SURFNet: Super-resolution of Turbulent Flows with Transfer Learning using Small Datasets

***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 learns* 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$\times$ 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$\times$ larger than the coarse model. On four test geometries and eight flow configurations unseen during training, we observe a consistent $2 - 2.1\times$ 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 $3.6\times$ and $10.2\times$, 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.

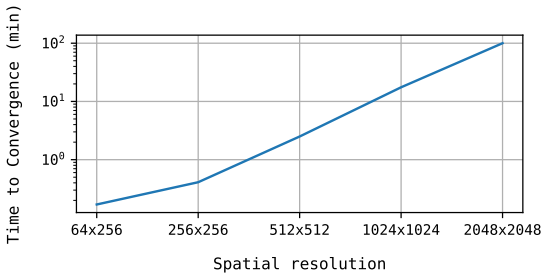

Figure 1: Time to convergence for flow around a NACA4415 airfoil at different spatial resolutions on a dual-socket Intel Xeon Gold 6148 CPU (total 40 cores).

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 I summarizes the state-of-the-art approaches for accelerating CFD simulations using DL and categorizes them across several features. First, the majority of Convolutional Neural Network (CNN) based approaches are finite-dimensional maps and hence lack resolution-invariance [21, 22, 23, 25, 26]. Mesh-free, resolution-invariant DL methods are a potential alternative to query fine-scale flow physics [14, 16, 17]. However, these approaches use low-resolution data downsampled from high-resolution solutions to train the network. As a result, current mesh-free methods suffer from the same limitation as classical CNN-based approaches that require a large number of computationally expensive simulations to collect training data at high resolutions. Practical CFD simulations require high spatial resolutions (such as 1024×1024) according to NASA's FUN3D and CFL3D solvers [27] making current DL approaches computationally prohibitive (see Figure 1). Second, most approaches [13, 14, 15, 16, 17, 21] lack generalization

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 [16, 17, 22, 24], only test on geometry domains that were part of the training phase [13, 14, 21], replace partly [24] or entirely [13, 14, 21, 22] traditional solvers with a neural network surrogate not meeting convergence constraints, and most importantly, train with data downsampled from high-resolution simulations [13, 14, 16, 17]. *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 [14]	Darcy	🔴	🔴	🟢	🔴	421×421	RE	1×10^{-2}	NO
Fourier Neural Operator [14]	Navier-Stokes	🔴	🔴	🟢	🔴	256×256	RE	8.6×10^{-3}	NO
Graph Kernel Network [16]	Darcy	🔴	🔴	🟢	🔴	241×241	Relative L_2	3.7×10^{-2}	NO
Bhattacharya et al. [17]	Darcy	🔴	🔴	🟢	🔴	421×421	RE	2×10^{-3}	NO
MeshFreeFlowNet [13]	Rayleigh-Bénard	🔴	🔴	🟢	🔴	4×512	NMAE	3.3×10^{-3}	CNN
Gao et al. [15]	Laminar	🔴	🔴	🟢	🔴	200×200	RE	0.025	CNN
TFNet [21]	Rayleigh-Bénard	🔴	🔴	🔴	🔴	1792×256	RMSE	200	CNN
Guo et al. [22]	Laminar	🟢	🔴	🔴	🟢	128×256	RME	1.76%	CNN
CFDNet [23]	Navier-Stokes	🟢	🟢	🔴	🟢	64×256	RME	0%	CNN
Smart-fluidnet [24]	Eulerian	🟢	🟡	🟢	🟡	1024×1024	MAE	9×10^{-3}	CNN
SURFNet (this paper)	Navier-Stokes	🟢	🟢	🟢	🟢	2048×2048	RME	0%	TL

to unseen geometries where the test geometry is a subset of the training data – a key limitation for CFD researchers/practitioners. Third, only a limited number of approaches support super-resolution of turbulent flows that are significantly more challenging than laminar flows [15] and ubiquitous with a wide range of applications in aerodynamics, atmospheric science, turbomachinery, and propulsion [13, 23, 25, 26]. Lastly, it's common to use DL algorithms as a pure surrogate model that entirely replaces the CFD solver, thereby not providing the same convergence guarantees as the latter. There is a pressing need to design DL based models for super-resolution of turbulent flows that (a) eliminates the need for extensive data collection at high resolutions, (b) provides discretization and resolution-independent acceleration, and (c) can generalize to unseen geometries and flow configurations while meeting the convergence guarantees of the traditional physics solvers.

