high-fidelity simulations and extends modestly beyond those bounds. Within the high-fidelity parameter envelope, low-fidelity samples are significantly denser, providing improved coverage of the input space while preserving alignment between fidelity levels. This design enables the neural operator to learn dominant global trends from abundant low-cost simulations, while sparse high-fidelity

solutions anchor the model to physically accurate flow behavior. The slight extension of the low-fidelity parameter space further promotes robustness by exposing the model to nearby out-of-distribution conditions, improving generalization without requiring additional expensive high-fidelity computations.

As illustrated in Figure 2, the cross-plots demonstrate a clear positive correlation between the low- and high-fidelity datasets for all thermodynamic quantities considered. In each case, the data points cluster around the diagonal reference line, indicating that increases in the low-fidelity predictions correspond to consistent increases in the high-fidelity responses. The pressure and density fields exhibit particularly strong linear alignment, with relatively tight clustering across Mach–altitude conditions, suggesting that the low-fidelity model captures much of the dominant physical behavior. The translational and vibrational temperatures also show a clear monotonic relationship, though with moderately increased dispersion, indicating the presence of nonlinear discrepancies while still preserving strong statistical dependence.

Overall, these figures confirm that the low-fidelity data retains significant informative structure with respect to the high-fidelity solutions. The observed correlation across all variables supports the assumption that the low-fidelity model provides a meaningful prior approximation of the high-fidelity response. Consequently, these results provide a priori justification for the use of multi-fidelity learning, as the demonstrated inter-fidelity correlation suggests strong potential for performance gains through information transfer between fidelity levels.




**(A)** Translational Temperature

**(B)** Vibrational Temperature

**(C)** Density

**(D)** Pressure

Figure 2: Cross-plots comparing normalized low- and high-fidelity predictions for translational temperature, vibrational temperature, density, and pressure. The dashed red line denotes $y = x$.

Table 2: Comparison of high-fidelity and low-fidelity aerodynamic solvers used for data generation

| **Attribute** | **High-Fidelity Solver: NASA FUN3D** | **Low-Fidelity Solver: CBAero** |
|---|---|---|
| Solver type | Full computational fluid dynamics (CFD) solver | Engineering-level aerodynamic analysis tool |
| Governing equations | Reynolds Averaged Navier-Stokes Equations with 18 species Park chemistry model and carbon ablation | No governing PDEs solved; relies on algebraic and semi-empirical models |
| Spatial domain | Three-dimensional volumetric flow field | Surface only |
| Numerical method | Finite volume method | Independent local panel methods (Modified Newtonian, Tangent Wedge, Tangent Cone) |
| Flow coupling | Globally coupled solution across the computational domain | No global coupling; panels treated independently |
| Viscous effects | Resolved directly (boundary layers) | Approximated using engineering correlations (e.g., Eckert reference enthalpy method) |
| Velocity field | Computed as part of the coupled flow solution | Reconstructed algebraically from surface normals and freestream conditions |
| Geometry handling | 2,534,400 brick elements for volume and 21,120 quad elements for surface | 42,240 triangular elements (elements from high-fidelity surface mesh divided to two triangles each) |
| Computational cost | High (hours to days per configuration) | Low (seconds to minutes per configuration) |

# 6 Results

This section presents a quantitative and qualitative comparison between the single-fidelity (SF) neural operator and the proposed multi-fidelity (MF) neural operator. Predictions are evaluated against high-fidelity CFD solutions for pressure ($p$), density ($\rho$), translational temperature ($T_t$), and vibrational temperature ($T_v$). Both global relative $L_2$ errors and spatially resolved error distributions are examined to assess overall accuracy and localized performance.

## 6.1 Global Error Comparison

Table 4 summarizes the relative $L_2$ errors for the single- and multi-fidelity models. The most significant improvement is observed in the pressure field, where the error is reduced from 9.77% to 2.04%, corresponding to a substantial improvement in predictive accuracy. Density error is also reduced from 4.67% to 3.49%, indicating consistent benefit from incorporating low-fidelity information.

For the temperature fields, improvements are more modest. The translational temperature error decreases slightly from 5.18% to 4.98%, while the vibrational temperature error shows a small increase from 4.70% to 4.92%. Overall, these results indicate that multi-fidelity learning provides the largest gains for pressure and density, while temperature predictions are already reasonably captured by the single-fidelity model.

## 6.2 Computation Time Comparison

Table 3 compares the computational cost associated with the numerical solvers and neural operator surrogates. As expected, the surrogate models incur an upfront training cost, with the MF DeepONet requiring a longer average training time (262.41 s) than the SF DeepONet (73.87 s) due to the additional complexity associated with learning from both fidelity levels. However, once training is complete, both surrogate models provide extremely rapid predictions. The SF DeepONet achieves an average inference time of $2.41 \times 10^{-4}$ s, while the MF DeepONet requires $1.69 \times 10^{-2}$ s per evaluation.

Although the MF model is more expensive to train and evaluate than the SF model, its inference cost remains negligible compared to the cost of generating new high-fidelity CFD solutions. This trade-off is favorable for applications involving repeated evaluations, such as parametric studies or design-space exploration, where the one-time training expense can be amortized over many predictions. These results further support the use of neural operator surrogates as efficient alternatives to repeated numerical simulation in hypersonic flow modeling.

Table 3: Comparison of average training, inference, and solution times across methods. GPU used for DeepONet training was the Nvidia RTX 3090. NASA FUN3D computations were conducted and provided by Draper Laboratory and took $768 \times 12$ hours for one solution. CBAero computations were conducted on a Dell Latitude 5400 with an Intel i7-8665U.

| | NASA FUN3D | CBAero | SF DeepONet | MF DeepONet |
|---|---|---|---|---|
| Average Training Time | – | – | 73.87 s | 262.41 s |
| Average Inference/Solution Time | 9,216 core-hours | 1 core-minute | $2.41 \times 10^{-4}$ s | $1.69 \times 10^{-2}$ s |

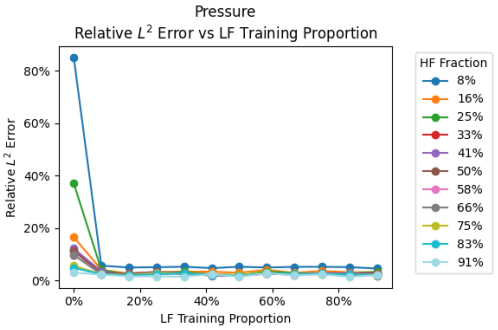
## 6.3 Sensitivity to Low- and High-Fidelity Training Proportions

To further investigate the impact of multi-fidelity learning, a sensitivity study was conducted by varying the proportion of low-fidelity (LF) and high-fidelity (HF) training data. Figures 3A-3D show the relative $L_2$ error as a function of the LF training proportion for pressure, density, translational temperature, and vibrational temperature. Each curve corresponds to a fixed fraction of high-fidelity training data.

