Abstract—Deep Learning (DL) algorithms are emerging as a key alternative to computationally expensive CFD simulations. However, state-of-the-art DL approaches require large and high-resolution training data to learn accurate models. The size and availability of such datasets are a major limitation for the development of next-generation data-driven surrogate models for turbulent flows. This paper introduces SURFNet, a transfer learning-based super-resolution flow network. SURFNet primarily trains the DL model on low-resolution datasets and transfer learn the model on a handful of high-resolution flow problems — accelerating the traditional numerical solver independent of the input size. We propose two approaches to transfer learning for the task of super-resolution, namely one-shot and incremental learning. Both approaches entail transfer learning on only one geometry to account for fine-grid flow fields requiring 15× less training data on high-resolution inputs compared to the tiny resolution (64×256) of the coarse model significantly, reducing the time for both data collection and training.

We empirically evaluate SURFNet’s performance by solving the Navier-Stokes equations in the turbulent regime on input resolutions up to 256× larger than the coarse model. On four test geometries and eight flow configurations unseen during training, we observe a consistent 2 – 2.1× speedup over the OpenFOAM physics solver independent of the test geometry and the resolution size (up to 2048×2048), demonstrating both resolution-invariance and generalization capabilities. Moreover, compared to the baseline model (aka oracle) that collects large training data at 256×256 and 512×512 grid resolutions, SURFNet achieves the same performance gain while reducing the combined data collection and training time by 3.6× and 10.2×, respectively. Our approach addresses the challenge of reconstructing high-resolution solutions from coarse grid models trained using low-resolution inputs (i.e., super-resolution) without loss of accuracy and requiring limited computational resources.

Index Terms—ML for science, computational fluid dynamics, deep learning, super-resolution, transfer learning, turbulent flows

I. INTRODUCTION

Computational Fluid Dynamics (CFD) simulations that solve the complex Navier-Stokes equations are ubiquitous in both laminar and turbulent flows [1, 2, 3, 4, 5, 6, 7]. Engineering systems of interest such as aerospace design exploration (e.g., designing airfoils for aircraft wings) require resolving fine-scale physics in the turbulent regime to produce high-fidelity solutions. To account for more aspects of the physical phenomena being modeled necessitates an increase in the resolution of the system, resulting in high computational costs. Figure 1 shows the time-to-solution with increasing grid size for a turbulent flow simulation around a National Advisory Committee for Aeronautics (NACA) airfoil. A dual-socket, 40-core system requires ten seconds at a resolution of 64×256, and 100 minutes at 2048×2048 for a single airfoil shape in one flow configuration (one test case). In aircraft design, there are typically many airfoil shapes [8], each of which requires simulations performed across a range of parameters such as Reynolds (Re) numbers, various angles of pitch, yaw, and roll to account for rotation as well as different angles of attack, resulting in thousands of test cases. To address this challenge, super-resolution is an approach to reconstruct fine-scale flow physics from coarse-grid solutions.

Figure 1: Time to convergence for flow around a NACA4415 airfoil at different spatial resolutions on a dual-socket Intel Xeon Gold 6148 CPU (total 40 cores).

Although mathematical models such as Large Eddy Simulations (LES) [9] and Reynolds-Averaged Navier-Stokes (RANS) [5] that relate coarse-scale physical effects with fine-scale solutions are common place, most recent applications of super-resolution employ deep learning (DL) [10, 11, 12, 13, 14, 15, 16, 17]. This popularity of DL is due to its tremendous success in natural language processing [18] and computer vision [19, 20].

Table 1 summarizes the state-of-the-art approaches for accel-
Table I: Comparing state-of-the-art approaches in DL for 2D CFD simulations on nine different features. SURFNet is a novel network-based transfer learning (TL) framework that (1) generates accurate solutions up to 2048 × 2048 spatial resolutions for turbulent flows from low-resolution models, (2) generalizes to unseen-in-training geometries, (3) meets the original convergence constraints of traditional CFD solvers, and (4) collects data at low-resolution on coarse grids for training – whereas prior works target non-turbulent or non-viscous flows, only test on geometry domains that were part of the training phase, replace partly traditional solvers with a neural network surrogate not meeting convergence constraints, and most importantly, train with data downsampled from high-resolution simulations or are unsupervised. NO stands for Neural Operator and CNN for Convolutional Neural Network.

| Related Work | Target Flow (PDE) | Test Geometry Unseen in Training | Meets convergence constraints | Resolution Invariant | Training data collected on coarse-grids | Highest Test Spatial Resolution | Error metric | Error value (best) | Technique |
|--------------|-----------------|----------------------------------|-----------------------------|--------------------|--------------------------------------|---------------------------------|-------------|-------------------|-----------|
| Fourier Neural Operator | Darcy | ● | ● | ● | 421 × 421 | RE | 1 × 10⁻² | NO |
| Fourier Neural Operator | Navier-Stokes | ● | ● | ● | 256 × 256 | RE | 8.6 × 10⁻³ | NO |
| Graph Kernel Network | Darcy | ● | ● | ● | 241 × 241 | Relative L₂ | 3.7 × 10⁻² | NO |
| Bhattacharya et al. | Darcy | ● | ● | ● | 421 × 421 | RE | 2 × 10⁻³ | NO |
| MeshFreeFlowNet | Rayleigh-Bénard | ● | ● | ● | 4 × 512 | NMAE | 3.3 × 10⁻³ | CNN |
| Guo et al. | Laminar | ● | ● | ● | 200 × 200 | RE | 0.025 | CNN |
| TFNet | Rayleigh-Bénard | ● | ● | ● | 1792 × 256 | RMSE | 200 | CNN |
| Guo et al. | Laminar | ● | ● | ● | 128 × 256 | RME | 1.76% | CNN |
| CFFNet | Navier-Stokes | ● | ● | ● | 64 × 256 | RME | 0% | CNN |
| Smart-fluidnet | Eulerian | ● | ● | ● | 1024 × 1024 | MAE | 9 × 10⁻³ | CNN |
| SURFNet (this paper) | Navier-Stokes | ● | ● | ● | 2048 × 2048 | RME | 0% | TL |

To address the above need, we develop SURFNet (Super-Resolution Flow Network), a novel approach to reconstruct fine-scale flow physics from coarse grid data by primarily training the DL model on low-resolution inputs. Using this coarse-model (CM), SURFNet transfer learns the model on high-resolution turbulent flow solutions, significantly reducing the overall data collection time and the total size of the training set. More specifically, this paper makes the following contributions:

- First, we present an 8-layer CNN that is adequately expressive to capture different geometries and flow configurations. The CNN is trained on low-resolution inputs (64 × 256 grid size) using ten geometries and nine variations of each geometry to produce a coarse-model (CM) (Section III-B).

