Toward Improving Boussinesq Flow Simulations by Learning with Compressible Flow

Nurshat Mangnike
Vanderbilt University
Nashville, Tennessee, USA
nulixiati.mangnike@vanderbilt.edu

David Hyde
Vanderbilt University
Nashville, Tennessee, USA
david.hyde.1@vanderbilt.edu

ABSTRACT

In computational fluid dynamics, the Boussinesq approximation is a popular model for the numerical simulation of natural convection problems. Although using the Boussinesq approximation leads to significant performance gains over a full-fledged compressible flow simulation, the model is only plausible for scenarios where the temperature differences are relatively small, which limits its applicability. This paper bridges the gap between Boussinesq flow and compressible flow via deep learning: we introduce a computationally-efficient CNN-based framework that corrects Boussinesq flow simulations by learning from the full compressible model. Based on a modified U-Net architecture and incorporating a weighted physics penalty loss, our model is trained with and evaluated against a specific natural convection problem. Our results show that by correcting Boussinesq simulations using the trained network, we can enhance the accuracy of velocity, temperature, and pressure variables over the Boussinesq baseline—even for cases beyond the regime of validity of the Boussinesq approximation.

CCS CONCEPTS

• **Applied computing** → **Physics**; • **Computing methodologies** → **Modeling and simulation**; *Neural networks.*

KEYWORDS

Computational fluid dynamics, Boussinesq approximation, compressible flow, neural network, natural convection

ACM Reference Format:
Nurshat Mangnike and David Hyde. 2024. Toward Improving Boussinesq Flow Simulations by Learning with Compressible Flow. In *Platform for Advanced Scientific Computing Conference (PASC '24), June 3–5, 2024, Zurich, Switzerland*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3659914.3659919

1 INTRODUCTION

Fluid dynamics is a crucial scientific field for domains like aerospace, civil engineering, biomechanics, environmental science, and computer graphics [25]. In particular, modelers in these communities use computational fluid dynamics (CFD) to numerically simulate and analyze the behavior of fluids. Traditional CFD methods involve numerically solving governing partial differential equations (PDEs). However, CFD simulations are notorious for being computationally demanding, especially when using a general-purpose governing model like compressible flow.

To mitigate these computational costs, practitioners often use simpler models for the simulated fluid, such as the incompressible Navier-Stokes equations (plausible for nearly-incompressible fluids like water). Another approximating model is due to Boussinesq [6, 49]. The Boussinesq approximation is designed for buoyancy-driven flow problems, such as natural convection. This approximation is commonly understood to consist of the following assumptions [17]:

- Density is assumed constant, except when it directly causes buoyant forces;
- All other thermophysical fluid properties are constant;
- Viscous dissipation is assumed negligible;
- The temperature differences are small.

The first point means that compressibility is ignored in the continuity equation and that variations in density do not affect the flow field, except the buoyancy in the gravitational term. The model also assumes a linear change in density that is dependent on temperature.

Flows that can be accurately described by the Boussinesq approximation include those in the natural world (e.g., katabatic winds, atmospheric fronts, oceanic circulation) and those in industrial settings (e.g., ventilation in fume cupboards, dispersion) [4]. In particular, the Boussinesq approximation is the basis for a large number of recent works on thermal convection [2].

Gray and Giorgini [17] introduced conditions under which the Boussinesq approximation is appropriate to apply. They found valid ranges of temperature differences over reference length scales: these are $\Delta T/L_{\text{ref}} \leq 1.0 \times 10^3 \text{cm}/^\circ\text{C}$ and $\Delta T/L_{\text{ref}} \leq 9.9 \times 10^4 \text{cm}/^\circ\text{C}$ for air and water, respectively [17, 29]. Within the realm of applicability, Boussinesq flow simulations have been validated as closely agreeing with the results of corresponding compressible flow simulations; for example, Vierendeels et al. [44] and Mazumder [30] numerically simulated the square cavity benchmark problem of natural convection up to $Ra = 10^6$ and $\epsilon = 0.6$ and found tight agreement between Boussinesq and compressible flow results.

Nonetheless, in various scientific and industrial contexts, temperature differences and length scales extend beyond these limits where the Boussinesq approximation remains applicable [29] (e.g., see Figure 1). For example, when simulating foundry operations, thermal insulation in nuclear reactors, solar collectors, and astrophysical magnetohydrodynamics, the Boussinesq approximation provides results of limited accuracy [29]. We seek to overcome the limitations of the Boussinesq approximation—while preserving its substantial computational performance advantage—by introducing

PASC '24, June 3–5, 2024, Zurich, Switzerland

ACM ISBN 979-8-4007-0639-4/24/06.
https://doi.org/10.1145/3659914.3659919

deep learning techniques that leverage data from fully compressible natural convection simulations.

Applicability of Learning: Applying learning to CFD problems is not new [16]; however, many methods that have sought to combine physical simulation with learning sacrifice convergence or fidelity to the underlying physics [34, 41, 48]. This is at odds with the attitude of computational physics, which generally seeks methods that converge to a true solution [16, 21]. In turn, practitioners have questioned the robustness or reliability of learning-based approaches for real-world engineering problems with strict tolerances and possibly safety considerations [9, 43, 47]. Given these considerations, we argue that the present work is an attractive opportunity for applying learning: because Boussinesq flow is *already* an approximation of true physics (compressible flow equations), it is not harmful to introduce a learning method that brings results more closely in line with compressible flow, even if the method performs imperfectly. Users of Boussinesq flow simulations already accept a trade-off between physical fidelity and performance, so they should readily adopt the present method: one that still offers significant performance gains over compressible flow simulation while yielding results that more closely follow the laws of compressible flow.

Contributions: We introduce a CNN-based architecture, inspired by U-Net [36], that learns from compressible simulations of natural convection problems in order to correct Boussinesq-based simulations. Our specific contributions can be summarized as follows:

- We propose a neural network model that takes results from Boussinesq flow simulations as input and learns from the fully compressible simulation of natural convection problems. Our model can predict velocity, temperature, and pressure values that are more accurate than the pure Boussinesq approach, particularly in the case of large temperature variations in which the Boussinesq approximation is typically not appropriate.
- Inspired by physics-informed neural networks (PINNs) [34], we introduce a weighted mean-squared error (MSE) loss with a physics-informed penalty based on the ideal gas law. This penalty helps correlate the predicted temperature and pressure, compelling them to satisfy underlying physics constraints.
- We provide a novel alternative for solving natural convection problems that are beyond the valid range of the Boussinesq approximation without introducing a fully compressible scheme, avoiding the associated prohibitive computational costs. Our results demonstrate that our model is capable of inferring accurate flow approximation for square cavity benchmark problems where the classical Boussinesq approach fails.

Although our method requires us to run a number of simulations for training data, our aspiration is to create a model will have sufficient generalizability that any user of Boussinesq flow simulations can simply download our pre-trained model and only pay the small cost of inference; i.e., training is only required once, and we have performed it, so the cost of training becomes irrelevant for application of our technique. The present work does not fully realize this vision (see Section 7) but is a first step in this direction.

2 RELATED WORK

Hybridizing Learning and Simulation: In recent years, numerous works have sought to leverage machine and deep learning techniques to accelerate the traditionally computationally-intensive workflows of computational physics, see for example the reviews of Gibou et al. [16], Willard et al. [46], and Karniadakis et al. [22]. Generally speaking, such works seek to replace portions or substantially all of a simulation pipeline with evaluation of a neural network. For instance, Kaneda et al. [21] replace a portion of the conjugate gradients algorithm with a network inference in order to accelerate solving discrete Poisson equations. On a similar level, works like Um et al. [42] and Luna et al. [27] use networks to provide initial guesses for solvers used in simulation codes. Other works, such as Farimani et al. [13], use networks (in that case, a conditional generative adversarial network) to directly predict the solutions for steady state heat conduction and incompressible fluid flow. In general, various combinations of traditional simulation techniques and learning approaches are actively being considered by researchers.

Physics-Informed Neural Networks: A highly-cited technique for blending learning and simulation is the physics-informed neural network (PINN) [35] and its derivatives. The essential idea of PINNs is to encode the governing equations of a physical system as part of the loss function for training a network (moving all the terms of a PDE to the left-hand side, for example, and incorporating that left-hand side into a mean-squared loss). This amounts to making the governing physics a soft penalty for the network. A relevant application of PINNs in the literature is heat transfer problems [8], where the authors obtained relatively accurate temperature and velocity fields for forced and mixed convection with various (limited) thermal boundary condition information. However, soft penalties mean that governing equations are never exactly satisfied (up to numerical precision); furthermore, Krishnapriyan et al. [24] (among others) point out that soft constraints and the nature of PINNs themselves—regardless of the underlying network architecture—can lead to poor convergence, ill-conditioning, and other subtle problems. In turn, papers like Geng et al. [15], Zhu et al. [51], and Lu et al. [26] have all explored hard constraints that exactly enforce known physical laws or initial/boundary conditions, achieving these for example by incorporating sufficiently differentiable physics constraints as layers in the network itself.