To address the above need, we develop **SURFNet** (**SU**per-**R**esolution **F**low **Net**work), 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$\times$ 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 256×256 and 512×512 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:

$$\frac{\partial \bar{U}_i}{\partial x_i} = 0 \tag{1}$$

$$\bar{U}_j \frac{\partial \bar{U}_i}{\partial x_j} = \frac{\partial}{\partial x_j}\left[-(\bar{p})\,\delta_{ij} + (\nu + \nu_t)\left(\frac{\partial \bar{U}_i}{\partial x_j} + \frac{\partial \bar{U}_j}{\partial x_i}\right)\right] \tag{2}$$

where $\bar{U}$ is the mean velocity, $\bar{p}$ is the kinematic mean pressure, ν is the fluid viscosity, and ν_t is the eddy viscosity.

The RANS equations provide a time-averaged solution to the incompressible Navier-Stokes equations at the expense of yielding a non-closed equation. Closing the equation is usually done through Boussinesq's approximation [28], which yields the eddy viscosity. The eddy viscosity is found through turbulence modeling. The Spalart-Allmaras one-equation model shown below provides a transport equation to compute the modified eddy viscosity, $\tilde{\nu}$.

$$\bar{U}_i \frac{\partial \tilde{\nu}}{\partial x_i} = C_{b1}\left(1 - f_{t2}\right)\tilde{S}\tilde{\nu} - \left[C_{w1} f_w - \frac{C_{b1}}{\kappa^2} f_{t2}\right]\left(\frac{\tilde{\nu}}{d}^2\right) + \frac{1}{\sigma}\left[\frac{\partial}{\partial x_i}\left(\left(\nu + \tilde{\nu}\right)\frac{\partial \tilde{\nu}}{\partial x_i}\right) + C_{b2}\frac{\partial \tilde{\nu}}{\partial x_j}\frac{\partial \tilde{\nu}}{\partial x_j}\right] \quad (3)$$

Then, we can compute the eddy viscosity from $\tilde{\nu}$ as $\nu_t = \tilde{\nu} f_{v1}$. These equations represent the most popular implementation of the Spalart-Allmaras model. The terms f_{v1}, $\tilde{S}$, and f_{t2} are specific to the model. They contain, for example, first order flow features (magnitude of the vorticity). C_{b1}, C_{w1}, C_{b2}, κ, and σ are constants also specific to the model, found experimentally. The first original reference of the model details the equations and the values of these constants [29].

III. SURFNet: Transfer Learning for Super-Resolution Flow Simulations

Our objective is to accelerate the convergence of high-resolution turbulent flow simulations. We achieve this by reconstructing fine-grid flow solutions from coarse grid models referred to as *super-resolution* with minimum data collection at higher resolutions.

In this section, we first describe the strategy used to accelerate the convergence of the equations described in Section II with DL. Then, we describe the design of the CNN architecture to train the *coarse model* (CM) with low-resolution data. Finally, we present SURFNet, a novel transfer learning-based super-resolution approach that relaxes the demand for generating expensive and time-consuming training data to train accurate models for discretizations 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 $x^N = G\left(x^I; \theta_{wl}\right)$ 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 first velocity component (U), the second velocity component (V), the pressure (P), and the modified eddy viscosity (ν). 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, 25, 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, 36] 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. We adhere to the latter approach in this paper. This design 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

Figure 2 illustrates the design of the CNN architecture and its input-output representation for learning the coarse model. The CNN is an eight-layer, symmetric, convolutional-deconvolutional neural network, consisting of 4 convolution layers followed by 4 deconvolutions (or transposed convolution) layers. Each layer has a filter of size 5×5, with a stride of 1, and uses the LeakyReLU activation function with a slope $s = 0.25$. The number of filters of the first four convolution layers is 16, 32, 128, and 256, respectively. The number of filters of the last four deconvolution layers is the same but reversed in decreasing order.