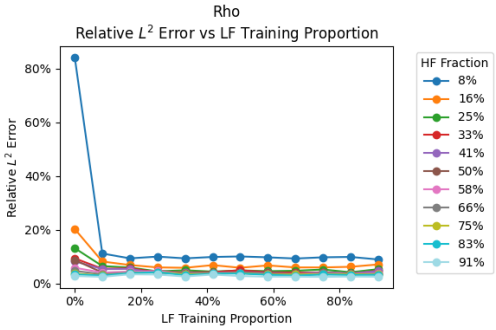
Across all variables, the largest errors occur when no low-fidelity data is used. In this regime, the model relies solely on sparse high-fidelity samples, resulting in significantly reduced predictive accuracy, particularly when the available HF fraction is small. For example, when only 8% of the HF data is used, the pressure error exceeds 80% without LF augmentation.

As the proportion of LF training data increases, the prediction error decreases rapidly. For pressure and density, even modest LF inclusion (approximately 10–20%) dramatically stabilizes the model and reduces error to below 10%. This behavior indicates that the low-fidelity model captures strong structural trends in the flow field that are highly informative for training the neural operator.

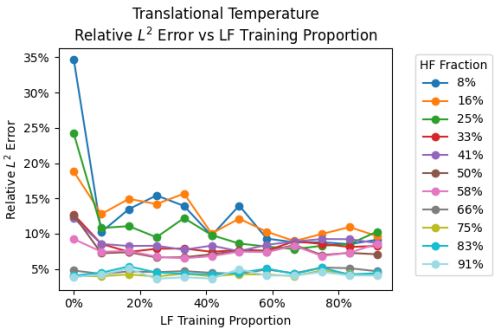
The temperature fields exhibit a similar but less pronounced trend. Translational and vibrational temperature errors decrease as LF data is incorporated, though the reduction is smaller than for pressure and density. This suggests that thermochemical effects governing the temperature fields are more weakly correlated between fidelity levels, which is illustrates in Figure 2 Thus,



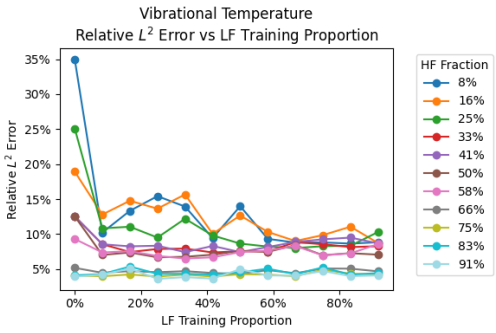

**(A)** Pressure relative $L_2$ error across varying high and low fidelity data proportions.




**(B)** Rho relative $L_2$ error across varying high and low fidelity data proportions.




**(C)** Translational temperature relative $L_2$ error across varying high and low fidelity data proportions.




**(D)** Vibrational temperature relative $L_2$ error across varying high and low fidelity data proportions.

Figure 3: Relative L2 error vs proportion of low fidelity training data.

the degree of information transfer.

Overall, these results confirm that incorporating even modest amounts of low-fidelity data significantly improves model stability and predictive accuracy, particularly in regimes where high-fidelity training data is sparse.

## 6.4 Surface Field Comparisons

Comparisons between CFD solutions and neural operator inferences in Figure 6 demonstrate that both models capture the dominant hypersonic flow features, including stagnation heating at the nose and gradual expansion along the cone surface . The SF and MF networks reproduce the overall magnitude and spatial distribution of $T_t$, $T_v$, $\rho$, and $p$, with discrepancies primarily localized near the nose and aft portion of the body.

The spatial distribution of prediction error is further examined using one-dimensional $L_1$ error profiles extracted along a surface-aligned slice from the nose ($x/L = 0$) to the aft ($x/L = 1$), as shown in Figure 9. These profiles provide a detailed comparison of error magnitude and localization between the SF and MF models.

For both pressure and density (Figures 9B and 9A), the largest errors occur near the stagnation region at the nose, where strong compression induced gradients are present. In this region, both models exhibit sharp error spikes; however, the MF model consistently reduces the peak magnitude relative to the SF model. Moving downstream along the body, the MF model maintains lower and more uniform error levels, while the SF model exhibits broader regions of elevated error, particularly in the mid-body region ($x/L \approx 0.2$–$0.6$).

Near the aft region ($x/L \to 1$), both models again show increased error due to strong gradients, though the MF model generally produces smaller peak values.

For translational temperature (Figure 9C), the error behavior is more complex. While the MF model reduces error near the stagnation region, it introduces larger deviations across portions of the mid-body, where the MF curve exhibits higher amplitude fluctuations compared to the SF model. This behavior indicates that, unlike pressure and density, temperature fields are strongly influenced by nonequilibrium thermochemical processes that are not captured by the low-fidelity model. Consequently, the cross-fidelity correlation is weaker, reducing the effectiveness of information transfer in the multi-fidelity framework.

A similar trend is observed for vibrational temperature (Figure 9D). The MF model shows improved accuracy near the nose but exhibits increased error in downstream regions relative to the SF model. In particular, the MF predictions display larger deviations across the mid-body, indicating that vibrational nonequilibrium effects are more difficult to capture through cross-fidelity correlation.

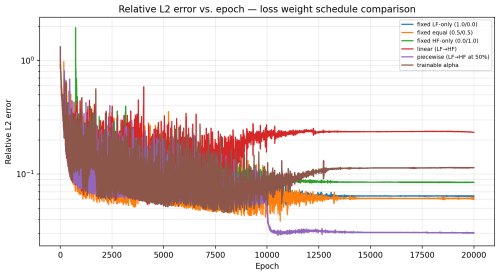
Overall, these results demonstrate that multi-fidelity learning provides the greatest benefit for flow quantities that are strongly governed by macroscopic flow features, such as pressure and density. In contrast, temperature fields, especially vibrational temperature, exhibit more complex nonlinear behavior that is less directly transferable from low-fidelity models. Nevertheless, the MF model reduces peak errors in critical regions such as the stagnation point,

indicating improved robustness in regions of strong flow gradients.