U-Nets: Convolutional neural networks (CNNs) are well-known in per-pixel predictions in images, such as FlowNet [12]. Recently, CNNs have also been applied to computational physics problems, such as solving the Navier-Stokes equations [7] and predicting non-uniform steady laminar flow [18]. U-Nets [36] form a particular class of CNN architectures that have been shown to have benefits over traditional CNNs, e.g., for medical image segmentation [39], image denoising [20], and even coastal engineering [37]. In CFD, Sharma et al. [38] applied a U-Net to learn the steady-state solution of the 2D heat equation using a physics-informed kernel in a weakly-supervised framework. Hou et al. [19] argue for a U-Net-LSTM model over a traditional CNN-LSTM model in predicting hydrodynamics results for a submarine simulation. Thuerey et al. [40] selected U-Nets for simulating Reynolds-averaged Navier-Stokes

flows past airfoils. Inspired by these works, our model architecture (Section 4.1) combines certain features of CNNs and U-Nets.

3 BACKGROUND

Many fluid dynamics scenarios, including natural convection problems, are appropriately modeled by the multi-dimensional Euler equations for compressible flow,

$$\frac{\partial}{\partial t}\begin{pmatrix}\rho\\ \rho\mathbf{u}\\ E\end{pmatrix}+\begin{pmatrix}\nabla\cdot(\rho\mathbf{u})\\ \nabla\cdot(\rho\mathbf{u})\,\mathbf{u}\\ \nabla\cdot(E\mathbf{u})\end{pmatrix}+\begin{pmatrix}0\\ \nabla p\\ \nabla\cdot(p\mathbf{u})\end{pmatrix}=\vec{0}, \tag{1}$$

where t is time, ρ is density, $\mathbf{u}$ is velocity, E is total energy per unit volume, and p is pressure. This system of partial differential equations is closed via an equation of state, such as the ideal gas law, that relates pressure to density and internal energy. Various challenges arise when numerically solving these equations to simulate compressible flow. One such issue is preserving positivity of variables like pressure and internal energy that should, according to physics, remain non-negative [33, 50]. In practice, issues like these lead to restrictions like taking small time steps, using implicit time integration, and other numerical strategies that ultimately hinder the performance and/or scalability of codes.

3.1 Boussinesq approximation

The Boussinesq approximation is a commonly used simplification of the compressible Euler equations, specifically in the study of buoyancy-driven flows. The approximation assumes that the density variations in a fluid are relatively small, except for the effects of buoyancy. In other words, it assumes that the fluid is nearly incompressible, except for density changes due to temperature variations. The approximation is accurate when density and temperature variations are small, and it reduces the nonlinearity of the problem since changes in density are assumed negligible.

Following the treatment of Barletta et al. [3] and Martineau et al. [28], we briefly review the Boussinesq approximation. Under the Boussinesq approximation, the first row of Equation 1 reduces to the incompressible form $\nabla\cdot\mathbf{u}=0$ since the magnitude of first term in the row is small with respect to the velocity divergence $\nabla\cdot\mathbf{u}$.

For the second row, which describes conservation of momentum, we may write a slight modification of the Navier-Stokes equations:

$$\rho_0\left(\frac{\partial\mathbf{u}}{\partial t}+\mathbf{u}\cdot\nabla\mathbf{u}\right)=-\nabla p+\mu\nabla^2\mathbf{u}+\rho\mathbf{g}, \tag{2}$$

where μ is dynamic viscosity, ρ_0 is the initial (constant) density, and ρ is the current (temperature-dependent) density, which exists only in the body force term. We also introduce the thermal expansion coefficient,

$$\beta=-\frac{1}{\rho_0}\left(\frac{\partial\rho}{\partial T}\right)_{\mathrm{p}}, \tag{3}$$

where T is temperature, and the subscript of p denotes an assumption of holding the pressure constant during the expansion of the derivative (cf. Blandford and Thorne [5, Chapter 18] for justification). By approximating the rate of change of density with a Taylor series expansion,

$$\rho\approx\rho_0+\frac{\partial\rho}{\partial T}(T-T_0)+\frac{\partial^2\rho}{\partial T^2}\frac{(T-T_0)}{2}+\ldots \tag{4}$$

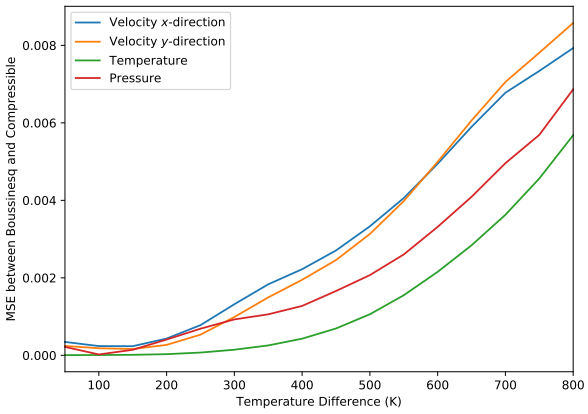


Figure 1: Mean-squared error (MSE) between Boussinesq and equivalent compressible flow simulations for a series of two-dimensional square cavity test problems (Section 5.1) with different initial temperature gradients across the domain. Differences are measured at the steady state. In the presence of larger temperature differences, the Boussinesq approximation breaks down, and the MSE between the Boussinesq and equivalent compressible flow simulations increases, illustrating the limitations of the Boussinesq approximation.

(where ρ_0 and T_0 are the reference density and temperature, respectively[1]), and ignoring the second-order and higher-order terms, we can use Equation 3 to express ρ as

$$\rho=\rho_0-\rho_0\beta(T-T_0). \tag{5}$$

For an ideal gas (we highlight this assumption), we take the thermal expansion coefficient as $\beta=\frac{1}{T_0}$[2]. This yields

$$\rho_0\left(\frac{\partial\mathbf{u}}{\partial t}+\mathbf{u}\cdot\nabla\mathbf{u}\right)=-\nabla P+\mu\nabla^2\mathbf{u}-\rho_0\frac{T-T_0}{T_0}\mathbf{g}, \tag{6}$$

where $P=p+\rho_0\mathbf{g}\cdot\mathbf{z}$ is a shifted pressure (with $\mathbf{z}$ such that $\mathbf{g}\cdot\mathbf{z}=||\mathbf{g}||$: the standard basis vector for the y-axis in 2D, and for the z-axis in 3D), which is helpful in avoiding numerical roundoff error in the evaluation of the body force (buoyancy) term when ρ is close to ρ_0.

[1]The reference density ρ_0 is chosen as the density of the material at the reference temperature T_0. The choice of reference temperature should be made wisely: it is clear that since the Taylor expansion of ρ used in the approximation is taken around the point T_0, T_0 should be close to temperatures T found in a simulation. One choice for T_0 is the anticipated mean temperature of a simulation. Although a choice of T_0 can affect the baseline level of accuracy of a Boussinesq flow simulation, Figure 1 demonstrates that with enough variation in temperature across a domain, even a good choice of T_0 cannot preserve the accuracy of the Boussinesq approximation, which desires $\rho\approx\rho_0 \iff |T-T_0|\ll 1$.

[2]Under an isobaric assumption (we again point to Blandford and Thorne [5, Chapter 18]), the ideal gas law (cf. Section 3.2) can be reduced to Charles' Law, which, for a unit volume of ideal gas, reads $\beta=1/T$. Clearly, T is not constant. However, in Boussinesq's original work [6, page 182], he appears to suggest a constant value of $T=273$K, as he writes, "[...] the coefficient of cubic (volumetric) expansion $\frac{1}{273}=0.00366$ common to all gases" (authors' translation). To reconcile this, observe that Equation 4 is a Taylor expansion of $\rho(T)$ *about* T_0; so the first-order derivative in the equation is to be evaluated at T_0, which implies that the derivative must be evaluated at the same point in Equation 3 to perform the substitution for Equation 5. Hence we arrive at $\beta=1/T_0$. We emphasize that, interestingly, this constant value seems to still be different, in general, than what appears to be Boussinesq's original suggestion of $1/273$.

Given the incompressibility condition and the above momentum equation, the Boussinesq approximation is then closed with an equation for temperature evolution, namely, the heat equation. The full model is thus given by:

$$\nabla \cdot \mathbf{u} = 0 \tag{7}$$

$$\rho_0 \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) = -\nabla P + \mu \nabla^2 \mathbf{u} - \rho_0 \frac{T - T_0}{T_0} \mathbf{g} \tag{8}$$

$$\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T = \alpha \nabla^2 T, \tag{9}$$

where α is thermal diffusivity. In this approximation, viscous heat dissipation and pressure work terms are assumed to be negligible. This is only valid when temperature and density variations are small, which introduces errors [17] of the order of 1% if the temperature differences are below 15°C (and of course, this error grows large when temperature differences are greater—see Figure 1).