We aim to learn a generalizable model that can predict flow around rotated geometries (i.e., varying pitch angle, θ) and changing flow fields (i.e., angle of attack, α) 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, all layers use a stride of 1 with the 5×5 filter. This overlap allows the filter of each convolution layer to cover all the regions of the flow field present in the input images. All layers use the LeakyReLU activation function to capture both positive and negative values in the input, output images, and intermediate activations.

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

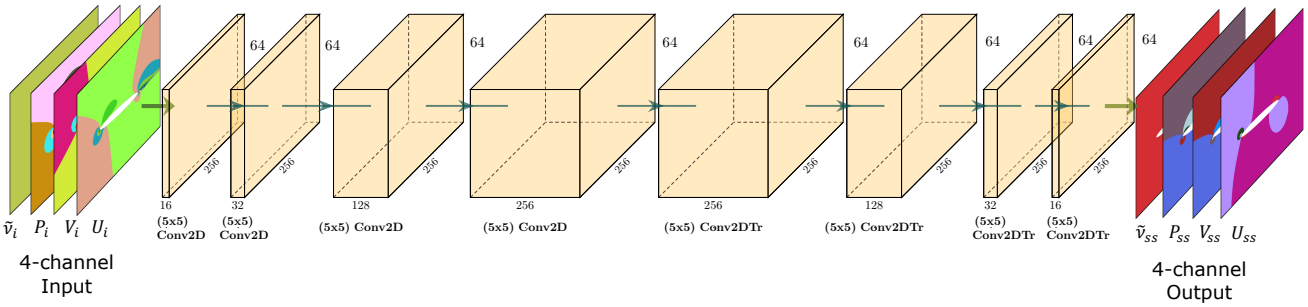

Figure 2: CNN and its input-output representation. The CNN is an eight-layer, symmetric, convolutional-deconvolutional neural network. Each layer has a 5×5 filter, a stride of 1, and the LeakyReLU activation function. The input and the output are a 4-channel tensor image: each channel represents one flow variable (first velocity component U, second velocity component V, pressure P and modified eddy viscosity $\tilde{\nu}$). 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).

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. It is an extended practice for CFD practitioners to start the fine-grid simulation of a specific flow problem with its coarse-grid solution because a large majority of flow structures are already present [2]. For instance, coarse grids can both capture and resolve simple flow regions, such as areas with small variable gradients (e.g., the flow in the freestream). However, they can only capture (not resolve) complex structures such as the wake region behind solid bodies after flow separation or the boundary layer (with the appropriate treatment [43]). 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 discretizations. Since coarse grids can capture complex flow structures but fail to resolve them accurately [2, 3, 44], we need to append the previously unresolved flow information. These include an accurate profile of the flow variables in regions of strong gradients or the recirculation structures in the wake region. An advantage of our super-resolution TL methodology is that the input is *truly* from coarse-grid simulations, not downsampled 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 $r_f : x_f \times y_f \mid f = 1, ..., t$ 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 2048×2048, we transfer learn from 64×256 to 2048×2048 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 (i.e., r_c->r_1->r_2...->r_t). For example, if we desire to recover high-resolution flow fields at 1024×1024, we transfer learn from 64×256 to 256×256, then from 256×256 to 512×512, and finally from 512×512 to 1024×1024. 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