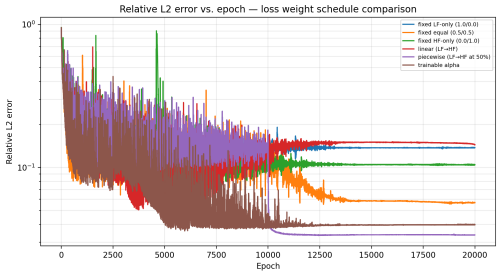
## 6.5 Multi-Fidelity Loss Weighting

The effect of loss weighting on MF-DeepONet training was investigated using several fixed and adaptive strategies. The fixed strategies included low-fidelity-only weighting, high-fidelity-only weighting, and equal weighting of the low- and high-fidelity losses. Two curriculum-style schedules were also considered: a linear schedule that gradually shifted emphasis from low-fidelity to high-fidelity data, and a piecewise schedule in which training initially emphasized low-fidelity data before switching to high-fidelity weighting at a prescribed epoch. In addition, a trainable weighting parameter $\alpha$ was introduced so that the relative contribution of the fidelity losses could be learned during optimization.

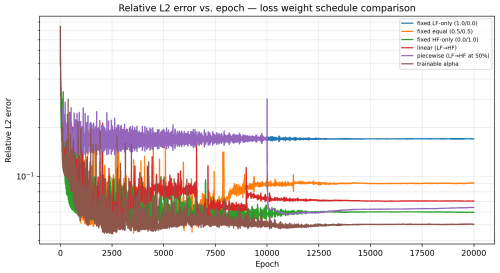
Figures 4A-4D show the relative $L_2$ error histories for pressure, density, translational temperature, and vibrational temperature under these weighting strategies. Several trends are evident. First, the weighting strategy has a substantial influence on both convergence behavior and final accuracy, particularly for the temperature variables. Second, purely low-fidelity weighting consistently performs poorly for pressure and density, indicating that low-fidelity data alone is insufficient for accurate recovery of the high-fidelity solution. Third, no single schedule is uniformly optimal across all variables. For pressure and density, strategies that retain strong high-fidelity influence, such as high-fidelity-only weighting or the trainable-$\alpha$ formulation, produce the smallest final errors. In contrast, the temperature variables benefit more



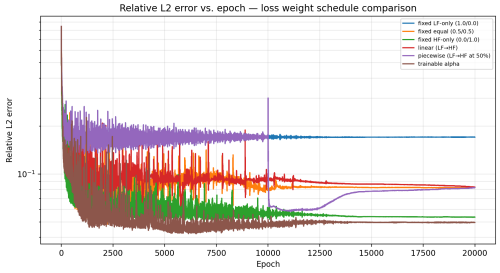

**(A)** Pressure relative $L_2$ error versus training epoch for different multi-fidelity loss-weighting strategies.




**(B)** Density relative $L_2$ error versus training epoch for different multi-fidelity loss-weighting strategies.

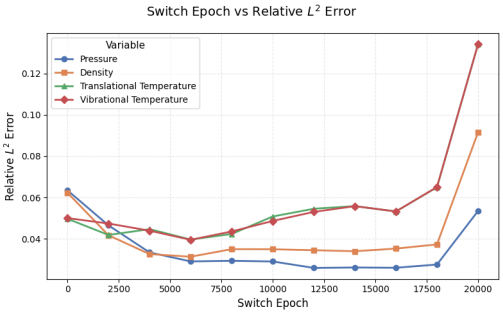



**(C)** Translational temperature relative $L_2$ error versus training epoch for different multi-fidelity loss-weighting strategies.

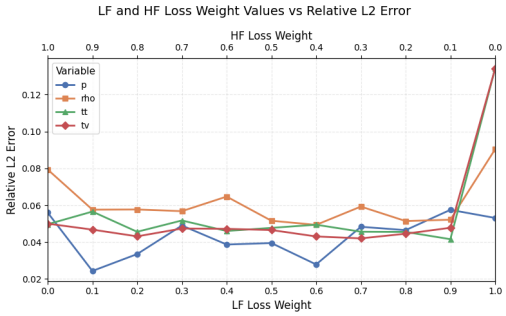



**(D)** Vibrational temperature relative $L_2$ error versus training epoch for different multi-fidelity loss-weighting strategies.

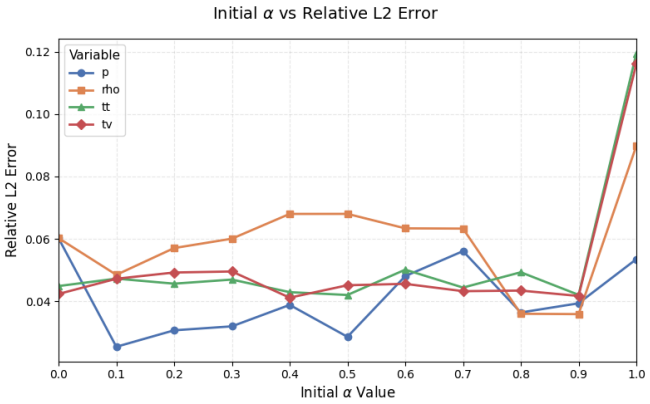
Figure 4: Comparison of relative $L_2$ error histories for several multi-fidelity loss-weighting strategies. The strategies include fixed low-fidelity-only weighting, fixed high-fidelity-only weighting, equal weighting, a linear low-to-high fidelity schedule, a piecewise low-to-high fidelity schedule, and a trainable weighting parameter $\alpha$.




**(A)** Effect of the switching epoch in the piecewise low-to-high fidelity schedule on final relative $L_2$ error.




**(B)** Effect of fixed low- and high-fidelity loss-weight combinations on final relative $L_2$ error.




**(C)** Effect of the initial value of the trainable weighting parameter $\alpha$ on final relative $L_2$ error.

Figure 5: Sensitivity of MF-DeepONet performance to different multi-fidelity loss-weighting choices. The plots show the influence of the switching epoch for the piecewise schedule, the fixed balance between low- and high-fidelity losses, and the initialization of the trainable weighting parameter $\alpha$.

from curriculum-style training, with the piecewise low-to-high fidelity schedule yielding the lowest final errors. This behavior suggests that pressure and density are more directly corrected by strong high-fidelity supervision, whereas the temperature fields benefit from first learning coarse low-fidelity structure before refinement with high-fidelity data.