3.2 Ideal gas law and reference parameters

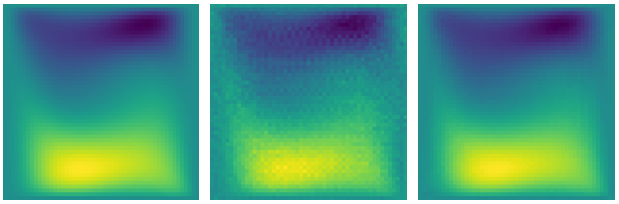
The ideal gas law describes the relationship between temperature, pressure, and density of a hypothetical ideal gas, and is often expressed as

$$pV = nRT, \tag{10}$$

where p, V and T are the pressure, volume, and temperature of the gas, respectively, n is the number of moles of the gas, and R is the ideal gas constant. By introducing the density as $\rho = m/V$ and $R_{\text{specific}} = R/M$, where M is total mass and m is molar mass, and replacing n with M/m, we obtain

$$p = \rho R_{\text{specific}} T, \tag{11}$$

where R_{specific} represents the specific gas constant for the gas being modeled. All of our numerical experiments use material parameters, such as ρ_0, μ, and α, for dry air. We also note that we use a constant reference temperature of $T_0 = 293.15$K throughout our numerical experiments; while an earlier footnote suggests that one may wish to choose a unique T_0 per simulation to potentially reduce error, our datasets involve such wide ranges of T within a given simulation that a suboptimal choice of T_0 is not the dominant source of error (and regardless, our learning model is demonstrated to overcome these potential sources of error).



(a) Target (b) U-Net (c) Our model

Figure 2: Comparison between standard U-Net and our model. The leftmost subfigure shows a target field (x-velocity) for a 2D simulation. Our model predicts much smoother results than a a standard U-Net model of similar complexity.

4 LEARNING MODEL

4.1 Model architecture

For two-dimensional problems, our network's input and output consist of four channels with the same dimensions: velocity (in x and y directions), temperature, and pressure. We note that, for two-dimensional simulations, each of these channels can be viewed as a single-channel 2D image. The general training procedure (cf. Section 5) involves input data from Boussinesq flow simulations, while target simulations (for measuring a loss) use a full compressible flow solver.

Our learning model is inspired by the U-Net [36] architecture. We initially applied a traditional CNN architecture, which only has convolutions and deconvolutions, since the model input and output should have the same resolution. However, we found that an architecture based on U-Net could produce results with lower memory requirements. Yet in our experiments, we found that despite the memory savings, a vanilla U-Net architecture produced "noisy" results, where predicted fluid fields like velocity did not exhibit expected smoothness—see Figure 2. Intuitively, we believe this is related to the fact that the data passing through the network loses certain areas of the signal when down-sampling and is thus inhibited from matching smoothly; though a rigorous analysis of these architectural considerations is left for future work.

To address this limitation, our architecture includes two down-sampling and up-sampling processes, which matches the original U-Net paper [36], but it also includes convolution-deconvolution layers (symmetric, 8 layers) that connect the two paths at each level of different resolution (see Figure 3). Based on our generated data (Section 5.1), input image size is 52×52 for each channel, and in the first convolution layer, the number of feature channels is increased to 64 after a 2D convolution, where the kernel size is 3, the stride is 1, and reflection padding with 1. The convolution is followed by a LeakyRelu activation with negative slope parameter of 0.7, and then we apply batch normalization. The network layer connections and the number of feature channels in each layer are illustrated in Figure 3. The deepest layer has up to 2048 feature channels. We used a 2×2 max pooling operation with stride 2 for down-sampling and a 2×2 transposed convolution (deconvolution) followed by concatenation with the correspondingly cropped feature map from the convolution-deconvolution part. In total, our network consists of 41 convolutional layers. Figure 2 gives a glimpse into how our modified architecture compares to a vanilla U-Net implementation; detailed results using our model are given in Section 6.

4.2 Loss function

We train our network using a weighted mean-squared error (MSE) loss function $\mathcal{L}$. For two-dimensional test problems, the overall loss we measure is a linear combination of five distinct MSE losses: four of the losses correspond to standard MSE losses for the four physical variables we evaluate in the network, and one loss corresponds to a soft physics penalty based on the ideal gas law. Our overall loss function can be expressed as the following:

$$\mathcal{L} = \beta_0 \mathcal{L}_{\text{Vx}} + \beta_1 \mathcal{L}_{\text{Vy}} + \beta_2 \mathcal{L}_{\text{T}} + \beta_3 \mathcal{L}_{\text{P}} + \beta_4 \mathcal{L}_{\text{gas}}, \tag{12}$$

where $\mathcal{L}_{\text{Vx}}$, $\mathcal{L}_{\text{Vy}}$, $\mathcal{L}_{\text{T}}$, and $\mathcal{L}_{\text{P}}$ are the MSE losses for velocity in the x and y directions, temperature, and pressure, and where $\mathcal{L}_{\text{gas}}$

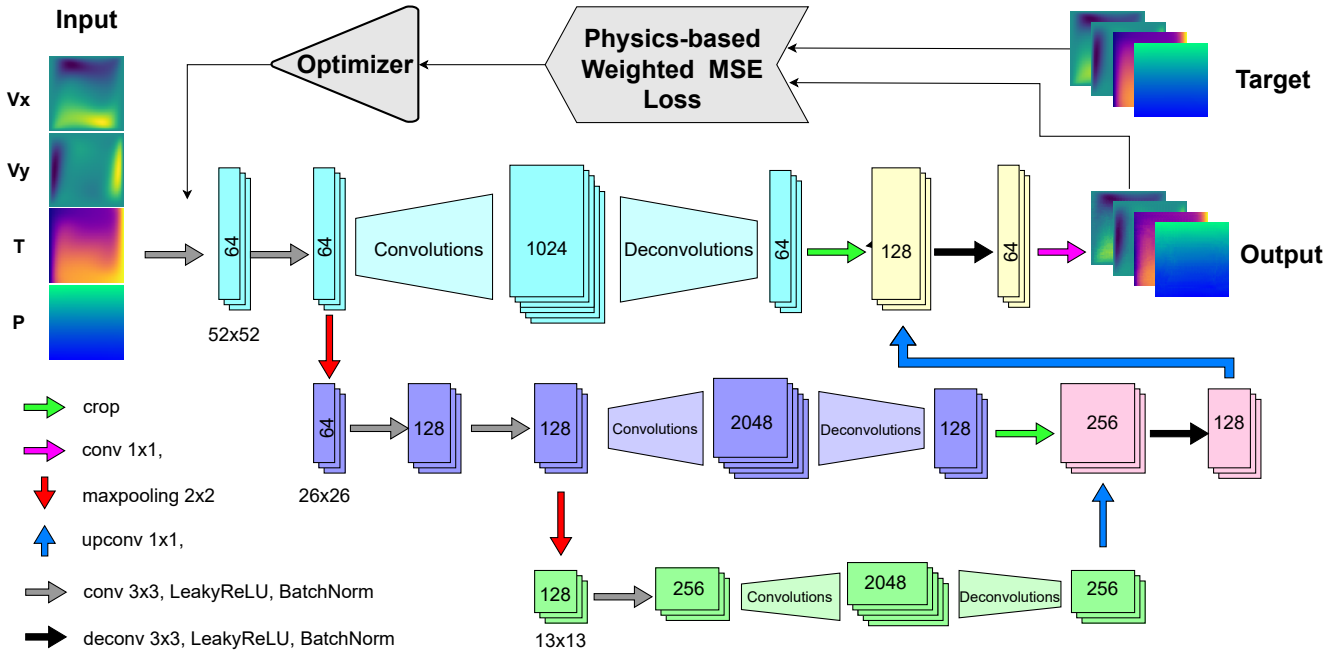


Figure 3: Model architecture. Each stacked box represents a multi-channel feature map (the number of channels is denoted on the box), and each colored arrow denotes different operations, which are illustrated in the bottom left of the figure. The input, output, and target consist of four different physical variables, each represented as a single-channel 2D image.

is our ideal gas law constraint, respectively. The β_i represent the corresponding weights.

The physics-informed loss $\mathcal{L}_{\text{gas}}$ is based on the ideal gas law. We assume the predicted density is the same as in the target compressible fluid simulation since we are not including the density in the neural network. Then we can write the ideal gas law in the form of Gay-Lussac's law, which states the pressure of a gas varies directly with the temperature:

$$\frac{P_{\text{pred}}}{T_{\text{pred}}} = \frac{P_{\text{target}}}{T_{\text{target}}}, \tag{13}$$

where P_{pred}, T_{pred} are predicted pressure and temperature and P_{target}, T_{target} are the target pressure and temperature. Then the physics constraint that we want to minimize can be written as

$$\left| P_{\text{pred}} T_{\text{target}} - P_{\text{target}} T_{\text{pred}} \right|, \tag{14}$$

and our physics-informed loss $\mathcal{L}_{\text{gas}}$ can be expressed as

$$\mathcal{L}_{\text{gas}} = \frac{1}{N} \sum_{i=0}^{N} \left\| P_{\text{pred}}^{(i)} T_{\text{target}}^{(i)} - P_{\text{target}}^{(i)} T_{\text{pred}}^{(i)} \right\|^2, \tag{15}$$

where N is the total number of "pixels" in the 2D "image," and i is the index of a particular pixel. We note that even though we are applying a soft penalty rather than enforcing a hard physics constraint, this simple addition to our network is enough to help improve Boussinesq flow simulations; and, per the argument in Section 1, this is acceptable given that Boussinesq flow outputs are already not expected to perfectly match compressible flow results.