- Then, to enable efficient super-resolution, we propose two variations of transfer learning – 1) one-shot transfer learning (OSTL): learning is conducted from CM to the target resolution in a single shot, and 2) incremental transfer learning (ITL): learning is done step-by-step incrementally on the training set of intermediate discretizations up to the target resolution (Section III-D). We show that one-shot learning is inadequate for reaching oracle (i.e., a full model trained at high resolutions) accuracy, and incremental learning where the model is fine-tuned with data from intermediate discretizations is critical for reaching best-case speedups, especially at higher target discretizations (Section V-A).

- We empirically evaluate SURFNet by solving the RANS equations in the turbulent regime on four geometries and eight flow configurations unseen during training. SURFNet achieves a consistent speedup of 2 - 2.1 × with ITL at up to 2048 × 2048 spatial resolutions over the OpenFOAM physics solver independent of the resolution size and test geometry, demonstrating both resolution-invariance and generalization capabilities (Section V-B).

- SURFNet eliminates the need to collect exhaustive training datasets at high resolutions to account for fine-
scale physical phenomena. This computational efficiency enables SURFNet to achieve oracle accuracies while significantly reducing the size of the training dataset by $15 \times$, consequently reducing the combined data collection and training time by $3.6 \times$ and $10.2 \times$, respectively at $256 \times 256$ and $512 \times 512$ grid sizes (Section V-C).

II. BACKGROUND

We use the steady incompressible RANS equations to solve turbulent flows. The RANS governing equations describe the fluid motion as follows:

$$\frac{\partial U_i}{\partial x_i} = 0$$ (1)

$$\frac{\partial U_j}{\partial x_j} = - \frac{\partial (p)}{\partial x_j} + (\nu + \nu_t) \left[ \frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} \right]$$ (2)

where $\bar{U}$ is the mean velocity, $\bar{p}$ is the kinematic mean pressure, $\nu$ is the fluid viscosity, and $\nu_t$ is the eddy viscosity. The RANS equations provide a time-averaged solution to the incompressible Navier-Stokes equations at the expense of a non-closed equation. Closing the equation is usually done through Boussinesq’s approximation [29], which yields the eddy viscosity. The eddy viscosity is found through turbulence modeling. The Spalart-Allmaras one-equation model yields a non-closed equation. Closing the equation is usually at higher resolutions.

A. Accelerating the convergence of fluid simulations

To accelerate the numerical convergence of the equations in Section II, we define a finite-dimensional parametric map $G$ such that $x^N = G(x^I; \theta_{wl})$ where $x^N, x^I \in \mathbb{R}^{m \times n \times z}$ represent the tensors of $z$ flow variables on a $m \times n$ grid domain at convergence and any intermediate iteration respectively, $N$ is the number of iterations required for convergence with the physics solver, and $I$ ($0 \leq I \leq N - 1$) is any intermediate iteration of the solver.

At each $I$, the physics solver computes four flow fields in the 2D grid domain of the simulation (i.e., $z = 4$): the x-velocity component ($U$), the y-velocity component ($V$), the pressure ($P$), and the modified eddy viscosity ($\nu_t$). If the grid resolution is $m \times n$, each field is of size $m \times n$. We concatenate the four flow variables together, generating an input tensor, $x^I$ of size $m \times n \times 4$. The output tensor has the same shape as the input tensor. The difference between the input and the output is that the latter has the flow field values at steady-state, $x^N$. As a result, $G$ parameterizes the solution operator that takes an intermediate flow field and maps it to its steady-state solution.

A popular candidate for learning $G$ is CNNs. However, as seen from Table I, the majority of the CNN-based approaches present a solution that does not satisfy the conservation laws [22, 23, 30, 31], which is sub-optimal for CFD practitioners. There have been recent attempts to provide physically consistent solutions. However, these methods either severely restrict the generalization capacity of the network to specific flow configurations [13, 24] or present boundary-constrained loss functions that require training a new model for every new instance of the same fluid problem [32, 33, 34, 35].

To circumvent these limitations, other works [23, 30] train a domain-agnostic DL model and subsequently augment the model’s inference with a refinement stage, where the physics solver constrains the model’s prediction to reach accelerated convergence. Refinement consists of (a) feeding the CNN’s prediction back into the physics solver, (b) imposing the boundary conditions, and (c) running the solver to convergence. We adhere to the latter approach because it allows us to retain the advantages of domain-agnostic training while simultaneously meeting the original convergence constraints of the physics solver. The solution, therefore, has a 0% relative mean error.

B. Coarse Grid Network

We aim to learn a generalizable model that can predict flow around rotated geometries (i.e., varying pitch angle, $\theta$) and changing flow fields (i.e., angle of attack, $\alpha$) as they are critical features in design exploration and shape optimization. For instance, it is common practice to simulate the flow field by varying the angle of the incoming flow and rotating the airfoil geometry [31].

To capture such complex flow phenomena, we design an eight-layer, convolutional-deconvolutional neural network (4 convolution layers followed by 4 deconvolution layers). The number of filters are, in order: 16, 32, 128, 256, and 256, 128, 32, 16. An illustration can be seen in Figure 2. The choice
of the architecture is motivated by two reasons. First, recent works have successfully leveraged similar designs for physical system emulation [22, 23]. Second, the convolution operator is an optimal candidate to extract existing spatial correlations in and among the fluid variables. We use a deeper network compared to other approaches [22, 23] (8- vs 6-layer CNN) because the training dataset is larger and has more features to learn (e.g., rotation). All layers have a filter size of $5 \times 5$ and a stride of 1. This overlap captures both the short and long (spatial) range dependencies between the flow variables while covering all regions of the flow field present in the input image. We use the LeakyReLU activation function for all layers because the output image contains real-valued variables.