The epoch histories also clarify differences in optimization behavior. The trainable-$\alpha$ strategy generally produces rapid early error reduction and competitive final performance, particularly for pressure, density, and translational temperature. The linear schedule is more stable than the abrupt piecewise switch, but it does not consistently achieve the lowest final error and performs especially poorly for vibrational temperature. The piecewise schedule exhibits a visible transient at the switching epoch, reflecting the abrupt change in loss weighting, but this temporary instability is followed by improved asymptotic accuracy for the temperature fields.

To examine the piecewise schedule in more detail, the switching epoch was varied and the resulting final relative $L_2$ errors are summarized in Figure 5A. The results indicate that the optimal switch time is variable-dependent. For pressure and density, the best performance is obtained when the transition to high-fidelity emphasis occurs after an initial low-fidelity training period but before the final stages of optimization. For translational and vibrational temperature, earlier switching generally yields lower errors than very late switching, while excessively delayed transitions lead to clear degradation for all variables. These results indicate that prolonged training under predominantly low-fidelity weighting can bias the model toward low-fidelity structure that is

difficult to correct later.

Figure 5B shows the effect of fixed low- and high-fidelity loss weights on the final relative $L_2$ error. Intermediate weight combinations generally outperform the extreme cases, demonstrating that a balanced contribution from both fidelity levels is beneficial. Purely low-fidelity weighting produces the largest errors, especially for density and the temperature variables. Purely high-fidelity weighting is more competitive, but still not optimal across all quantities. The pressure field shows a stronger sensitivity to the chosen weight ratio, whereas the temperature fields are relatively flat over a broad range of intermediate weights before degrading at the extremes. This again supports the conclusion that the value of low-fidelity information is variable-dependent and should not be incorporated with a fixed uniform weighting chosen a priori.

Finally, the sensitivity of the trainable-$\alpha$ approach to its initialization is shown in Figure 5C. Over a broad range of intermediate initial values, the final errors remain relatively stable, indicating that the trainable weighting formulation is reasonably robust. However, initialization at the extreme low-fidelity-dominated limit leads to substantial degradation for all variables. This behavior suggests that although the weighting parameter can adapt during training, the optimization still benefits from an initial balance that allows both fidelity levels to influence the early stages of learning.

Overall, these results show that multi-fidelity loss weighting is a critical design choice in MF-DeepONet training. The best strategy depends on the output variable of interest: pressure and density favor stronger high-fidelity supervision, while the temperature fields benefit more from staged low-to-high

fidelity training. Among the strategies considered, the trainable-$\alpha$ and piecewise schedules provide the most favorable balance between convergence and final accuracy, though their relative benefit depends on the physics represented by each predicted quantity.

## 6.6 Discussion

Taken together, the results show that the success of the multi-fidelity framework depends on two related factors: the degree of physical correlation between fidelity levels and the training strategy used to transfer information from the low-fidelity model to the high-fidelity prediction. The strongest evidence of this is observed in the pressure field, where the MF DeepONet reduces the relative $L^2$ error from 9.77% to 2.04%, with density also showing consistent improvement (see Table 4. These quantities appear to retain strong structural agreement across fidelity levels, allowing the low-fidelity data to provide a useful prior that the network can refine with sparse high-fidelity supervision. In contrast, the translational and vibrational temperatures show only modest improvement, and in the case of vibrational temperature a slight degradation, indicating that the corresponding fidelity gap is larger for thermochemical quantities than for macroscopic aerodynamic quantities.

This interpretation is reinforced by the sensitivity study on training data proportions illustrated in Figure 3. When the amount of high-fidelity data is small, the inclusion of even a modest amount of low-fidelity data leads to large reductions in error, especially for pressure and density. Thus, the main value of the low-fidelity dataset is not simply to increase sample count, but to provide

broad coverage of the parameter space and stabilize learning when high-fidelity supervision is sparse. However, the weaker improvement observed for the temperature variables indicates that low-fidelity information is not equally informative for all outputs. In the present problem, the engineering-level low-fidelity model captures large-scale aerodynamic trends more effectively than the detailed nonequilibrium thermochemical behavior represented in the high-fidelity CFD solutions.

The spatial error comparisons further clarify this point. Along the surface slice from nose to aft, the MF model reduces pressure and density errors over much of the body, especially away from the most difficult regions near the nose and aft end. For the temperature variables, however, the MF model shows mixed behavior: although error is reduced near the nose in some cases, larger deviations persist or emerge over other portions of the surface. This suggests that low-fidelity information is most beneficial when the target variable is governed primarily by geometric and compressive flow structure, and less beneficial when the target depends more strongly on fidelity specific thermochemical modeling.

The loss-weighting study provides an important extension of this interpretation. These results show that multi-fidelity learning is not determined solely by whether both fidelity levels are included, but also by how their contributions are scheduled during optimization. For pressure and density, strategies with stronger high-fidelity influence or adaptive weighting produced the best final errors, indicating that these quantities can benefit from low-fidelity guidance but still require sustained high-fidelity correction for best accuracy. For trans-

lational and vibrational temperature, curriculum-style strategies performed better, particularly piecewise schedules that first allowed the model to learn coarse low-fidelity structure and then shifted emphasis to the high-fidelity data. This indicates that, for variables with weaker fidelity correlation, the training process must more carefully manage the transfer from low to high fidelity information.

The switching epoch and fixed weight sweeps also show that excessive reliance on low-fidelity loss can be detrimental. If the model is trained too long with predominantly low-fidelity weighting, it becomes biased toward the low-fidelity solution manifold, making later correction more difficult. Conversely, using only high-fidelity loss discards useful low fidelity structure and reduces the data efficiency advantage of the multi-fidelity framework. The best results therefore arise from a balance: enough low-fidelity influence to provide global structure and regularization, but sufficient high-fidelity emphasis to recover the physics absent from the low-fidelity model. In this sense, the loss-weighting results demonstrate that multi-fidelity learning is fundamentally an optimization problem as well as a data-fusion problem.