5 TRAINING AND EXPERIMENT SETUP

5.1 Data generation

We generated training and test data using the proprietary COMSOL multiphysics software, which provides out-of-the-box support for performing Boussinesq and compressible flow simulations. All of our generated simulation data uses the same computational grid and resolution. We create pairs of Boussinesq and compressible flow simulations in order to measure the loss due to the Boussinesq approximation for each simulation. Each pair uses identical initial and boundary conditions for the simulation; the only difference is the flow model used to evolve the simulation.

We remark that our simulations in COMSOL temporally discretize the continuous governing equations (whether the compressible Euler equations or the Boussinesq approximation) using an implicit BDF method [10, 14], with all default settings [1]. Spatially, we use COMSOL's default finite element discretization on a uniform Cartesian grid (i.e., a uniform square/cubic mesh), which involves linear/first-order shape functions throughout. Of course, these discretization choices can have an impact on our results; the entire field of computational physics emerged because different discretization schemes have different mathematical, numerical, and computational properties. We will simply remark that our high-level approach of correcting Boussinesq flow data using compressible flow data and a neural network is applicable regardless of the chosen discretization, and even if a user runs a Boussinesq flow simulation using a different discretization, our pre-trained model would still be applicable assuming the computational mesh is identical (though any minor

discretization artifacts could make the network perform slightly worse than if it were trained using the user's chosen discretization schemes).

The span of possible compressible flow scenarios to simulate is vast. For the present work, we focused on training and testing our model against a type of problem where—within certain ranges—the Boussinesq approximation is known to be appropriate. Specifically, we consider the differentially heated square cavity problem proposed by de Vahl Davis [11] under various initial and boundary conditions (see Figure 4). This problem is a common benchmark for validating numerical simulations. Although all our data is based on this type of square cavity setup, we emphasize that different initial and boundary conditions for this class of problem can yield significantly different results. Moreover, sufficiently increasing the initial temperature difference across the domain makes the Boussinesq approximation noticeably inaccurate, which means this problem setup is sufficiently challenging to show meaningful improvements over the baseline Boussinesq flow results when incorporating our neural network. In the future, though, we would like to extend the present idea to cover a vast array of "random" compressible flow scenarios, so that the present technique could generalize better to any practical CFD simulation.

The setup for the square cavity problem is shown in Figure 4. The square domain is L in width and height, and a 50×50 uniform Cartesian grid is used for spatial discretization. The natural convection problem is described by two vertical heated walls with initial temperatures: a hot temperature T_{h}, and a cold temperature T_{c}. The upper and lower boundaries are insulated, defined by zero heat flux in the y-direction, $\vec{q} \cdot \vec{n} = \vec{0}$ (where $\vec{n}$ is the unit normal to the boundary). All four walls have no-slip, zero-mass flux boundary conditions. The temperature differential produces density variation that drives the buoyant flow. The temperature differential between the two side walls is randomized for each simulation in our experiments. In order to address the null space introduced by having all-Neumann boundary conditions, we fix the P at the upper left corner of the domain with $P_0 = 0$ (for consistency, this pressure constraint is used for both Boussinesq and compressible simulations). Initial velocity is set to zero throughout the domain.

We run each simulation for a total of 50s, using a constant time step of 0.05s. We extract data from each simulation every 20 time steps, which yields a total of 50 data points from each simulation. After extracting the simulation data, which includes x-velocity, y-velocity, temperature, and pressure, we represent these four flow variables as 2D images, where each pixel value represents the variable value at the center of the corresponding grid cell. For training data, we generated 4×40,000 "images," where the initial temperature difference between the two side walls is randomized between 0K and 800K. The Boussinesq-based simulation is treated as input data, and the compressible flow simulation is used as target data (ground truth). For the validation set, we generated $4 \times 10{,}000$ data using a wider temperature difference range, from 0K to 1200K, which can prevent the network from overfitting. Finally, we generated two distinct test sets, each with 5,000 (Boussinesq, compressible) pairs of data. We used two different temperature difference ranges for these test sets. In the first test set, initial temperature differences vary from 0K to 800K (the same range as the training set). In the second test set, the initial temperature difference is chosen from 800K to 1400K, i.e., entirely outside the range of the training set. This second data set serves two purposes: it allows us to evaluate the performance of our approach in a regime *far* outside where the Boussinesq approximation is applicable, and it also demonstrates how well our network generalizes to unseen (and inherently more challenging) problems.

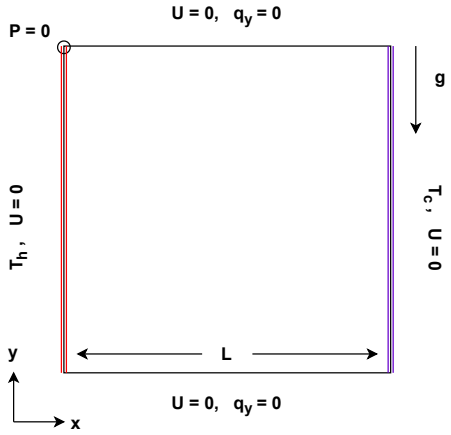


Figure 4: Domain geometry and boundary conditions for the simulations used as training and test data. A temperature gradient is set across the domain with a hot left wall (temperature $T_{\mathbf{h}}$) and a cooler right wall (temperature $T_{\mathbf{c}}$).

5.2 Network training

Our network architecture is implemented using PyTorch [32], and the training process is performed on a single NVIDIA RTX A6000 GPU. We normalize the data from each of the four channels to be in $[0, 1]$ since the four physical variables can have drastically different scales (this affects our choice of weights β_i). We note that when computing the physics-informed loss term (Equation 15), we rescale the pressure and temperature values to their original ranges, so that the expression in the equation is physically meaningful. We adjust the associated loss coefficient β_4 accordingly to compensate for scale disparities with the other loss terms (to give a sense, we find that $\mathcal{L}_{\text{gas}}$ initially can be ten orders of magnitude larger than the other loss terms).

The batch size for training and validation is chosen as 235. We use our separately generated training and validation sets, and all data is normalized to zero to one by the fixed min/max values based on prior knowledge, where the x-velocity is normalized by fixed min/max value $[-1.6, 1.6]$, x-velocity is $[-2.6, 2.6]$, temperature is $[200, 1600]$ and pressure is $[-0.55, 0.11]$[3]. For our loss function $\mathcal{L}$ in Equation 12, we selected the weights as $\beta_0 = 0.25$, $\beta_1 = 0.25$, $\beta_2 = 0.35$, $\beta_3 = 0.149999$, and $\beta_4 = 1{\times}10^{-6}$, by hand-tuning (in the future, a grid search or other technique could be used to optimize these weights). When we calculate the physics-informed loss $\mathcal{L}_{\text{gas}}$, we always rescale the normalized pressure and temperature values back

[3]We note that COMSOL uses relative pressures, so the pressure values observed during computation are relative to 1atm; we do not observe negative absolute pressures.