C. Why Transfer Learning?

The CNN in Figure 2 relies on intermediate and final solutions of flow simulations to train the solution operator, $G$. The ideal scenario in machine learning is the availability of vast amounts of labeled training data for supervised models. However, both collection of large-scale high-resolution data and training with large input sizes are prohibitively expensive (see Figure 1). An alternate approach is to use the convolutional operator calibrated with low-resolution data to predict high-resolution solutions. Although this is feasible, the accuracy reduces dramatically, especially for turbulent flows as we show empirically in Section V. Therefore, CNN-based approaches demand end-to-end re-training for different resolutions and discretizations to achieve constant error.

The above limitations motivate the use of transfer learning (TL) to both solve the problem of insufficient training data at higher resolutions and to achieve resolution-invariance across discretizations. TL is a popular technique that relies on transferring a trained model across different learning tasks or from one model to another [37, 38]. It has been successfully applied to different applications such as drug discovery [39], disease detection [40], natural language processing [41], and machine fault diagnosis [42].

We propose another novel application of TL that leverages the correlations inherent between low- and high-resolution inputs of turbulent flows. Low-resolution solutions of fluid simulations contain critical information that can be effectively extracted for high-resolution flow prediction: low-resolution solutions capture all present flow structures [2], even complex ones (for instance, the wake region behind solid bodies after flow separation). However, they do not resolve them, as high-resolution grids do. Therefore, given a pre-trained CNN model on large datasets at low-resolution (i.e., coarse model), our objective is to transfer the trained model to the target fine-grid discretization to append the unresolved flow information. An advantage of our super-resolution TL methodology is that the input is truly from coarse-grid simulations, not downsamples from high-resolution data as in the state-of-the-art [14, 16, 17]. This eliminates the requirement for generating computationally demanding (and in some cases intractable) data to build a high-resolution model. In the rest of this section, we describe the transfer learning pipeline and how we account for fine-scale physics and dynamics.

D. Super-Resolution with Transfer Learning

Given $r_c : x_c \times y_c$, where $r_c$ is the coarse-grid resolution corresponding to the discretization $x_c \times y_c$, the super-resolution task of SURFNet is to recover solutions at fine-scales $r_f : x_f \times y_f \mid f = 1, \ldots, t$ where $x_f \times y_f$ is a fine-grid discretization and $r_f > r_c, \forall f$. We implement network-based transfer learning as illustrated in Figure 3 and propose two variations for super-resolution as described below.

One-Shot Transfer Learning (OSTL). In this approach, model weights are transferred from the CM trained with $r_c$ to the target resolution, $r_t$ in a single training step. For example, if we desire to recover high-resolution flow fields at $2048 \times 2048$, we transfer learn from $64 \times 256$ to $2048 \times 2048$ in a single shot. This approach is illustrated in Figure 3 with a red, dashed line.

Incremental Transfer Learning (ITL). An alternative approach to OSTL is to train the CNN with the large-scale low-resolution dataset and perform TL in a step-wise manner. That is, instead of transfer learning from $r_c$ to the target resolution directly, $r_t$, we pass through intermediate resolutions step-by-step from the low resolution to the target high resolution at

$\nu(v)$. The difference between the input and the output tensors is that the former has the flow values of an intermediate iteration (i), while the latter is the flow field at steady-state (ss).

Figure 2: CNN and its input-output representation. The CNN is a symmetric, fully convolutional-deconvolutional neural network. The input and the output are a 4-channel tensor image: each channel represents one flow variable (x-velocity $U$, y-velocity $V$, pressure $P$ and modified eddy viscosity $\nu$). The difference between the input and the output tensors is that the former has the flow values of an intermediate iteration (i), while the latter is the flow field at steady-state (ss).
Figure 3: SURFNet (left) and its inductive transfer learning (right) for super-resolution of turbulent flows. A large dataset is collected at low resolutions to train the CNN and obtain the coarse model. Small data is collected at higher resolutions and SURFNet transfers the model weights from the coarse model to train the fine-scale model using either one-shot (red) or incremental transfer learning (black).

(i.e., \( r_c \rightarrow r_1 \rightarrow r_2 \rightarrow \ldots \rightarrow r_f \)). For example, if we desire to recover high-resolution flow fields at \( 512 \times 512 \), we transfer learn from \( 64 \times 256 \) to \( 256 \times 256 \), and finally from \( 256 \times 256 \) to \( 512 \times 512 \). A step size of 1 allows the model to incorporate new information from each intermediate discretization. We further discuss the impact of step size on the predictive accuracy in Section V. The black dashed line in Figure 3 shows this variation of TL. In ITL, the model learns from more data than OSTL without overfitting while still reducing the overall data collection since we avoid heavy data collection at any specific high resolution.

To recover the solution at the desired target discretization, we perform the steps outlined below for both transfer learning approaches (i.e., OSTL and ITL).

1. Coarse-grid data collection. The training dataset is created by performing large-scale simulations at low-resolution discretizations, \( x_c \times y_c \), using the physics solver detailed in Section IV-C.

2. Training. After generating low-resolution training data, the CNN in Figure 2 is trained with this large dataset. By carefully controlling for under- and over-fitting of the network with a validation dataset, we obtain the coarse model.

3. Fine-grid data collection. Due to the challenge of data acquisition at fine-scale discretizations, \( x_f \times y_f \), we limit the number of simulations at high resolutions to the bare minimum to create the transfer dataset. In Section IV we show that a single geometry is sufficient to transfer the coarse-grid features and recover the fine-scale physics.