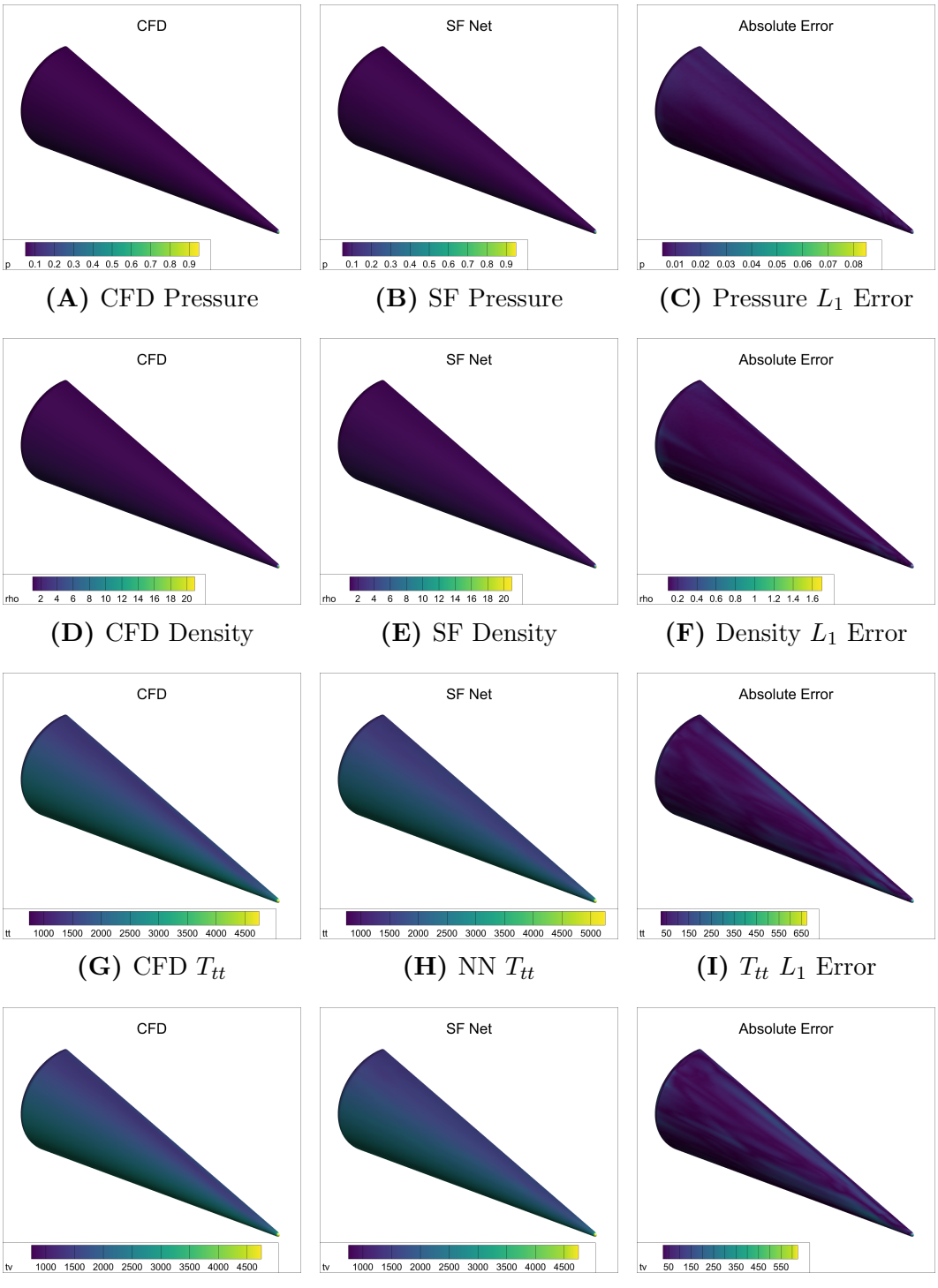
From a practical perspective, these findings are important for surrogate modeling of hypersonic flows. They show that low-fidelity aerodynamic simulations can substantially reduce the amount of expensive high-fidelity CFD data needed to obtain accurate predictions for quantities such as pressure and density, while still preserving extremely fast inference once the network is trained. At the same time, they indicate that temperature-related quantities, especially those associated with nonequilibrium effects, require more careful

treatment. For such variables, the design of the fidelity-transfer mechanism, including the weighting schedule, switching time, or adaptive coefficients, is critical to achieving robust performance.

Overall, the results support that multi-fidelity neural operators provide an effective and computationally efficient surrogate modeling strategy for hypersonic flowfields, but their performance is strongly variable-dependent and closely tied to the fidelity relationship embedded in the data. The low-fidelity model is most valuable when it captures the dominant structural trends of the high-fidelity solution, and the greatest gains are obtained when the training process is designed to exploit that structure without allowing low fidelity bias to dominate the final prediction.

Table 4: Relative $L_2$ error comparison between single-fidelity and multi-fidelity neural operator models.

| Quantity | Single-Fidelity Error (%) | Multi-Fidelity Error (%) |
|---|---|---|
| Pressure | 9.77 | 2.04 |
| Density | 4.67 | 3.49 |
| Translational Temperature | 5.18 | 4.98 |
| Vibrational Temperature | 4.70 | 4.92 |