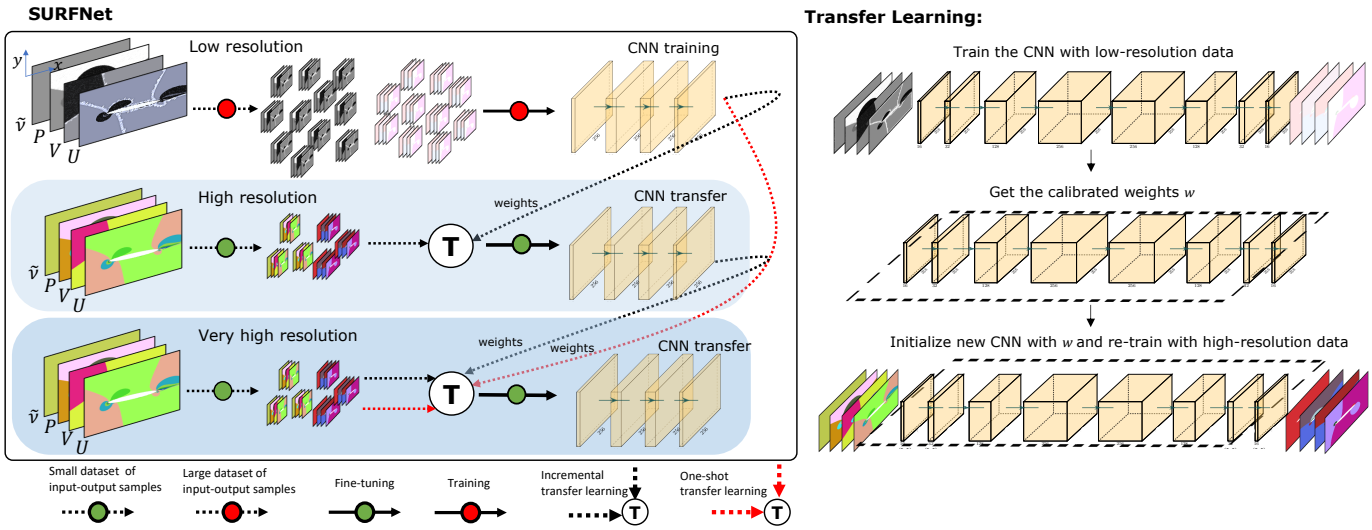

Figure 3: SURFNet (left) and its inductive transfer learning (right) for super-resolution of turbulent flows. A large dataset is collected at low-resolutions to train the CNN and obtain the coarse model. Small data is collected at higher resolutions and SURFNet transfers the model weights from the coarse model to train the fine-scale model using either one-shot (red) or incremental transfer learning (black).

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 V, 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 network-based or 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) [45]. 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 α and θ). 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 6×10^5 using the RANS equations and the Spalart-Allmaras turbulence model presented in Section II.

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, the composition of the transfer dataset is identical. The validation and test datasets are also kept constant across all resolutions (from low to high). 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	10	•
	NoFC	90	
	Type	ellipses	
Transfer	NoG	•	1
	NoFC		6
	Type		ellipse $AR = 0.1$
Validation	NoG	3	
	NoFC	9	
	Type	NACA0012, ellipse $AR = 0.22$, cylinder	
Test	NoG	4	
	NoFC	8	
	Type	NACA1412, NACA0015, ellipse $AR = 0.3$, cylinder	

Training dataset. First, we collect a large-scale, low-resolution dataset to train the coarse model (CM). The discretization size of the training dataset is 64×256. 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, α, and four different pitch angles, θ, chosen randomly for each ellipse in the range between $-2-6^\circ$ for a total of 90 flow configurations. The angle of attack is an important variable in determining the magnitude of the force of lift. The θ 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. The α 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°. These configurations are illustrated in Figure 4.

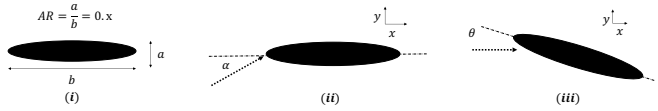

Figure 4: Geometry configurations used in training. The dotted arrow shows the direction of the incoming flow and the dashed line shows the chord of the solid body. Sketch of the (i) ellipse aspect ratio, (ii) angle of attack α, and (iii) pitch angle θ.