4. Transfer learning. In this paper, we choose to implement inductive transfer learning, where we reuse the network (including its structure and parameters) that was pre-trained to learn the source model (i.e., CM) \(^{14}\). Since the source and target domains are the same, we intuitively expect this technique to preserve the common features extracted between the two learning tasks. We treat the coarse model as a feature extractor of high-resolution flow fields. All layers of the CNN use a stride of 1 to not reduce the dimensionality of the input-to-output map during the low-resolution training phase. However, for high-resolution input-output pairs, this model is a low-dimensional representation – an extractor of the prevalent flow features across discretizations (e.g., flow effects due to the boundary conditions, fields’ shape in the free stream, and flow variations due to the change in \( \alpha \) and \( \theta \)). Since coarse discretizations do not have enough domain points to define accurate flow field solutions in areas of strong gradients, the features extracted need to be fine-tuned but not re-learned. The transferred network is updated by fine-tuning the weights as follows (see Figure 3).

   a. Re-initialize the transferred network with the weights obtained from the CM for OSTL. For ITL, the transferred network is initialized with the weights from the largest model pre-trained at resolution \( r_f \) such that \( r_f < r_t \).

   b. Start training the transferred network with the transfer dataset. At this stage, it is critical to append the fine-grid flow features. Since we start with a good initial calibration of the weights from a pre-trained model of the same domain, only fine-tuning is required to update the weights of the transferred network. Fine-tuning of network parameters is done with a low learning rate (i.e., small updates to the weights) and for very few epochs (1 or 2 at most) to avoid overfitting to the geometry (or geometries) of the small transfer dataset while preserving the generalization capacity of the model.

In summary, SURFNet is a TL-based super-resolution flow network that learns three distinct CNN models to reconstruct high-resolution turbulent flow fields – (1) the low-resolution CM, (2) the OSTL model, and (3) the ITL model.

IV. EXPERIMENT SETUP

In this section, we first describe the case study to evaluate SURFNet’s potential for super-resolution. Then, we describe the low- and high-resolution datasets for training, transfer
learning, validation, and testing and outline the training process of the CM.

A. Case Study

We consider external aerodynamics as the paradigm for aerospace design space exploration. Understanding flow around solid bodies is an important research topic for industrial aerospace applications, and airfoils are the core geometries in aerodynamics studies. In real scenarios, the exploration involves different geometries (for instance, airfoil shapes) simulated under various flow configurations such as rotation of the solid body and wind angle. Therefore, to apply to a large class of CFD problems, challenges in generalizing to unseen geometries need to be addressed. In this paper, we aim to evaluate SURFNet’s ability to accelerate the flow around solid bodies such as airfoils and cylinder. Accordingly, these geometries are excluded from the training and transfer learning datasets. We resolve turbulent flow around solid bodies at a Reynolds number of $6 \times 10^5$ using the equations presented in Section IV-C. This setup - the flow regime (turbulent flow), the geometries (extrapolating to NACA airfoils), and the grid resolutions - is representative of real NASA case studies [21].

B. Dataset Creation

We create a total of 15 distinct datasets and perform the simulations of all flow configurations with the solver described in Section IV-C. Table II summarizes the datasets, including the composition, resolution, and scope of each dataset.

Table II: Summary of datasets. The training dataset is from low-resolution simulations. At all high resolutions we collect identical transfer, validation, and test datasets. NoG is for the number of different geometries, and NoFC is for the number of flow configurations (i.e., total number of simulations).

| Datasets   | Low resolution | Higher resolutions |
|------------|----------------|--------------------|
| Training   |                |                    |
| NoG NoFC Type | 10, 90 ellipses | 1                  |
|            |                | ellipse $AR = 0.1$ |
| Transfer   |                |                    |
| NoG NoFC Type | 3              | 6                  |
|            |                | ellipse $AR = 0.1$ |
| Validation |                |                    |
| NoG NoFC Type | NACA0012, ellipse $AR = 0.22$, cylinder |
| Test       |                |                    |
| NoG NoFC Type | 4              | 8                  |
|            |                | NACA1412, NACA0015, ellipse $AR = 0.3$, cylinder |

**Training dataset.** First, we collect a large-scale, low-resolution dataset to train the coarse model (CM), at a $64 \times 256$ resolution (a common resolution for low-resolution solutions of similar case studies [21]). The core of this dataset is formed by simulations of flow around 10 different ellipses obtained by changing the aspect ratio $AR$ that is defined as the ratio of the vertical to the horizontal semiaxis length, as shown in Figure 4. The $AR$’s considered for the training dataset are: 0.05, 0.07, 0.09, 0.1, 0.15, 0.2, 0.25, 0.35, 0.55, and 0.75. For each ellipse, we consider 9 flow variations: five different angles of attack, $\alpha$, and four different pitch angles, $\theta$, chosen randomly for each ellipse in the range between $-2 - 6^\circ$ for a total of 90 flow configurations. We choose from thin to thick ellipses and angles of attack to cover a wide spectrum of physical phenomena that is commonplace in aerospace design exploration. For instance, the angle of attack is an important variable in determining the magnitude of the force of lift. The $\alpha$ angles are obtained by changing the direction of the flow while maintaining the angle between the chord of the solid body and the cartesian x-direction at $0^\circ$. The $\theta$ angles are obtained by pitching the nose of the solid body up or down and maintaining a flow direction at a 0-degree angle with its longitudinal axis. These configurations are illustrated in Figure 4.

**Test dataset.** We collect a new dataset to evaluate SURFNet’s performance in accelerating high-resolution turbulent flow simulations. The test datasets consist of flow around four geometries: a non-symmetric NACA1412 airfoil, a symmetric NACA0015 airfoil, an ellipse $AR = 0.3$, and
a cylinder. The airfoil and cylinder geometries are shown in Figure 5. The test dataset contains geometries unseen during the pre-training or transfer learning phases. Nonetheless, some geometries are, a priori, more challenging than others. For example, the non-symmetric NACA1412 airfoil has three unique features distinct from the training dataset: the flat trailing edge, the non-symmetry, and the chamber thickness (12% of the chord of the airfoil). Comparing the airfoil to the ellipse in the test set, the only feature unseen during the training phase is its chamber thickness ($AR = 0.3$). Moreover, even though a cylinder is a special case of an ellipse from a geometrical perspective, the physics in the rear of the cylinder has a large recirculation area not present in the training flows. For each geometry, we consider 2 flow variants: one $\alpha$ angle and one $\theta$ angle. Table III summarizes the different test cases. The test dataset is collected at all resolutions.

Figure 5: Non-symmetric NACA1412 airfoil (left), symmetric NACA0015 airfoil (center), and cylinder (right) as test geometries. The last two digits in the 4-digit NACA denomination represent the maximum thickness percentage of the chamber of the airfoil with respect to the airfoil’s chord (dashed line).