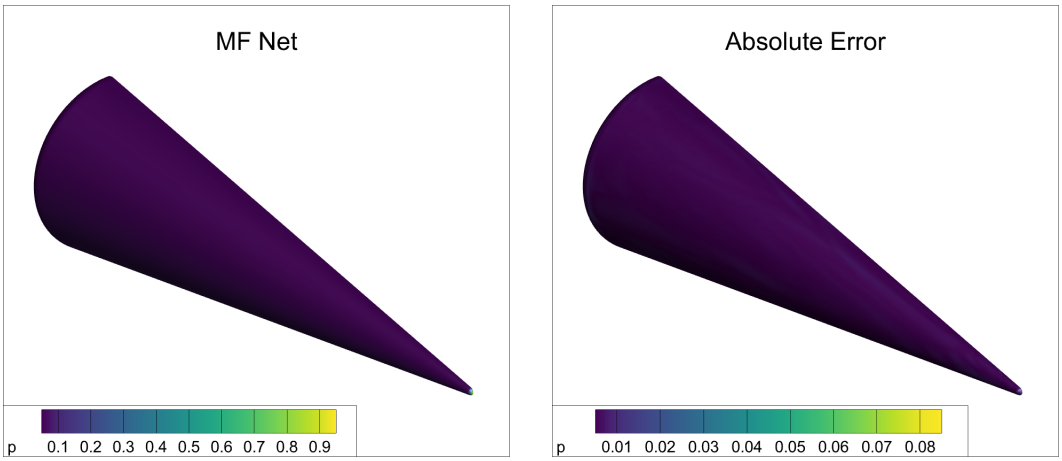

**(A)** CFD Pressure **(B)** SF Pressure **(C)** Pressure $L_1$ Error

**(D)** CFD Density **(E)** SF Density **(F)** Density $L_1$ Error

**(G)** CFD $T_{tt}$ **(H)** NN $T_{tt}$ **(I)** $T_{tt}$ $L_1$ Error

**(J)** CFD Vibrational Temperature **(K)** SF Vibrational Temperature **(L)** Vibrational Temperature $L_1$ Error

Figure 6: Comparison between high-fidelity CFD solutions and DeepONet inference on test sample inputs of Mach 20 and an altitude of 30km.



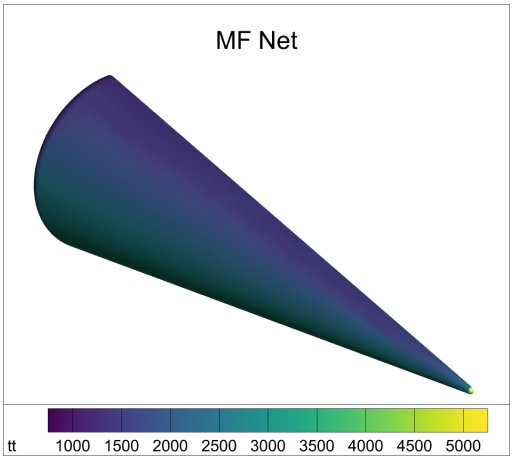

**(A)** MF Pressure

**(B)** Pressure $L_1$ Error

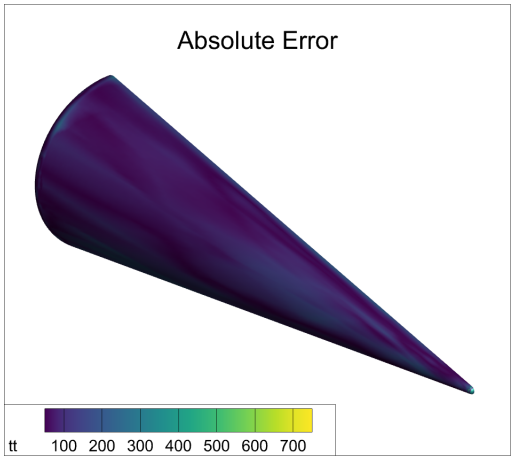
**(C)** MF Density

**(D)** Density $L_1$ Error

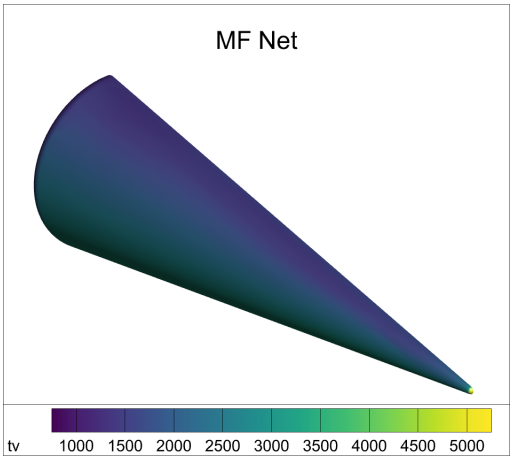
Figure 7: MF DeepONet inferences and absolute error on test sample inputs of Mach 20 and an altitude of 30km (pressure and density).