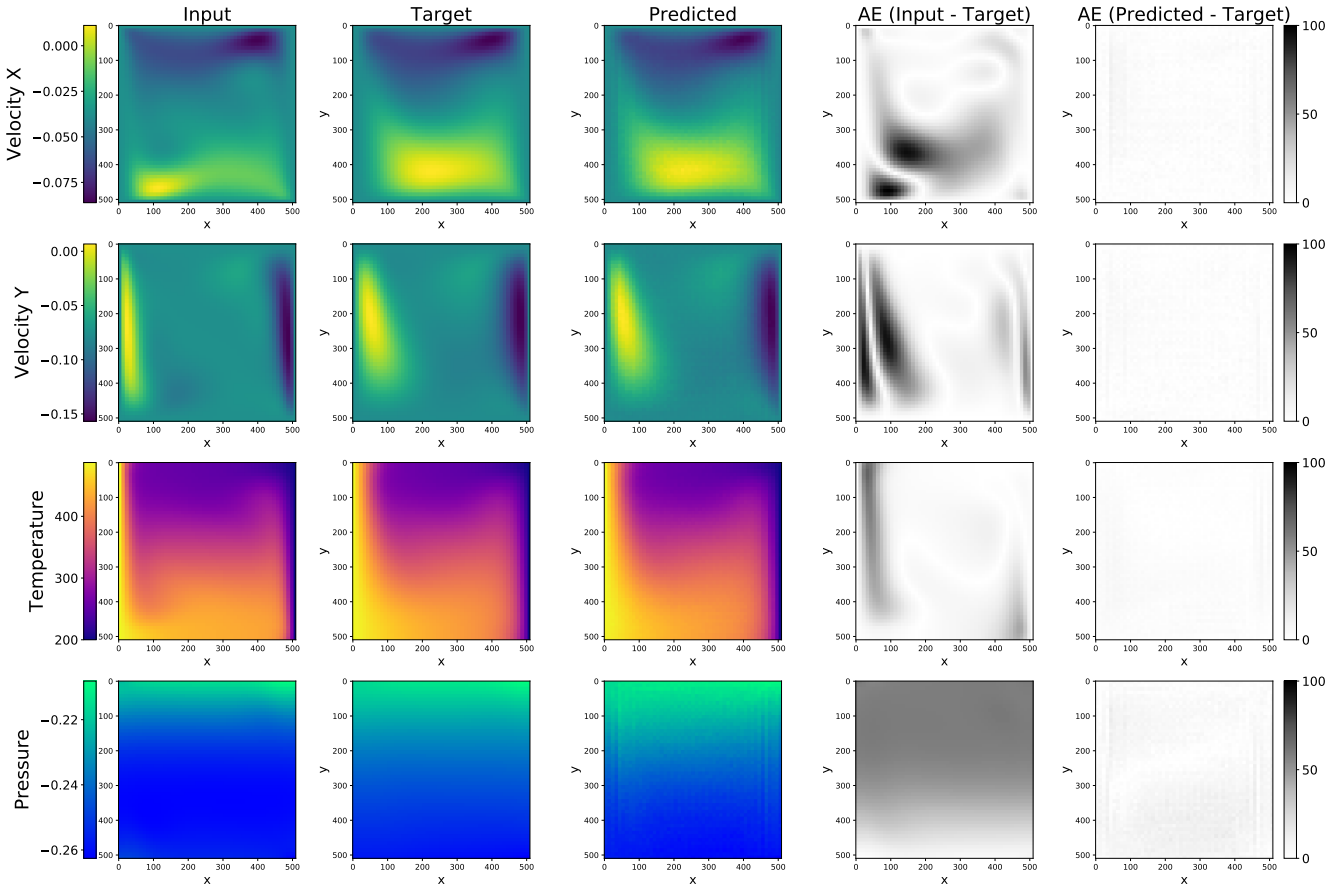


Figure 5: A representative result from our first 2D test set. The four rows display x-direction velocity, y-direction velocity, temperature, and pressure. The first three columns visualize the input, target, and predicted data, respectively, while columns 4–5 denote the absolute value of absolute error (scaled by 500) between input-target and predicted-target values.

to the original physical values to ensure it is physically consistent. We use PyTorch's Adam [23] implementation as the optimization algorithm for our training loop, with a constant learning rate of 2×10^{-5} (all other parameters are default). We use 70 epochs for training. In total, training took approximately 12 hours.

6 RESULTS AND EVALUATIONS

We emphasize that the overall computing workflow of our proposed method involves several steps. First, training and test data is generated using COMSOL (which could easily be substituted for any other CFD package). Next, we train our network (cf. Section 5.2). To leverage the trained model, we run a new Boussinesq simulation in COMSOL, and after the final time step, we correct the state of the simulation by evaluating the network and using the resulting fluid field predictions. Our test system for the results in this paper is a single workstation with two AMD EPYC 75F3 CPUs, for a total of 128 threads, all running with stock settings. The workstation has 512GB RAM, an NVIDIA RTX A6000, and runs Debian 12. We note that we mainly benefit from these system specs (particularly, the GPU) during training, and that even an average laptop is capable of running the experiments in the following subsections, given the pre-trained model.

6.1 Validity of Boussinesq approximation

We first demonstrate that the Boussinesq approximation becomes inaccurate as the initial conditions of the square cavity problem become more extreme. We generated several steady-state simulations with various temperature differences (T_h and T_c) using the square cavity benchmark problem. We analyzed the difference between the four variables from the two simulations, taking the compressible flow simulation results as ground truth. Figure 1 shows the results. The figure demonstrates that for all four variables—particularly for velocity—larger temperature differences lead to superlinearly greater errors in the Boussinesq flow simulations. Hence, there is a substantial gap for our learning-based model to close.

6.2 Learning-based results

We trained and validated our model using numerical simulation data from the square cavity benchmark problem (see Section 5.1). We evalute our model against two distinct test data sets. Our first test

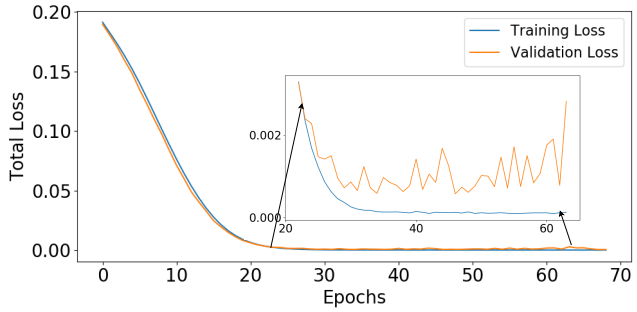


Figure 6: Training and validation loss. The inset shows both losses from epoch 22 to 65. We ultimately use the model from epoch 40 for our 2D tests.

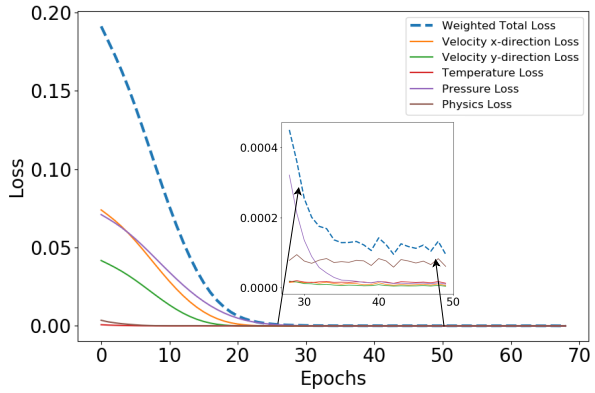


Figure 7: Weighted total loss and weighted individual losses. The inset shows these total and individual losses from epoch 25 to 50.

set for evaluation, of size 5,000, includes simulations with random initial temperature differences from 0K to 800K (the same range as the training data). By Figure 1, these simulations are sufficiently challenging tests that a basic Boussinesq flow simulation incurs substantial error. We also evaluate our model against a second test data set (also of size 5,000). Simulations in this second data set have initial temperature differences that are randomly chosen from 800K to 1400K; they are entirely outside the range of data seen during training, and are far outside the regime where the Boussinesq approximation is valid.

Figure 5 shows a representative result from the first test set. We apply our network to the output data from the associated Boussinesq flow simulation *once*, at time $t = 50$s. (For visualizing an entire simulation, this network correction could be applied independently to each time step.) The figure breaks down the performance of our method on the four variables included in our network: x- and y-velocities, temperature, and pressure. The first three columns in Figure 5 visualize the corresponding variable in the 2D grid for input data (Boussinesq approach), target data (compressible approach), and predicted data (Boussinesq flow data corrected by one application of our network model). The absolute values of absolute errors between input, inferred, and target (ground truth) values are demonstrated as a 2D grayscale heat map in the fourth and fifth columns, where the black regions imply the higher errors. In these last two columns, all values are scaled by 500 for ease of visibility, and resulting scaled values outside the range $[0, 100]$ are clamped.

Figure 5 clearly shows that the absolute error of our method (correcting a Boussinesq flow with our neural network) can be substantially lower than when using a Boussinesq flow model alone. Despite the significant success in reducing error, we observe that some subtle vertical patterns are evident in the predicted pressure field, i.e., the predicted field is not as smooth as the ground truth. We believe that additional convolutional layers may assist here, and that these effects may be less noticeable with higher-resolution results; we aim to perform higher-resolution studies in future work.

Quantitative results across both the first test set as well as the more challenging second test set are provided in Table 1, which reports the mean MSE in each of the four physical variables across each test dataset. For the first test set, MSE is reduced about three orders of magnitude for each variable. In the second test set, where Boussinesq and compressible flow simulations are more divergent and where our network has not previously seen data, our model still performs respectably, reducing MSE by around 5–10x. This suggests some degree of successful generalization of our model. We remark that when assessing our training set, we noticed that the velocities and temperatures included more initial error than the pressure variable (a trend we also observed in Figure 1). This is the main reason we set the pressure loss weight β_3 to be relatively smaller in Equation 12.

Critically, Table 1 also reports the performance of the three methods considered: a pure Boussinesq flow simulation, a Boussinesq simulation corrected with our network, and a full compressible flow simulation. The Boussinesq approximation enables a 44% speedup over the full compressible flow model; however, as seen in the table, these results incur a significant loss of accuracy (e.g., in the first decimal place, for pressure). In contrast, the results of our method are closely in line with the compressible flow results, yet involve only a small overhead to use. This highlights the key advantage of our method: significantly improving accuracy while maintaining a 42% speedup over compressible flow.

Assessing our neural network model, Figure 6 shows the total training and validation loss over the course of training. These losses drop dramatically in the first few epochs; we ultimately used a model from epoch 40 for our 2D tests. In the inset in the figure, we observe that the validation loss starts fluctuating around the value of the training loss starting from epoch 25, and does not seem to materially decrease after around epoch 40. In Figure 7, we demonstrate each loss separately without weights, where we also find the pressure loss converges more slowly.