Transfer dataset. Next, we collect a small-scale, high-resolution dataset. The transfer dataset consists of flow around *one* unique geometry, ellipse $AR = 0.1$. For this geometry, we consider 6 flow variants: three α angles and three θ angles, chosen randomly as in the training dataset. Note that we collect a transfer dataset at each target resolution: 256×256, 512×512, 1024×1024, and 2048×2048. Each of the four transfer datasets consists of the same flow configurations – the only difference between them is the grid resolution. This choice is notable for two reasons. First, it stresses the extreme case of whether a single *and* identical geometry is sufficient to recover high-resolution flow fields at any target resolution, both with OSTL and ITL. Second, each transfer dataset has 6 flow configurations, compared to 90 in the training dataset. Therefore, the former has $15\times$ fewer flow configurations than the latter, dramatically reducing the overall data collection at fine-scale discretizations.

Validation dataset. We collect validation datasets to control the under- and over-fitting of the network during both the pre-training and fine-tuning phases. The validation dataset consists of flow around three different unseen-in-training geometries: an ellipse $AR = 0.22$, a symmetric NACA0012 airfoil, and a cylinder. For each geometry, we consider 3 flow variations – two α angles and one θ angle, chosen randomly as in the training dataset for a total of nine flow configurations. Note that we collect the same validation dataset at all resolutions, from 64×256 to 2048×2048, the only difference being the grid discretization as in the transfer datasets.

Test dataset. We create test datasets 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 α angle and one θ angle. Table III summarizes the different test cases. Note that we also collect the test dataset at all resolutions for a total of five test datasets.

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 between 1×10^{-5} and 1×10^{-6} for the velocity and pressure and 1×10^{-4} for the modified eddy viscosity is the criteria to consider the training simulations converged. We adhere to tolerances that are extended practice in CFD simulations [46]. We use the `GAMG` solver for computing the pressure at every iteration, with a tolerance of 1×10^{-8} and the `GaussSeidel` smoother from OpenFOAM. The `smoothSolver` and the `GaussSeidel` smoother compute both the velocity and modified eddy viscosity with the same tolerances as the pressure.

Architecture and Libraries. All the OpenFOAM simulations are run in parallel on a dual-socket Intel Xeon Gold 6148. 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. We implement the CNN using Keras [47] 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 1×10^{-4} with no decay for all training. The training is stopped using the EarlyStopping Keras callback [48] by monitoring the validation loss with patience of 6 epochs. After 41 epochs, the training loss reaches 7.5×10^{-4} and validation loss reaches a value of 8.3×10^{-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 1.5×10^{-3} at 256×256 to 2.1×10^{-1} at 2048×2048. 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.

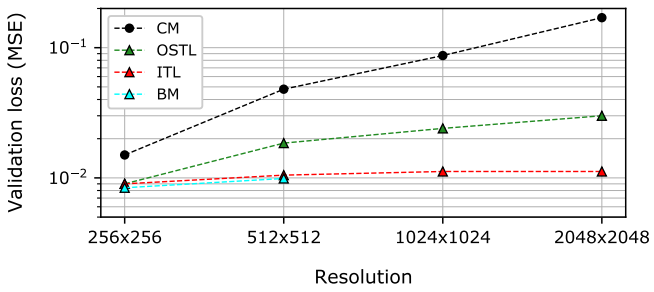

Figure 6: Coarse model (CM), one-shot transfer learning (OSTL), incremental transfer learning (ITL), and baseline model (BM) losses on the validation dataset at every target resolution.

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 1×10^{-4} to 1×10^{-5}), and train

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 256×256, OSTL reduces the validation loss from 1.5×10^{-2} for CM to 9×10^{-3}, resulting in 60% overall reduction. We observe a similar trend with subtle improvements at larger resolutions. At 2048×2048 the validation loss reduces from 2×10^{-1} for CM to 3×10^{-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 flow with 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 (2048×2048, for instance), the new flow field seen during the transfer phase produces sizable changes to the weights of the network compared to transferring to "mid-resolutions" (256×256, for instance). 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 approach is counter-productive, as it 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, which we avoid explicitly. 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 256×256. At 256×256, since we transfer from 64×256 with no intermediate resolution between the two, the ITL loss is the same as the OSTL loss as expected. However, at higher resolutions, ITL improves the generalizability of the network. The validation loss drops by half at 512×512 and $3\times$ at 2048×2048 compared to OSTL. ITL reaches a loss that is invariant to the resolution: the validation loss remains constant at around 1×10^{-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 256×256 and 512×512. 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 the performance of the two TL approaches 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)[1].