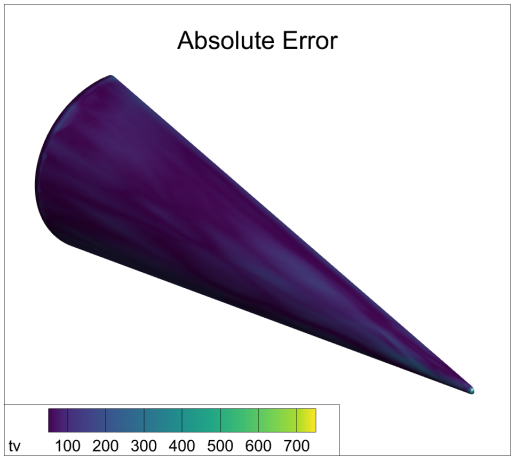

**(A)** MF Translational Temperature




**(B)** Translational Temperature $L_1$ Error

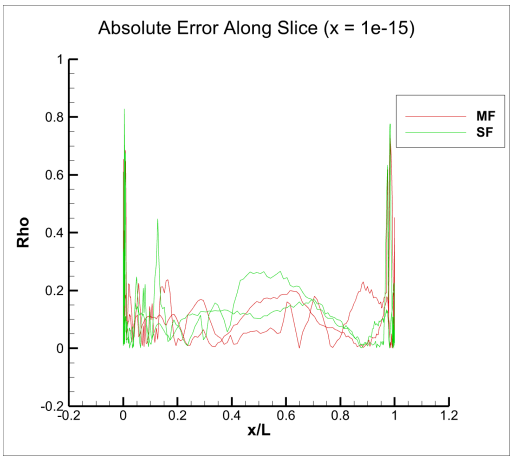



**(C)** MF Vibrational Temperature

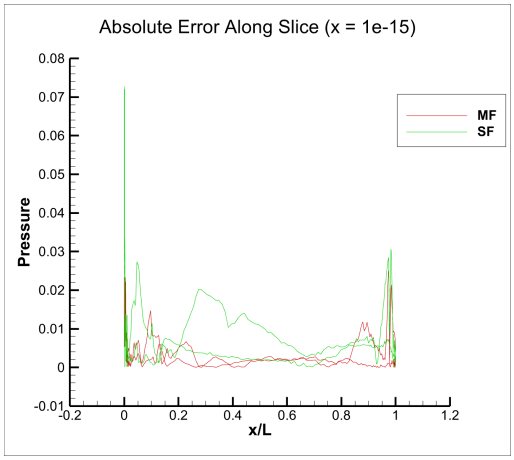



**(D)** Vibrational Temperature $L_1$ Error

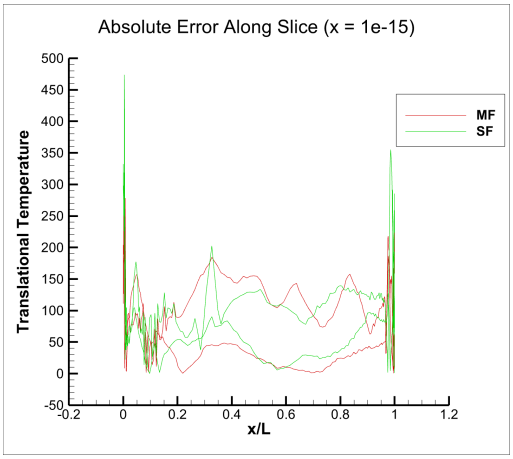
Figure 8: MF DeepONet inferences and absolute error on test sample inputs of Mach 20 and an altitude of 30km (translational and vibrational temperatures).



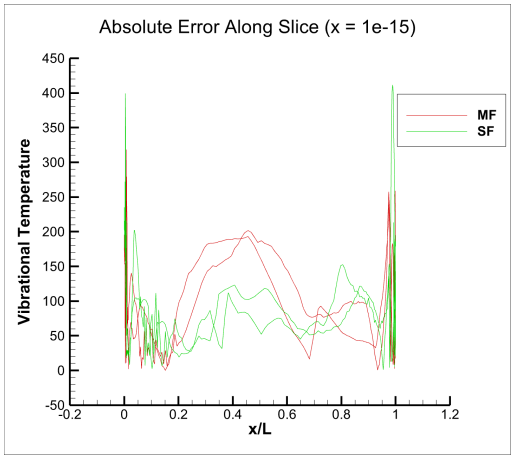

**(A)** Rho $L_1$ error along $x = 10^{-15}$ slice.




**(B)** Pressure $L_1$ error along $x = 10^{-15}$ slice.




**(C)** Translational temperature $L_1$ error along $x = 10^{-15}$ slice.




**(D)** Vibrational temperature $L_1$ error along $x = 10^{-15}$ slice.

Figure 9: Comparison of $L_1$ error of SF and MF models along $x = 10^{-15}$ slice for a Mach number of 20 and an altitude of 30 km. $\frac{x}{L}$ is the non-dimensional length of the sphere cone, where 0 is the nose and 1 is the aft.

# 7 Conclusion

This thesis investigated the use of neural operator methods as surrogate models for hypersonic aerothermodynamic simulations involving thermochemical nonequilibrium and carbon ablation. Accurate prediction of these flows is essential for atmospheric entry vehicle design, yet high-fidelity CFD simulations are computationally expensive due to the need to resolve strong compression, thin boundary layers, and complex chemical reaction networks. These challenges motivate the development of efficient surrogate models capable of approximating CFD solutions at significantly reduced cost.

In this work, DeepONets were applied to model steady-state laminar hypersonic flow over a generic sphere–cone configuration with an 18-species reacting flow model. Both single-fidelity and multi-fidelity neural operator architectures were investigated. High-fidelity training data were generated using the NASA FUN3D solver, while lower-cost aerodynamic data were obtained using the engineering-level CBAero code. Results demonstrate that neural operators can successfully approximate complex hypersonic flow fields, reproducing the dominant spatial structures in pressure, density, and temperature distributions. The multi-fidelity model significantly improved predictive accuracy for quantities strongly correlated across fidelity levels, particularly pressure and density, while temperature fields exhibited more modest improvements due to the increased complexity of thermochemical effects.

Sensitivity studies further showed that incorporating even modest amounts of low-fidelity data substantially improves model stability and prediction accuracy when high-fidelity training data are sparse. These findings highlight

the effectiveness of multi-fidelity neural operators for leveraging inexpensive aerodynamic models to augment limited high-fidelity CFD data. As a result, the proposed framework provides a promising approach for accelerating hypersonic flow simulations and enabling more efficient parametric exploration of atmospheric entry conditions.

Future research may extend this work in several directions. First, the present study focused on steady-state laminar flow; extending neural operator models to turbulent or unsteady hypersonic flows would significantly broaden their applicability. Second, more complex vehicle geometries could be investigated to evaluate the generalization capability of the surrogate model. Third, improved multi-fidelity training strategies, including adaptive fidelity weighting and active learning approaches for selecting high-fidelity simulations, may further enhance data efficiency. Finally, incorporating additional physics-based constraints into neural operator training may improve accuracy in regions with strong gradients, such as shock layers and stagnation regions.

Overall, this work demonstrates that multi-fidelity neural operators offer a viable and computationally efficient surrogate modeling strategy for hypersonic aerothermodynamic simulations, with potential applications in rapid design iteration and reduced-cost simulation workflows for atmospheric entry vehicle analysis.

# References

[1] Sokratis J. Anagnostopoulos, Juan Diego Toscano, Nikolaos Stergiopulos, and George Em Karniadakis. Residual-based attention and connection to information bottleneck theory in PINNs, July 2023. arXiv:2307.00379 [cs].

[2] William K. Anderson, Robert T. Biedron, Jan-Renee Carlson, Joseph M. Derlaga, Boris Diskin, Jr. Druyor, Cameron T., Peter A. Gnoffo, Dana P. Hammond, Kevin E. Jacobson, William T. Jones, Bil Kleb, Elizabeth M. Lee-Rausch, Yi Liu, Mark W. Lohry, Gabriel C. Nastac, Eric J. Nielsen, Emmett M. Padway, Michael A. Park, Christopher L. Rumsey, James L. Thomas, Kyle B. Thompson, Joshua J. Wagner, Aaron Walden, Li Wang, Stephen L. Wood, William A. Wood, and Xinyu Zhang. Fun3d manual: 14.2. NASA Technical Memorandum 20250002308, NASA Langley Research Center, 2025.