**C. Physics Solver**

The training, transfer, validation, and test datasets are generated by solving the RANS equations with the Spalart-Allmaras one-equation model [29]. We use the incompressible solver simpleFoam from OpenFOAM v8 as the physics solver that implements the semi-implicit method for pressure-linked equations (SIMPLE) algorithm. A residual value below $1 \times 10^{-4}$ is the criteria to consider the training simulations converged. We adhere to tolerances that are extended practice in CFD simulations [43]. We use the GAMG solver for computing the pressure at every iteration, with a tolerance of $1 \times 10^{-8}$. The smoothSolver and the GaussSeidel smoother compute both the velocity and modified eddy viscosity.

**Architecture and Libraries.** All the OpenFOAM simulations are run in parallel on a dual-socket Intel Xeon Gold 6148 using double precision. Each socket has 20 cores, for a total of 40 cores. We use the OpenMPI implementation of MPI integrated with OpenFOAM v8 that is optimized for shared-memory communication. The grid domain is decomposed into 40 partitions using the integrated Scotch partitioner and each partition is assigned to 1 MPI process that is pinned to a single core. We set the numactl -localalloc flag to bind each MPI process to its local memory.

**D. Coarse Model Training**

We train the CNN in Figure 2 using the training dataset described in Section IV-B using double precision. We implement the CNN using Keras [45] and perform distributed training on four Tesla V100 GPUs connected with PCIe, using the TensorFlow 2.4 backend. No specific initialization is used in training. The batch size is 64, the optimizer is Adam, and the loss function is mean squared error (MSE). The learning rate is set to $1 \times 10^{-4}$ with no decay for all training. The training is stopped using the EarlyStopping Keras callback [46] by monitoring the validation loss with patience of 6 epochs. After 41 epochs, the training loss reaches $7 \times 10^{-4}$ and validation loss reaches a value of $8 \times 10^{-4}$.

**V. RESULTS AND DISCUSSION**

After pre-training and validating the CM, we perform fine-tuning using both transfer learning approaches (i.e., OSTL and ITL) and evaluate SURFNet’s ability for super-resolution at various target high-resolution discretizations. We start by empirically demonstrating the inefficiency of CM without fine-tuning to recover high-resolution turbulent flows, especially at fine-scale discretizations. Then, we evaluate SURFNet’s TL and its ability to generalize to geometries unseen during the pre-training and fine-tuning phases. Finally, we evaluate its performance in reconstructing high-resolution turbulent flows with respect to the OpenFOAM physics solver. Besides, we also compare SURFNet against the baseline model (aka oracle), which performs full training with a large training dataset collected at higher resolutions.

**A. Validation Loss**

One of our objectives is to maintain prediction accuracy across discretizations to build a resolution-invariant DL algorithm. Figure 6 shows the validation loss of the different models with increasing resolution size.

**CM loss.** We observe that the validation loss of CM increases significantly with increasing resolution size from $1.5 \times 10^{-3}$ at 256 x 256 to $2.1 \times 10^{-1}$ at 2048 x 2048. These results indicate that CM trained with low-resolution data is unable to recover high-resolution flow fields. This lack of fidelity is because coarse discretizations learned with CM are incapable of capturing sharp gradients and resolving flow instabilities prevalent at fine discretizations. Hence, CM alone is inadequate for super-resolution.

!!!Figure 6: Coarse model (CM), one-shot transfer learning (OSTL), incremental transfer learning (ITL), and baseline model (BM) losses on the validation dataset at every target resolution.!!!

**OSTL loss.** To augment the prediction capabilities of CM, we perform one-shot TL from CM to the target high-resolution using the transfer dataset as discussed in Section III. We set the learning rate to one order of magnitude lower than the training learning rate for CM (from $1 \times 10^{-4}$ to $1 \times 10^{-5}$), and train

---

3We also considered a 10- and 12-layer CNN. However, adding additional layers did not improve performance.
for only one or two epochs. The loss of the fine-tuned OSTL model on the validation dataset significantly reduces at all target resolutions compared to CM as seen from Figure 6. At $256 \times 256$, OSTL reduces the validation loss from $1.5 \times 10^{-2}$ for CM to $9 \times 10^{-3}$, resulting in 60% overall reduction. We observe a similar trend with subtle improvements at larger resolutions. At $2048 \times 2048$ the validation loss reduces from $2 \times 10^{-1}$ for CM to $3 \times 10^{-2}$ for OSTL – almost one order of magnitude.

We make two main observations from the OSTL results. First, the gap in the validation losses between OSTL and CM increases with the grid size. Coarse discretizations do not contribute sufficient points in the domain to obtain accurate solutions. In areas of strong gradients, fine discretizations shape the flow field very differently from coarse discretizations as seen from Figures 7 and 8 for the NACA airfoils. The finer the discretization, the more distinct the resulting flow field is from the baseline low resolutions. Therefore, when we transfer from CM to very high resolutions (e.g., $2048 \times 2048$), the new flow field seen during the transfer phase produces sizable changes to the weights of the network compared to transferring to "mid-resolutions" (e.g., $256 \times 256$). Second, this divergence of the flow features between low- and high-resolution discretizations also explains why the OSTL validation loss increases with the grid size. Although OSTL substantially improves the accuracy of super-resolution, it still does not achieve resolution-invariance.

To alleviate these limitations, we consider three alternatives. First, to train for more epochs during fine-tuning. However, this comes with the risk of overfitting to the unique geometry in the transfer dataset, which is undesirable. Second, to include more geometries in the transfer dataset. This approach is also detrimental because it would require substantial data collection at high resolutions. Third, use incremental transfer learning (ITL). This approach is promising as the model is further fine-tuned with data at intermediate resolutions until the target discretization. We choose a step size of 1 to avoid the drawbacks of OSTL described above and improve the accuracy of the network at very high resolutions while still reducing the overall data collection.

**ITL loss.** SURFNet does incremental transfer learning using the same approach as in OSTL. Figure 6 plots the error of SURFNet after ITL on the validation dataset at each target resolution. SURFNet’s OSTL and ITL approaches produce different validation loss values at every target resolution except $256 \times 256$. ITL improves the generalizability of the network. The validation loss drops by half at $512 \times 512$ and $3 \times$ at $2048 \times 2048$ compared to OSTL. ITL reaches a loss that is invariant to the resolution: the validation loss remains constant at around $1 \times 10^{-2}$ for every target discretization. Because the transfer dataset at each resolution is $15 \times$ smaller than the training dataset, we still learn incrementally using far less data. Most importantly, ITL achieves similar accuracy to the baseline model or oracle trained using a dataset as large as the pre-training dataset for CM at $256 \times 256$ and $512 \times 512$. This result is particularly notable because ITL reaches oracle-level accuracy without the need for exhaustive training with large input sizes and large-scale high-resolution datasets. We present a more detailed performance comparison against the oracle in Section V-C.