The TTC of the PS is computed by using in tandem two popular convergence criteria in the CFD literature [46]: (1) the residual of each flow variable drops 4 to 6 orders of magnitude (4-5 for the eddy viscosity and 5-6 for the pressure and velocity) and (2) the PS reaches the steady-state solution by monitoring steady-state physical quantities. In contrast, SURFNet reaches convergence in three stages. First, we start the simulation with the PS and let the residual drop one order 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, we use these intermediate flow variables to generate the input image described in Section III-B. We then use the proposed CNN models to infer the steady-state from this input tensor. Finally, the CNN's output is constrained with the physics solver 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).

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 64×256 indicate good generalization capacity at

[1]A comparison is presented against the BM (or oracle) for only 256×256 and 512×512 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.

		64×256		256×256				512×512					1024×1024				2048×2048			
Test case		PS	C-SN	PS	C-SN	O-SN	BM	PS	C-SN	O-SN	I-SN	BM	PS	C-SN	O-SN	I-SN	PS	C-SN	O-SN	I-SN
NACA 1412 @ $\theta = 5^\circ$	TTC (min)	0.16	0.08	0.4	0.22	0.2	0.2	2.4	1.7	1.3	1.2	1.2	17.2	14.3	10.1	8.6	95	95	59.4	47.5
	ITC	1865	825	3636	1874	1672	1669	7273	5018	3864	3460	3473	10118	8259	5779	4887	12338	12183	7556	6014
	Speedup	1×	**2.1**×	1×	**1.8**×	**2**×	**2**×	1×	**1.4**×	1.8×	**2**×	**2**×	1×	**1.2**×	1.7×	**2**×	1×	**1**×	1.6×	**2**×
NACA 1412 @ $\alpha = 6^\circ$	TTC (min)	0.19	0.09	0.53	0.29	0.27	0.25	3.3	2.5	1.9	1.7	1.7	19.3	16.1	12.06	9.7	98.5	89.5	61.6	49.3
	ITC	2215	991	4818	2531	2263	2199	10000	7516	5706	4823	4830	11353	9289	6506	5504	12792	11474	7840	6241
	Speedup	1×	**2.1**×	1×	**1.8**×	**2**×	**2.1**×	1×	**1.3**×	1.7×	**2**×	**2**×	1×	**1.2**×	1.6×	**2**×	1×	**1.1**×	1.5×	**2**×
NACA 0015 @ $\theta = 3^\circ$	TTC (min)	0.15	0.07	0.34	0.18	0.16	0.17	2.5	1.7	1.4	1.2	1.2	15.8	12.2	8.8	7.9	88.7	88.7	55.4	44.4