We also considered the Structural Similarity Index (SSIM) as an evaluation metric. SSIM was first introduced by Wang et al. [45] for comparing the structural similarity of two images. The SSIM score ranges from 0 to 1, with an SSIM score of 1 meaning the two images are identical, while a score of zero implies they are completely uncorrelated. SSIM scores across variables in our test set are also reported in Table 1. We see that, as hoped, SSIM scores between the Predicted and Target images are higher than those between the Input and Target images.

Table 1: MSE and SSIM scores for our two-dimensional test data sets (each of size 5,000). In Test Set 1, the temperature differential ranges from 0K to 800K (within the range of training data), while in Test Set 2, it ranges from 800K to 1400K (outside the training data range). Relative errors (with respect to the compressible flow results, taken as ground truth) are reported for both the Boussinesq simulations and our network-corrected Boussinesq approach. All errors are averaged over the respective data set.

Models	Flow Variables								Time (s)
	Velocity x-direction		Velocity y-direction		Temperature		Pressure		
	MSE	SSIM	MSE	SSIM	MSE	SSIM	MSE	SSIM	
Compressible	-	-	-	-	-	-	-	-	105.84
Test Set 1									
Boussinesq	1.93×10^{-3}	0.836	1.91×10^{-3}	0.835	8.34×10^{-4}	0.914	6.19×10^{-3}	0.857	73.63
Bous. + Model	2.77×10^{-5}	0.984	1.82×10^{-5}	0.987	9.39×10^{-6}	0.954	8.42×10^{-5}	0.952	73.63 + 0.93
Bous. (Rel)	5.02×10^{-2}	-	4.62×10^{-2}	-	8.04×10^{-2}	-	9.58×10^{-2}	-	-
Bous. + Model (Rel)	7.27×10^{-3}	-	6.38×10^{-3}	-	4.97×10^{-2}	-	1.06×10^{-2}	-	-
Test Set 2									
Boussinesq	8.62×10^{-3}	0.632	9.84×10^{-3}	0.588	1.33×10^{-2}	0.768	8.68×10^{-2}	0.295	-
Bous. + Model	1.76×10^{-3}	0.772	1.41×10^{-3}	0.786	1.50×10^{-3}	0.925	1.63×10^{-2}	0.487	-
Bous. (Rel)	1.22×10^{-1}	-	1.28×10^{-1}	-	1.69×10^{-1}	-	5.10×10^{-1}	-	-
Bous. + Model (Rel)	5.40×10^{-2}	-	4.96×10^{-2}	-	6.19×10^{-2}	-	1.72×10^{-1}	-	-

Table 2: MSE and SSIM scores evaluated across our three-dimensional test sets (each of size 5,000). As in 2D, our method reduces MSE and increases SSIM for all variables, across both test sets, except for y-velocity in the challenging Test Set 2. We also report relative errors (with respect to the compressible flow results, taken as ground truth) for the Boussinesq simulations as well as our network-corrected Boussinesq approach. All errors are averaged over the respective data set.

Models	Flow Variables										Time (s)
	Velocity x-direction		Velocity y-direction		Velocity z-direction		Temperature		Pressure		
	MSE	SSIM	MSE	SSIM	MSE	SSIM	MSE	SSIM	MSE	SSIM	
Compressible	-	-	-	-	-	-	-	-	-	-	305.74
Test Set 1											
Boussinesq	1.70×10^{-3}	0.958	9.94×10^{-4}	0.945	1.32×10^{-3}	0.944	9.23×10^{-4}	0.967	4.20×10^{-3}	0.969	192.16
Bous. + Model	6.68×10^{-5}	0.996	7.55×10^{-5}	0.994	4.14×10^{-5}	0.996	4.04×10^{-5}	0.990	6.81×10^{-5}	0.995	192.16 + 1.06
Bous. (Rel)	4.51×10^{-2}	-	3.27×10^{-2}	-	3.47×10^{-2}	-	7.92×10^{-2}	-	8.72×10^{-2}	-	-
Bous. + Model (Rel)	1.21×10^{-3}	-	1.34×10^{-2}	-	9.96×10^{-3}	-	7.21×10^{-2}	-	1.11×10^{-2}	-	-
Test Set 2											
Boussinesq	6.75×10^{-3}	0.890	3.27×10^{-3}	0.868	6.13×10^{-3}	0.822	1.35×10^{-2}	0.869	5.56×10^{-2}	0.579	-
Bous. + Model	3.90×10^{-3}	0.906	4.29×10^{-3}	0.846	2.38×10^{-3}	0.883	2.85×10^{-3}	0.929	3.19×10^{-2}	0.683	-
Bous. (Rel)	1.01×10^{-1}	-	6.65×10^{-2}	-	8.95×10^{-2}	-	1.71×10^{-1}	-	4.45×10^{-1}	-	-
Bous. + Model (Rel)	7.86×10^{-2}	-	8.83×10^{-2}	-	6.36×10^{-2}	-	7.93×10^{-2}	-	2.96×10^{-1}	-	-

6.3 Three-dimensional results

To highlight the practicality of our method, we further demonstrate our technique on three-dimensional simulation problems. In 3D, we extend the differentially heated square cavity problem domain to a "cubic cavity," where the two x-direction side walls are initialized with high and low temperatures. As in the 2D case, there is no heat or mass flux at the boundaries. The computational grid resolution is $20 \times 20 \times 20$. We set the zero-pressure constraint point along the colder wall at the front-top corner (with respect to Figure 8). Other aspects of the initialization are analogous to the 2D problem. For our 3D data, we also sample the simulations every 20 time steps and run each simulation until a final time of 50s, yielding 50 data points per simulation.

To apply our technique in 3D, we adjusted our network architecture. Input and output channels are increased from 4 to 5 since we include z-direction velocity. For efficiency purposes, we decreased the number of feature maps and increased the downsampling and upsampling path from two to three; 3D convolution and deconvolutions are used, and other parameters are unchanged. We also

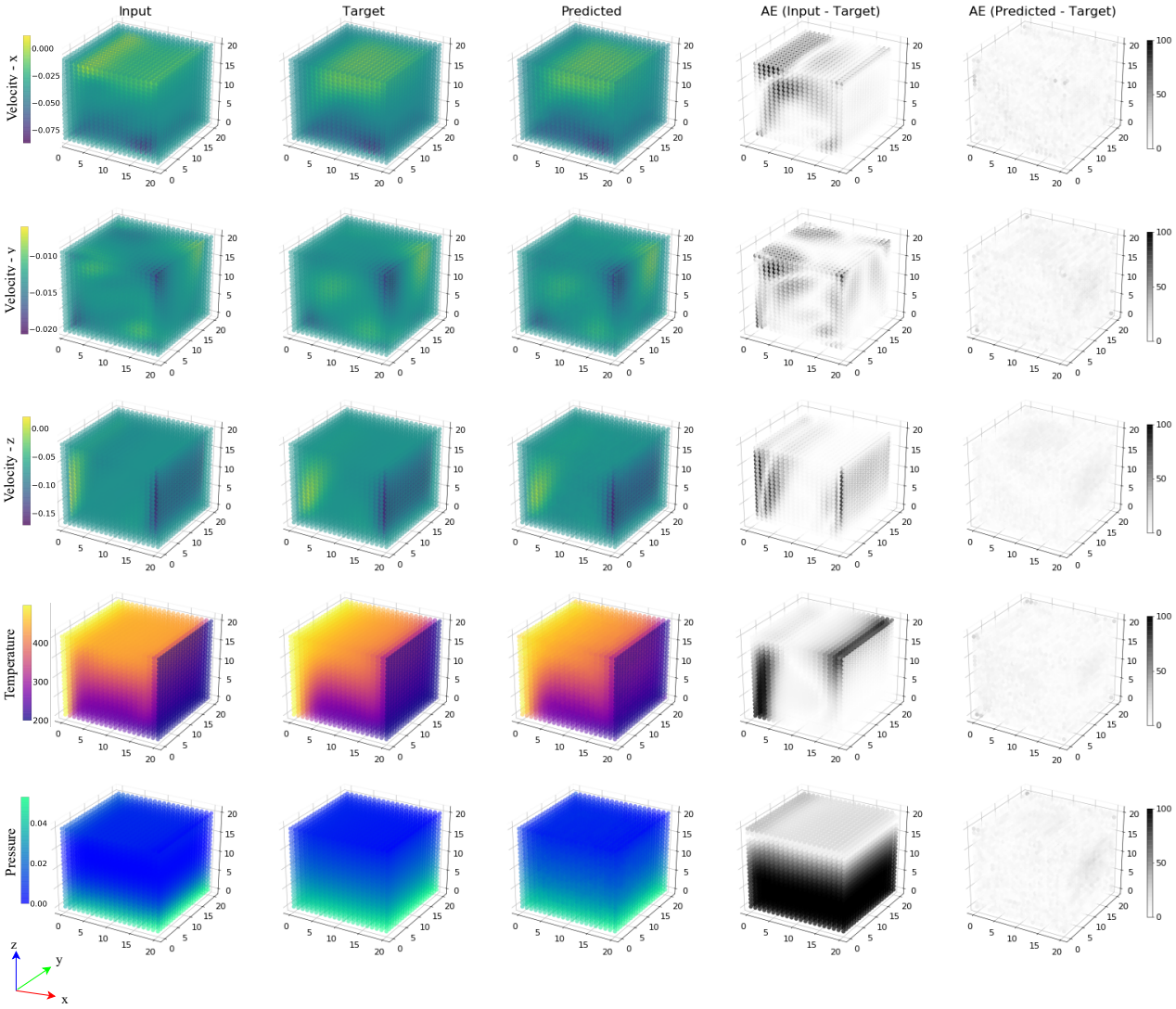


Figure 8: A representative result from the first 3D test set. The five rows display x-direction velocity, y-direction velocity, z-direction velocity, temperature, and pressure. The first three columns visualize the input, target, and predicted data, respectively, while columns 4-5 denote the absolute value of the absolute error (scaled by 1000 for ease of visibility) of input-target and predicted-target values.