### B. Performance Analysis

We now study SURFNet’s performance in super-resolution of turbulent flows. Recall that SURFNet’s pre-training consists of training the base network with a large number of inputs of low-resolution data. The transfer phase adds a minimum amount of high-resolution data for fine-tuning the network. Therefore, in addition to performance, we also evaluate the ability of SURFNet in recovering the *same* high-resolution turbulent flow solution as the physics solver.

We test SURFNet in simulations of turbulent flow around 4 unseen-in-training geometries at 2 flow configurations each, for a total of 8 test cases at all target resolutions. Table III presents the comparison against the OpenFOAM physics solver. SURFNet creates three models – (i) low-resolution coarse model, (ii) OSTL model, and (iii) ITL model. Therefore, at every target resolution, we evaluate the time-to-convergence (TTC) using each model, namely: C-SURFNet, O-SURFNet, and I-SURFNet. We compare the TTC of each one of these models with the TTC of the physics solver (PS) and baseline model (BM).

The TTC of the PS is computed by using in tandem two popular convergence criteria in the CFD literature [43]: (1) the residual of each flow variable drops 4 to 6 orders of magnitude and (2) by monitoring when physical quantities reach steady-state. In contrast, SURFNet reaches convergence in three stages. First there is *warmup* (W), where we start the simulation with the PS and let the residual drop between one and two orders of magnitude for each variable. This is sufficient for the fluid parameters close to the physical boundaries to capture the geometry of the new problem. Next there is *inference* (I), where we use these intermediate flow variables as input to the CNN, which infers the steady-state. Finally, the CNN’s output is constrained with the PS in *refinement* to reach the same convergence criteria as the PS [23]. The TTC of SURFNet is the sum of the time spent in the three stages. The three stages are run in parallel on the CPU described in Section IV-C for a fair comparison against OpenFOAM’s solver (which doesn’t support GPU acceleration). Table IV presents the time spent at W and I and the number of iterations in W. The refinement time can be found by adding W and I and substracting this sum from the TTC of Table III.

**C-SURFNet vs. physics solver.** The first column in Table III presents the eight cases in our test dataset. The results of C-SURFNet at $64 \times 256$ indicate good generalization capacity at the lowest resolution to *unseen* geometries. We observe consistent speedups around $2 – 2.1 \times$, independent of the geometry or flow configuration. On the other hand, the first row shows the results for the first test case, flow around a

\(^{\text{A comparison is presented against the BM (or oracle) for only } 256 \times 256 \text{ and } 512 \times 512 \text{ due to the computational cost of collecting large datasets and training at higher resolutions.}}\)
Table III: Summary of the performance results. TTC is the time-to-convergence and ITC is the number of iterations-to-convergence of the physics solver (PS), C-SURFNet (C-SN), O-SURFNet (O-SN), I-SURFNet (I-SN) and the Baseline Model (BM). The speedup of all SURFNet models is calculated with respect to the physics solver.

| Test case | PS | C-SN | O-SN | I-SN | BM | PS | C-SN | O-SN | I-SN | BM |
|-----------|----|------|------|------|----|----|------|------|------|----|
| NACA 1412 | 1.6e-2 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 |
| α = 6° | 0.05 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| NACA 1412 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 |
| θ = 3° | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 |
| NACA 1412 | 0.15 | 0.15 | 0.15 | 0.15 | 0.15 | 0.15 | 0.15 | 0.15 | 0.15 | 0.15 |
| α = 0° | 0.15 | 0.15 | 0.15 | 0.15 | 0.15 | 0.15 | 0.15 | 0.15 | 0.15 | 0.15 |

Table IV: Warmup (W) and inference (I) times (T) and number of iterations (NI) for each test case at each spatial resolution. Times are in accordance with the observations in Section V-A, where OSTL losses increase the more dissimilar the target flow features have sufficient overlap with the input. The performance gains also degrade consistently with increasing resolutions across all test cases. These results show a similar trend across all the test cases. These results are in accordance with the observations in Section V-A where the CM that has been learned from low-resolution inputs is incapable of predicting accurate high-resolution solutions.

**O-SURFNet vs. physics solver.** In Table III all high resolutions have an O-SURFNet (O-SN) column, and we make three main observations from the results. First, O-SURFNet outperforms C-SURFNet for all test cases and resolutions. For instance, for the symmetric NACA0015 at θ = 3° at 2048 × 2048, the TTC of the PS is 88.7 minutes. O-SURFNet reduces this time to 55.4 minutes resulting in a 1.6× speedup (C-SURFNet achieved no performance gain). Figure 7 shows the corresponding qualitative results of O-SURFNet in the task of super-resolution. It resolves the fields of all fluid variables - velocity, pressure, and the modified eddy viscosity - at 2048 × 2048 faster than the PS while being pre-trained with low-resolution data with only fine-tuning at the highest spatial resolution. Second, O-SURFNet’s performance is better at lower resolutions than higher resolutions. For the same NACA0015 test case, at 1024 × 1024, 512 × 512, and 256 × 256, the performance gains are 1.7×, 1.8×, and 2.1× respectively. We observe a similar trend of decaying performance with increasing resolutions across all test cases similar to the coarse model, albeit not to the same extent. This is in line with the results presented in Section V-A where OSTL losses increase the more dissimilar the target flow field is from the low-resolution flow. OSTL at 256 × 256 discretization achieves a 2× speedup irrespective of the test case, demonstrating its potential to generalize to higher resolutions provided the target flow features have sufficient overlap.
with the tiny resolution used for pre-training the CM from which it fine-tunes. However, this is also an indication that OSTL is incapable of achieving resolution-invariance. Third, the speedups achieved by O-SURFNet remain constant and stable among test cases (for instance, we observe speedups around $1.8 \times$ at $512 \times 512$), indicating that fine-tuning the model did not result in overfitting to the transfer geometry. Neither CM nor OSTL yield overfitted models and exhibit stable generalization capacities.