	ITC	1748	769	3091	1481	1326	1349	7576	4874	4032	3431	3424	9294	6977	4991	4475	11519	11365	7045	5605
	Speedup	1×	**2.1**×	1×	**1.9**×	**2.1**×	**2**×	1×	**1.5**×	1.8×	**2.1**×	**2.1**×	1×	**1.3**×	1.7×	**2**×	1×	**1**×	1.6×	**2**×
NACA 0015 @ $\theta = \alpha = 0^\circ$	TTC (min)	0.13	0.06	0.32	0.18	0.16	0.16	2.3	1.6	1.3	1.2	1.2	14.5	10.4	8.5	7.3	82.4	82.4	51.5	41.2
	ITC	1515	626	2909	1470	1308	1311	6970	4802	3696	3308	3300	8529	5920	4845	4093	10701	10546	6533	5196
	Speedup	1×	**2.2**×	1×	**1.8**×	**2**×	**2**×	1×	**1.4**×	1.8×	**2**×	**2**×	1×	**1.2**×	1.7×	**2**×	1×	**1**×	1.6×	**2**×
Ellipse $AR = 0.3$ @ $\theta = 5^\circ$	TTC (min)	0.22	0.1	0.61	0.36	0.31	0.3	4.1	3.2	2.4	2.1	2.1	20.8	17.3	13.0	10.4	107.2	97.5	67.0	53.6
	ITC	2564	1158	5545	3116	2627	2615	12424	9381	7132	6036	6039	12235	10024	7475	5946	13922	12502	8546	6806
	Speedup	1×	**2.1**×	1×	**1.8**×	**2**×	**2**×	1×	**1.3**×	1.7×	**2**×	**2**×	1×	**1.2**×	1.6×	**2**×	1×	**1**×	1.5×	**2**×
Ellipse $AR = 0.3$ @ $\alpha = 1^\circ$	TTC (min)	0.25	0.13	0.64	0.36	0.32	0.33	4.6	3.3	2.6	2.3	2.3	22.3	17.2	13.1	11.2	114.7	104.3	71.7	57.4
	ITC	2914	1394	5818	3086	2763	2777	13939	9780	7568	6793	6790	13118	9918	7544	6387	14896	13387	9155	7293
	Speedup	1×	**2**×	1×	**1.8**×	**2**×	**2**×	1×	**1.4**×	1.8×	**2**×	**2**×	1×	**1.3**×	1.7×	**2**×	1×	**1.1**×	1.6×	**2**×
Cylinder @ $\theta = 0^\circ$	TTC (min)	0.30	0.19	0.72	0.42	0.36	0.36	5.2	3.7	2.9	2.6	2.5	30.6	25.5	18.0	15.3	165.4	165.4	103.4	82.7
	ITC	4663	2157	6545	3704	3127	3127	15758	11079	8578	7702	7691	18000	14828	10416	8828	21481	21326	13270	10585
	Speedup	1×	**2.1**×	1×	**1.7**×	**2**×	**2**×	1×	**1.4**×	1.8×	**2**×	**2.1**×	1×	**1.2**×	1.7×	**2**×	1×	**1**×	1.6×	**2**×
Cylinder @ $\alpha = 1^\circ$	TTC (min)	0.36	0.16	0.68	0.40	0.34	0.34	5.0	3.6	2.9	2.5	2.5	27.1	22.6	16.9	13.6	153.0	139.1	95.6	76.5
	ITC	4196	1844	6182	3490	2945	2941	15152	10646	8736	7399	7400	15941	13112	9791	7799	19870	17909	12264	9780
	Speedup	1×	**2.2**×	1×	**1.7**×	**2**×	**2**×	1×	**1.4**×	1.7×	**2**×	**2**×	1×	**1.2**×	1.6×	**2**×	1×	**1.1**×	1.6×	**2**×