[3] William K. Anderson, Robert T. Biedron, Jan-Renée Carlson, Joseph M. Derlaga, Boris Diskin, Cameron T. Druyor Jr., Peter A. Gnoffo, Dana P. Hammond, Kevin E. Jacobson, William T. Jones, Bil Kleb, Elizabeth M. Lee-Rausch, Yi Liu, Gabriel C. Nastac, Eric J. Nielsen, Emmett M. Padway, Michael A. Park, Christopher L. Rumsey, James L. Thomas, Kyle B. Thompson, Aaron C. Walden, Li Wang, Stephen L. Wood, William A. Wood, and Xinyu Zhang. Fun3d manual: 14.0.2. NASA Technical Memorandum NASA/TM–20230004211, NASA Langley Research Center, Hampton, Virginia, November 2023.

[4] W.Kyle Anderson and Daryl L Bonhaus. An implicit upwind algorithm for computing turbulent flows on unstructured grids. *Computers & Fluids*, 23(1):1–21, January 1994.

[5] J.W. Bandler, R.M. Biernacki, Shao Hua Chen, P.A. Grobelny, and R.H. Hemmers. Space mapping technique for electromagnetic optimization. *IEEE Transactions on Microwave Theory and Techniques*, 42(12):2536–2544, December 1994.

[6] Qianying Cao, Somdatta Goswami, and George Em Karniadakis. Laplace neural operator for solving differential equations. *Nature Machine Intelligence*, 6(6):631–640, June 2024.

[7] Russell M Cummings. Aerothermodynamic Challenges of Hypersonic Flight. October 2025.

[8] Bo Gao, Ruoxia Yao, and Yan Li. Physics-informed neural networks with adaptive loss weighting algorithm for solving partial differential equations. *Computers & Mathematics with Applications*, 181:216–227, March 2025.

[9] Peter A. Gnoffo, William A. Wood, Bill Kleb, Stephen J. Alter, Jose Padilla, and Jeffery A. White. Functional Equivalence Acceptance Testing of FUN3D for Entry, Descent, and Landing Applications. In *21st AIAA Computational Fluid Dynamics Conference*, San Diego, CA, June 2013. American Institute of Aeronautics and Astronautics.

[10] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[11] Amanda A. Howard, Mauro Perego, George Em Karniadakis, and Panos Stinis. Multifidelity deep operator networks for data-driven and physics-informed problems. *Journal of Computational Physics*, 493:112462, November 2023.

[12] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.

[13] Su Jiang and Louis J. Durlofsky. Use of multifidelity training data and transfer learning for efficient construction of subsurface flow surrogate models. *Journal of Computational Physics*, 474:111800, February 2023.

[14] M. Kennedy. Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87(1):1–13, March 2000.

[15] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, January 2017. arXiv:1412.6980 [cs].

[16] David Kinney. Aero-Thermodynamics for Conceptual Design. In *42nd AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, January 2004. American Institute of Aeronautics and Astronautics.

[17] David J. Kinney. Aerothermal anchoring of cbaero using high fidelity cfd. In *AIAA Conference*, 2007.

[18] Juyoung Lee, Mingyu Lee, Bong Jae Lee, and Ikjin Lee. A comprehensive multi-fidelity surrogate framework based on Gaussian process for datasets with heterogeneous responses. *Knowledge-Based Systems*, 295:111827, 2024.

[19] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier Neural Operator for Parametric Partial Differential Equations, May 2021. arXiv:2010.08895 [cs].

[20] Long Liang, John G Stevens, and John T Farrell. A dynamic adaptive chemistry scheme for reactive flow computations. *Proceedings of the Combustion Institute*, 32(1):527–534, 2009.

[21] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, March 2021.

[22] Lu Lu, Raphaël Pestourie, Steven G. Johnson, and Giuseppe Romano. Multifidelity deep neural operators for efficient learning of partial differential equations with application to fast inverse design of nanoscale heat transport. *Physical Review Research*, 4(2):023210, June 2022.

[23] Levi D. McClenny and Ulisses M. Braga-Neto. Self-adaptive physics-informed neural networks. *Journal of Computational Physics*, 474:111722, February 2023.

[24] Michael Penwarden, Shandian Zhe, Akil Narayan, and Robert M. Kirby. Multifidelity modeling for Physics-Informed Neural Networks (PINNs). *Journal of Computational Physics*, 451:110844, February 2022.

[25] P. Perdikaris, M. Raissi, A. Damianou, N. D. Lawrence, and G. E. Karniadakis. Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2198):20160751, February 2017.

[26] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, February 2019.

[27] Milad Ramezankhani, Amir Nazemi, Apurva Narayan, Heinz Voggenreiter, Mehrtash Harandi, Rudolf Seethaler, and Abbas S. Milani. A Data-driven Multi-fidelity Physics-informed Learning Framework for Smart Manufacturing: A Composites Processing Case Study. In *2022 IEEE 5th International Conference on Industrial Cyber-Physical Systems (ICPS)*, pages 01–07, Coventry, United Kingdom, May 2022. IEEE.

[28] Francesco Regazzoni, Stefano Pagani, Alessandro Cosenza, Alessandro Lombardi, and Alfio Quarteroni. A physics-informed multi-fidelity ap-

proach for the estimation of differential equations parameters in low-data or large-noise regimes. *Rendiconti Lincei, Matematica e Applicazioni*, 32(3):437–470, December 2021.

[29] Advisory Group For Aerospace Research and Development. Aerodynamic Problems of Hypersonic Vehicles. *AGARD Lecture Series*, 1(42), July 1972.

[30] Mehrnaz Rouhi Youssefi and Doyle Knight. Assessment of CFD Capability for Hypersonic Shock Wave Laminar Boundary Layer Interactions. *Aerospace*, 4(2):25, April 2017.

[31] Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6(4):911–917, July 1995.

[32] Juan Diego Toscano, Daniel T. Chen, Vivek Oommen, Jérôme Darbon, and George Em Karniadakis. A Variational Framework for Residual-Based Adaptivity in Neural PDE Solvers and Operator Learning, September 2025. arXiv:2509.14198 [cs].

[33] James G Wnek. *Hypersonic Conceptual Design Tool Comparison*. PhD thesis, Wright State University, 2022.

[34] Chen Xu, Ba Trung Cao, Yong Yuan, and Günther Meschke. A multi-fidelity deep operator network (DeepONet) for fusing simulation and monitoring data: Application to real-time settlement prediction during

tunnel construction. *Engineering Applications of Artificial Intelligence*, 133:108156, 2024.