### I-SURFNet vs. physics solver

O-SURFNet’s accuracy decreases as the resolution increases. An ideal solution is one that maintains accuracy (i.e., resolution-invariant) while simultaneously requiring limited computational resources. In Table III we observe that across the board (i.e., all test geometries and flow configurations unseen during pre-training and TL), I-SURFNet achieves a $2 - 2.1 \times$ speedup against the PS. Not only does I-SURFNet maintain the performance gain over PS across the different test cases but also across all target resolutions demonstrating both generalization and resolution-invariance. Since ITL incrementally fine-tunes the model, it requires more data than OSTL. However, fine-tuning is done on the same geometry across resolutions with $15 \times$ lesser data, thereby significantly reducing the overall data collection at high spatial discretizations (compared to prior approaches that require considerably more high-resolution data). Figure 8 shows the qualitative results of I-SURFNet’s flow solution around the nose of the non-symmetric NACA1412 airfoil at $\theta = 5^\circ$. I-SURFNet successfully recovers high-resolution turbulent flow simulations on a geometry with at least three distinct features (i.e., flat trailing edge, non-symmetry, and different chamber thickness) not present in the training or transfer datasets. This validates that SURFNet pre-trained with low-resolution data with only fine-tuning can generalize to unseen geometries. The largest target resolution studied (i.e., $2048 \times 2048$) is $256 \times$ the size of the tiny discretization (i.e., $64 \times 256$) used in pre-training the CM to stress SURFNet’s ability in super-resolution. It is guaranteed to converge to a unique solution as long as the problem is well-posed [47].

![Figure 7: Velocity in m s$^{-1}$ (top), kinematic pressure in m$^2$/s$^2$ (middle), and modified eddy viscosity in m$^2$/s (bottom) around the NACA 0015 airfoil at $\theta = 3^\circ$, $Re = 6 \times 10^5$. Comparison between the low-resolution ($64 \times 256$) training data to train the coarse model and the ground truth OpenFOAM’s solution at $2048 \times 2048$.](image)

![Figure 8: Detail of the velocity in m s$^{-1}$ (top), kinematic pressure in m$^2$/s$^2$ (middle), and modified eddy viscosity in m$^2$/s (bottom) at the nose of the nonsymmetric NACA 1412 airfoil at $\theta = 5^\circ$, $Re = 6 \times 10^5$. Comparison between the low-resolution ($64 \times 256$) training data to train the coarse model (left), I-SURFNet’s output after the refinement phase at $2048 \times 2048$ (middle), and the ground truth OpenFOAM’s solution at $2048 \times 2048$ (right).](image)

**C. SURFNet vs. Oracle**

We now compare SURFNet with the oracle or baseline model (BM) at $256 \times 256$ and $512 \times 512$ target discretizations. We define BM as conducting full-scale data collection and training at the higher resolutions. Specifically, a dataset as large as the training dataset outlined in Table III collected at the tiny resolution ($64 \times 256$) for pre-training the CM is collected for training the CNN at the two target discretizations. The CNN is trained using a learning rate of $1 \times 10^{-4}$ and a batch size of 32 and 16, respectively. Figure 6 and Table III show that the loss on the validation dataset and speedup compared to the OpenFOAM physics solver achieved by ITL is similar to that of the BM. To understand how much time SURFNet
saves with respect to the oracle, Table V compares the data collection and the training time of SURFNet with that of BM. SURFNet’s data collection time is the sum of the time spent collecting low-resolution training data and the transfer datasets at higher resolutions. Similarly, the total training time includes pre-training the CM and the time spent fine-tuning at the target higher resolution.

In terms of data collection time, SURFNet takes 0.4 hrs (0.33 + 0.06) vs. 1 hr for BM at 256 × 256. At 512 × 512, SURFNet’s data collection time is 0.8 hrs (0.33 + 0.06 + 0.41) vs. 3.86 hrs for BM. Training at 256 × 256 and 512 × 512 takes 2.56 hrs and 3 hrs for SURFNet vs. 9.25 hrs and 38 hrs for BM, respectively. Overall, SURFNet reduces the combined data collection and training time by 3.6× and 10.2×, respectively, while achieving similar performance gain and accuracy as BM. Note that the computational advantage of SURFNet increases with increasing resolution size. Moreover, these results highlight the impracticality of performing exhaustive data collection and training at 1024 × 1024 and beyond, underscoring the impact and potential of TL (specifically ITL) in enabling super-resolution of complex turbulent flows.

Table V: Comparison with the baseline model (BM) on data collection and training for reaching similar accuracy at 256 × 256 and 512 × 512 spatial resolutions.

|             | SURFNet | BM          |
|-------------|---------|-------------|
|             | 256 × 256| 512 × 512   |
|             | 256 × 256| 512 × 512   |
| Data collection time for training | 0.33    | 0.33        |
| Data collection time for TL       | 0.06    | 0.41        |
| Training time                        | 2.5   | 2.5         |
| TL time                              | 0.06   | 0.5         |

VI. RELATED WORK

DL for CFD. Several recent approaches aim to find DL-based accelerators for turbulent flows with promising results. Maulik et al. predict the eddy viscosity field, but not other flow properties such as the velocity. Thuerey et al. use an encoder-decoder type of network but their approach does not account for the eddy viscosity field. Although they present real-time solutions, the geometry in the training and prediction stages are same (i.e., airfoils) making the solution less generalizable. Alternatively, Obiols-Sales et al. feed the network’s prediction back into the physics solver to enable generalization with the same model without relaxing the convergence constraints. These efforts are not resolution-invariant unless the network is trained with large-scale data from a variety of high-resolution simulations. Tompson et al. accelerate Eulerian fluid simulations by minimizing the divergence of the velocity field using an unsupervised method. This approach maintains accuracy up to 1024 × 1024 resolution. However, Eulerian fluid simulations ignore the viscous effects that are critical for most engineering systems of interest. SURFNet predicts all relevant fluid variables in the entire domain of turbulent flows to accelerate CFD simulations.