added another loss and weight for the z-direction velocity. The loss function weights we used for 3D training are $\beta_0 = 0.15$, $\beta_1 = 0.18$, $\beta_2 = 0.18$, $\beta_3 = 0.3$, $\beta_4 = 0.18985$, and $\beta_5 = 5 \times 10^{-5}$, which correspond to x, y, z velocity, temperature, pressure, and physics-informed loss, respectively. We trained this modified network using a training and validation set of size ($5 \times 10{,}000$) and ($4 \times 2{,}500$), and temperature ranges are similar to 2D for both sets.

Table 2 shows our results for two test tests, which are 3D analogues of the two test sets in 2D (same sizes and temperature ranges). Figure 8 shows a representative result from the first test set, which involves data in the range of the training data set. We computed SSIM scores in 3D using the implementation by Park [31]. From the table, we observe that MSE is reduced and SSIM is increased significantly for all variables in both test sets, except for y-velocity in the more challenging Test Set 2. With more degrees of freedom, the 3D tests are more challenging than those in 2D, and we believe it may require further parameter tuning and perhaps a deeper network in 3D in order to achieve better generalization. Still, we emphasize that *in 2D and 3D, we only evaluated against our test sets once,* with no attempts to re-adjust our weights or make other modifications (we believe this is in the true spirit of machine learning), and we were pleased with the relatively successful generalization we observed. As a preliminary demonstration of feasibility, these results show

that our general approach has potential in 3D for improving the results of Boussinesq flow simulations. A 59% speedup is observed for the Boussinesq flow simulation over compressible flow, on average, while the speedup is 58% when the learning model is applied.

7 CONCLUSIONS AND FUTURE WORK

We proposed a novel application of deep learning techniques to the Boussinesq approximation of compressible flow. Our physics-informed weighted loss function is used to train a network that helps correct Boussinesq flow simulation output to more closely match simulation results using a full compressible model. Evaluating a class of 2D differentially heated square cavity test problems demonstrated that our model is capable of accurately predicting velocity, temperature, and pressure in regimes ($T_h - T_c$) that are traditionally beyond the applicable region of the Boussinesq approximation. Thus, our approach is advantageous for simulating natural convection problems that would normally be simulated using the Boussinesq approximation, or for those where use of a computationally-costly compressible flow model would be required—especially since the marginal cost of performing inference using our network is negligible compared to overall simulation time. A preliminary three-dimensional feasibility study was also shown; and for both 2D and 3D results, we observed reasonable success in generalizing to data outside the training range, as well as on data that is far outside the regime where the Boussinesq approximation is applicable. We remark that our general motivation of correcting an approximation of compressible flow via deep learning is not just limited to the Boussinesq approximation, and could be applied to other compressible flow approximations as well.

Several limitations of the present work offer opportunities for future research. For example, our data was at relatively low resolution, and it would be useful to explore how our method performs with higher-resolution data (e.g., 3D likely necessitates a sparser network architecture). Another limitation is that our network is trained for simulations at a particular grid resolution. Through upscaling and downscaling input and output data, we could adapt our pre-trained model to handle data at different resolutions, but we imagine this would produce smaller improvements. Another approach to achieve resolution independence might be a patch-based correction method that is trained on and applied to small fixed-size patches of the computational grid, rather than using the entire grid as a sample. We are generally interested in the question of resolution-agnostic networks and avoiding the need for training new networks for each new problem resolution of interest. Methodologically, we would like to compare our approach of applying a network once, to the approach of correcting a Boussinesq flow simulation at every time step (or every few time steps) via application of a network—this should require the network to make smaller corrections on each time step and would perhaps yield better test errors and generalization. Finally, although the square cavity problems studied in this paper do present substantially different flow behaviors, we believe that our current model's weights would generalize poorly to an arbitrary compressible flow problem. We are ultimately interested in developing datasets of plausible, randomized compressible flow scenarios, and using such datasets to train a network that can generalize to arbitrary Boussinesq/compressible flow problems. We expect that this may require a deeper or wider network and anticipate more thorough model ablation studies in future work.

ACKNOWLEDGMENTS

D.H. was supported by NSF Grant No. 2324735. This work is supported by DARPA through contract number FA8750-20-C-0537. Any opinions, findings, conclusions, or recommendations expressed are those of the authors and do not necessarily reflect the views of the sponsor.

REFERENCES

[1] 2023. Time-Dependent Solver. https://doc.comsol.com/6.1/doc/com.comsol.help.comsol/comsol_ref_solver.35.129.html [Online; accessed 21-February-2024].
[2] Antonio Barletta. 2022. The Boussinesq approximation for buoyant flows. *Mechanics Research Communications* 124 (2022), 103939. https://doi.org/10.1016/j.mechrescom.2022.103939
[3] Antonio Barletta, Michele Celli, and D. Andrew S. Rees. 2023. On the use and misuse of the Oberbeck–Boussinesq approximation. *Physics* 5, 1 (2023), 298–309. https://doi.org/10.3390/physics5010022
[4] Imre Ferenc Barna, Mihály András Pocsai, Sándor Lökös, and László Mátyás. 2017. Rayleigh–Bènard convection in the generalized Oberbeck–Boussinesq system. *Chaos, Solitons & Fractals* 103 (2017), 336–341. https://doi.org/10.1016/j.chaos.2017.06.024
[5] Roger D. Blandford and Kip S. Thorne. 2013. *Applications of Classical Physics*. http://www.pmaweb.caltech.edu/Courses/ph136/yr2012/
[6] Joseph Boussinesq. 1903. *Théorie analytique de la chaleur mise en harmonic avec la thermodynamique et avec la théorie mécanique de la lumière: Refroidissement et échauffement par rayonnement, conductibilité des tiges, lames et masses cristallines, courants de convection, théorie mécanique de la lumière*. Vol. 2. Gauthier-Villars, Paris, France.
[7] Steven L. Brunton, Bernd R. Noack, and Petros Koumoutsakos. 2020. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics* 52 (2020), 477–508. https://doi.org/10.1146/annurev-fluid-010719-060214
[8] Shengze Cai, Zhicheng Wang, Sifan Wang, Paris Perdikaris, and George Em Karniadakis. 2021. Physics-informed neural networks for heat transfer problems. *Journal of Heat Transfer* 143, 6 (2021), 060801. https://doi.org/10.1115/1.4050542
[9] Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. 2022. Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next. *Journal of Scientific Computing* 92, 3 (26 Jul 2022), 88. https://doi.org/10.1007/s10915-022-01939-z
[10] Charles Francis Curtiss and Joseph O. Hirschfelder. 1952. Integration of stiff equations. *Proceedings of the National Academy of Sciences* 38, 3 (1952), 235–243. https://doi.org/10.1073/pnas.38.3.235
[11] Graham de Vahl Davis. 1983. Natural convection of air in a square cavity: a bench mark numerical solution. *International Journal for Numerical Methods in Fluids* 3, 3 (1983), 249–264. https://doi.org/10.1002/fld.1650030305
[12] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. 2015. FlowNet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*. Computer Vision Foundation, Santiago, Chile, 2758–2766. https://doi.org/10.1109/ICCV.2015.316
[13] Amir Barati Farimani, Joseph Gomes, and Vijay S. Pande. 2017. Deep learning the physics of transport phenomena. *arXiv preprint arXiv:1709.02432* (2017). https://arxiv.org/pdf/1709.02432.pdf
[14] Charles W. Gear. 1967. The numerical integration of ordinary differential equations. *Math. Comp.* 21, 98 (1967), 146–156. https://doi.org/10.1090/S0025-5718-1967-0225494-5
[15] Zhenglin Geng, Daniel Johnson, and Ronald Fedkiw. 2020. Coercing machine learning to output physically accurate results. *J. Comput. Phys.* 406 (2020), 109099. https://doi.org/10.1016/j.jcp.2019.109099
[16] Frederic Gibou, David Hyde, and Ron Fedkiw. 2019. Sharp Interface Approaches and Deep Learning Techniques for Multiphase Flows. *J. Comput. Phys.* 380 (2019), 442–463. https://doi.org/10.1016/j.jcp.2018.05.031
[17] Donald D. Gray and Aldo Giorgini. 1976. The validity of the Boussinesq approximation for liquids and gases. *International Journal of Heat and Mass Transfer* 19, 5 (1976), 545–551. https://doi.org/10.1016/0017-9310(76)90168-X
[18] Xiaoxiao Guo, Wei Li, and Francesco Iorio. 2016. Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 481–490. https://doi.org/10.1145/2939672.2939738
[19] Yuqing Hou, Hui Li, Hong Chen, Wei Wei, Jiayue Wang, and Yicang Huang. 2022. A novel deep U-Net-LSTM framework for time-sequenced hydrodynamics

prediction of the SUBOFF AFF-8. *Engineering Applications of Computational Fluid Mechanics* 16, 1 (2022), 630–645. https://doi.org/10.1080/19942060.2022.2030802