Mesh-independent DL approaches. Raissi et al. introduced physics-informed neural networks (PINNs) - networks trained to respect any physics laws. These methods substitute traditional solvers. However, this approach has several constraints. First, for complex turbulent flow problems, including the conservation laws in the loss function may lead to stiffer optimizations. Second, training a new model is required for every new instance of a distinct flow configuration (unless the initial/boundary conditions are an input to the network), instead of a simple forward pass of the network, severely restricting its generalization capabilities. Lu et al. introduced an infinite-dimensional operator with neural networks, known as neural operators (NO), that learns the nonlinear operation from partial differential equations without knowledge of the underlying PDE — only with data. NO provides a single set of network parameters that are compatible with different discretizations. Hence, they are resolution-invariant. However, these approaches train the network with data downsampled from high-resolution simulations — which is impractical for many practitioners and suffers from the same computational constraints of traditional solvers. Jiang et al. introduced a CNN-based resolution-invariant approach that satisfies the underlying PDE. However, it also suffers from the same data-collection limitation as the former approaches. SURFNet overcomes the above limitations by training the CNN models primarily from data gathered at low-resolutions while enabling super-resolution by only minimal fine-tuning.

VII. CONCLUSIONS

This paper presented SURFNet, a super-resolution flow network that accelerates high-resolution turbulent CFD simulations. SURFNet is primarily trained on low-resolution simulation data and applies this information (via transfer learning) to high-resolution inputs. We proposed two variations, one-shot transfer learning (OSTL) and incremental transfer learning (ITL). Both approaches yield consistent speedups across test geometries unseen during the training or transfer stages and exhibit good generalization capacities. We demonstrated resolution-invariance with ITL on domains up to 256 × larger than the tiny discretization used in training and a uniform 2 × 2.1 speedup across target resolutions and test geometries compared to the OpenFOAM CFD solver. SURFNet is able to recover high-resolution flow features with 15 × less data at high resolutions and reducing the combined data collection and training time by 3.6× and 10.2× at 256 × 256 and 512 × 512 grid sizes, respectively.

Future work includes expanding SURFNet to a wide range of flow and boundary conditions. As SURFNet is computationally inexpensive, and data from low-resolution CFD simulations are widely available, further explorations of the generalization capabilities are warranted. Since SURFNet’s underlying CNN architecture is unconstrained, SURFNet could be beneficial for transfer learning across physical domains, such as molecular dynamics or solid mechanics.
[29] P. Spalart and S. Allmaras, “A one-equation turbulence model for aerodynamic flows,” in 30th aerospace sciences meeting and exhibit, 1992, p. 439.

[30] C. Duru, H. Alemdar, and Ö. U. Baran, “Cnnfoil: convolutional encoder decoder modeling for pressure fields around airfoils,” Neural Computing and Applications, pp. 1–15, 2020.

[31] W. Peng, Y. Zhang, and M. Desmarais, “Spatial convolution neural network for efficient prediction of aerodynamic coefficients,” in AIAA Scitech 2021 Forum, 2021, p. 0277.

[32] 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, vol. 378, pp. 686–707, 2019.

[33] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, “Deepxde: A deep learning library for solving differential equations,” arXiv preprint arXiv:1907.04502, 2019.

[34] E. Haghighat and R. Juanes, “Sciann: A keras wrapper for scientific computations and physics-informed deep learning using artificial neural networks,” arXiv preprint arXiv:2005.08803, 2020.

[35] X. Jin, S. Cai, H. Li, and G. E. Karniadakis, “Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations,” Journal of Computational Physics, vol. 426, p. 109951, 2021.

[36] R. Maulik, H. Sharma, S. Patel, B. Lusch, and E. Jennings, “Accelerating rans turbulence modeling using potential flow and machine learning,” arXiv preprint arXiv:1910.10878, 2019.

[37] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, “A survey on deep transfer learning,” in International conference on artificial neural networks. Springer, 2018, pp. 270–279.

[38] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, “A comprehensive survey on transfer learning,” Proceedings of the IEEE, vol. 109, no. 1, pp. 43–76, 2020.

[39] C. Cai, S. Wang, Y. Xu, W. Zhang, K. Tang, Q. Ouyang, L. Lai, and J. Pei, “Transfer learning for drug discovery,” Journal of Medicinal Chemistry, vol. 63, no. 16, pp. 8683–8694, 2020.

[40] Y. Pathak, P. K. Shukla, A. Tiwari, S. Stalin, and S. Singh, “Deep transfer learning based classification model for covid-19 disease,” Irbm, 2020.

[41] S. Ruder, “Neural transfer learning for natural language processing.” Ph.D. dissertation, NUI Galway, 2019.

[42] S. Shao, S. McAleer, R. Yan, and P. Baldi, “Highly accurate machine fault diagnosis using deep transfer learning,” IEEE Transactions on Industrial Informatics, vol. 15, no. 4, pp. 2446–2455, 2018.

[43] M. Long, H. Zhu, J. Wang, and M. I. Jordan, “Deep transfer learning with joint adaptation networks,” in International conference on machine learning. PMLR, 2017, pp. 2208–2217.

[44] J. Ling, A. Kurzawski, and J. Templeton, “Reynolds averaged turbulence modelling using deep neural networks with embedded invariance,” Journal of Fluid Mechanics, vol. 807, pp. 155–166, 2016.

[45] A. Gulli and S. Pal, Deep learning with Keras. Packt Publishing Ltd, 2017.

[46] “EarlyStopping keras callback,” https://keras.io/api/callbacks/early_stopping/.

[47] E. Ott, Chaos in dynamical systems. Cambridge university press, 2002.

[48] J. Tompson, K. Schlachter, P. Sprechmann, and K. Perlin, “Accelerating eulerian fluid simulation with convolutional networks,” pp. 3424–3433, 2017.

[49] S. Wang, Y. Teng, and P. Perdikaris, “Understanding and mitigating gradient pathologies in physics-informed neural networks,” arXiv preprint arXiv:2001.04536, 2020.

[50] H. Wang, R. Planas, A. Chandramowlishwaran, and R. Bostanabad, “Train once and use forever: Solving boundary value problems in unseen domains with pre-trained deep learning models,” arXiv preprint arXiv:2104.10873, 2021.

[51] L. Lu, P. Jin, and G. E. Karniadakis, “Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators,” arXiv preprint arXiv:1910.03193, 2019.