[20] Fan Jia, Wing Hong Wong, and Tieyong Zeng. 2021. DDUNet: Dense dense U-Net with applications in image denoising. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 354–364. https://doi.org/10.1109/ICCVW54120.2021.00044

[21] Ayano Kaneda, Osman Akar, Jingyu Chen, Victoria Alicia Trevino Kala, David Hyde, and Joseph Teran. 2023. A Deep Conjugate Direction Method for Iteratively Solving Linear Systems. In *Proceedings of the 40th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 202)*, Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (Eds.). PMLR, 15720–15736.

[22] George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. 2021. Physics-informed machine learning. *Nature Reviews Physics* 3, 6 (2021), 422–440. https://doi.org/10.1038/s42254-021-00314-5

[23] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). https://arxiv.org/pdf/1412.6980.pdf

[24] Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W. Mahoney. 2021. Characterizing possible failure modes in physics-informed neural networks. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., Red Hook, New York, 26548–26560.

[25] Mario Lino, Stathi Fotiadis, Anil A. Bharath, and Chris D. Cantwell. 2023. Current and emerging deep-learning methods for the simulation of fluid dynamics. *Proceedings of the Royal Society A* 479, 2275 (2023), 20230058. https://doi.org/10.1098/rspa.2023.0058

[26] Lu Lu, Raphael Pestourie, Wenjie Yao, Zhicheng Wang, Francesc Verdugo, and Steven G. Johnson. 2021. Physics-informed neural networks with hard constraints for inverse design. *SIAM Journal on Scientific Computing* 43, 6 (2021), B1105–B1132. https://doi.org/10.1137/21M1397908

[27] Kevin Luna, Katherine Klymko, and Johannes P. Blaschke. 2021. Accelerating GMRES with deep learning in real-time. *arXiv preprint arXiv:2103.10975* (2021). https://arxiv.org/pdf/2103.10975.pdf

[28] Richard C. Martineau, Ray A. Berry, Aurélia Esteve, Kurt D. Hamman, Dana A. Knoll, Ryosuke Park, and William Taitano. 2009. *Comparative analysis of natural convection flows simulated by both the conservation and incompressible forms of the Navier-Stokes equations in a differentially-heated square cavity.* Technical Report. Idaho National Lab.(INL), Idaho Falls, ID (United States).

[29] Peyman Mayeli and Gregory J. Sheard. 2021. Buoyancy-driven flows beyond the Boussinesq approximation: A brief review. *International Communications in Heat and Mass Transfer* 125 (2021), 105316. https://doi.org/10.1016/j.icheatmasstransfer.2021.105316

[30] Sandip Mazumder. 2006. On the Use of the Fully Compressible Navier-Stokes Equations for the Steady-State Solution of Natural Convection Problems in Closed Cavities. *Journal of Heat Transfer* 129, 3 (06 2006), 387–390. https://doi.org/10.1115/1.2430726

[31] Jinho Park. 2019. Jinh0park/Pytorch-SSIM-3D: Pytorch structural similarity (SSIM) loss for 3D images. https://github.com/jinh0park/pytorch-ssim-3D

[32] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems* 32 (2019).

[33] Saket Patkar, Mridul Aanjaneya, Wenlong Lu, Michael Lentine, and Ronald Fedkiw. 2016. Towards positivity preservation for monolithic two-way solid–fluid coupling. *J. Comput. Phys.* 312 (2016), 82–114. https://doi.org/10.1016/j.jcp.2016.02.010

[34] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. 2017. Physics informed deep learning (part I): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561* (2017). https://arxiv.org/pdf/1711.10561.pdf

[35] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics* 378 (2019), 686–707. https://doi.org/10.1016/j.jcp.2018.10.045

[36] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer, 234–241. https://doi.org/10.1007/978-3-319-24574-4_28

[37] Francisco J. Saez, Patricio A. Catalan, and Carlos Valle. 2021. Wave-by-wave nearshore wave breaking identification using U-Net. *Coastal Engineering* 170 (2021), 104021. https://doi.org/10.1016/j.coastaleng.2021.104021

[38] Rishi Sharma, Amir Barati Farimani, Joe Gomes, Peter Eastman, and Vijay Pande. 2018. Weakly-supervised deep learning of heat transport via physics informed loss. *arXiv preprint arXiv:1807.11374* (2018). https://arxiv.org/pdf/1807.11374.pdf

[39] Nahian Siddique, Sidike Paheding, Colin P. Elkin, and Vijay Devabhaktuni. 2021. U-Net and its variants for medical image segmentation: A review of theory and applications. *IEEE Access* 9 (2021), 82031–82057. https://doi.org/10.1109/ACCESS.2021.3086020

[40] Nils Thuerey, Konstantin Weißenow, Lukas Prantl, and Xiangyu Hu. 2020. Deep learning methods for Reynolds-averaged Navier–Stokes simulations of airfoil flows. *AIAA Journal* 58, 1 (2020), 25–36. https://doi.org/10.2514/1.J058291

[41] Jonathan Tompson, Kristofer Schlachter, Pablo Sprechmann, and Ken Perlin. 2017. Accelerating Eulerian fluid simulation with convolutional networks. In *International Conference on Machine Learning*. PMLR, 3424–3433. http://proceedings.mlr.press/v70/tompson17a/tompson17a.pdf

[42] Kiwon Um, Robert Brand, Yun Raymond Fei, Philipp Holl, and Nils Thuerey. 2020. Solver-in-the-loop: Learning from differentiable physics to interact with iterative PDE-solvers. *Advances in Neural Information Processing Systems* 33 (2020), 6111–6122.

[43] Kush R. Varshney. 2016. Engineering safety in machine learning. In *2016 Information Theory and Applications Workshop (ITA)*. IEEE, IEEE, 1–5. https://doi.org/10.1109/ITA.2016.7888195

[44] Jan Vierendeels, Bart Merci, and Erik Dick. 2001. Numerical study of natural convective heat transfer with large temperature differences. *International Journal of Numerical Methods for Heat & Fluid Flow* 11, 4 (2001), 329–341. https://doi.org/10.1108/09615530110389117

[45] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* 13, 4 (2004), 600–612. https://doi.org/10.1109/TIP.2003.819861

[46] Jared Willard, Xiaowei Jia, Shaoming Xu, Michael Steinbach, and Vipin Kumar. 2020. Integrating physics-based modeling with machine learning: A survey. *arXiv preprint arXiv:2003.04919* 1, 1 (2020), 1–34. https://arxiv.org/pdf/2003.04919.pdf

[47] Yanwen Xu, Sara Kohtz, Jessica Boakye, Paolo Gardoni, and Pingfeng Wang. 2023. Physics-informed machine learning for reliability and systems safety applications: State of the art and challenges. *Reliability Engineering & System Safety* 230 (2023), 108900. https://doi.org/10.1016/j.ress.2022.108900

[48] Cheng Yang, Xubo Yang, and Xiangyun Xiao. 2016. Data-driven projection method in fluid simulation. *Computer Animation and Virtual Worlds* 27, 3-4 (2016), 415–424. https://doi.org/10.1002/cav.1695

[49] Radyadour Kh. Zeytounian. 2003. Joseph Boussinesq and his approximation: a contemporary view. *Comptes Rendus Mécanique* 331, 8 (2003), 575–586. https://doi.org/10.1016/S1631-0721(03)00120-7

[50] Xiangxiong Zhang and Chi-Wang Shu. 2012. Positivity-preserving high order finite difference WENO schemes for compressible Euler equations. *J. Comput. Phys.* 231, 5 (2012), 2245–2258. https://doi.org/10.1016/j.jcp.2011.11.020

[51] Qiming Zhu, Zeliang Liu, and Jinhui Yan. 2021. Machine learning for metal additive manufacturing: predicting temperature and melt pool fluid dynamics using physics-informed neural networks. *Computational Mechanics* 67 (2021), 619–635. https://doi.org/10.1007/s00466-020-01952-9