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 non-symmetric NACA1412 at $\theta = 5^\circ$ at different spatial resolutions. Although we observe significant speedups at low resolutions and C-SURFNet exhibits improved TTC compared to the PS, its performance gain also degrades consistently with increasing discretizations. The speedup drops from $2.1\times$ at 64×256 to $1\times$ at 2048×2048 where at the highest discretization, it's no better than the PS. We observe a similar trend across all the test cases where C-SURFNet's performance drops significantly with increasing resolution. These results are in accordance with the observations presented in Section V-A, where the CM that has learned from only low-resolution inputs is incapable of predicting accurate solutions with high-resolution inputs.

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, the results indicate that O-SURFNet significantly outperforms C-SURFNet for all test cases and resolutions. For instance, for the symmetric NACA0015 at $\theta = 3^\circ$ at 2048×2048, the TTC of the PS is 88.7 minutes. O-SURFNet reduces this time to 55.4 minutes resulting in a $1.6\times$ speedup whereas, 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 $1.7\times$, $1.8\times$, and $2.1\times$ 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 $2\times$ 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, in the OSTL column of 512×512, we observe speedups around $1.8\times$), 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.

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

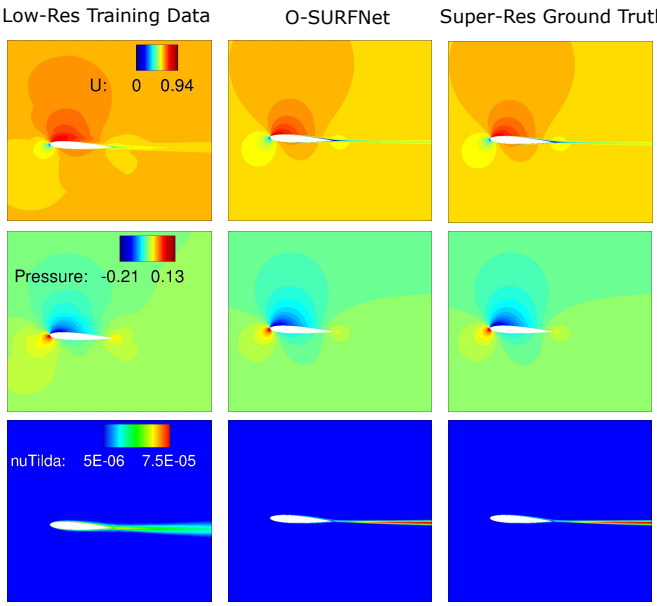

Figure 7: Velocity field in $\mathrm{m\,s^{-1}}$ (top), kinematic pressure field in $\mathrm{m^2/s^2}$ (middle), and modified eddy viscosity field in $\mathrm{m^2/s}$ (bottom) around the NACA 0015 airfoil at $\theta = 3°$, $Re = 6 \times 10^5$. Comparison between the low-resolution (64×256) training data to train the CM (left); O-SURFNet's super-resolution output at 2048×2048 (middle), and the ground truth OpenFOAM's solution at 2048×2048 (right).

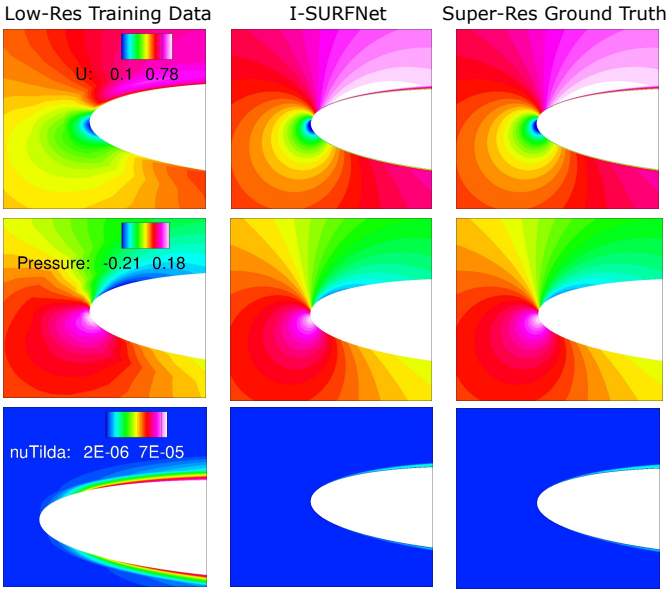

Figure 8: Detail of the velocity field in $\mathrm{m\,s^{-1}}$ (top), kinematic pressure field in $\mathrm{m^2/s^2}$ (middle), and modified eddy viscosity field in $\mathrm{m^2/s}$ (bottom) at the nose of the nonsymmetric NACA 1412 airfoil at $\theta = 5°$, $Re = 6 \times 10^5$. Comparison between the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

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 downsample from high-resolution inputs requiring considerably more data). Figure 8 shows the qualitative results of I-SURFNet's flow solution around the nose of the non-symmetric NACA1412 airfoil at $\theta = 5°$. 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., 2048×2048) is **256**$\times$ the size of the tiny discretization (i.e., 64×256) used in pre-training the CM to stress SURFNet's ability in super-resolution. Moreover, it is guaranteed to converge to a unique solution as long as the problem is well-posed [49].

We additionally explored the effect of the required size of the spatial step size to maintain accuracy for ITL. By incrementally fine-tuning with a step size of 1 up to 2048×2048, the performance gain is consistently $2\times$. This gain is the best possible achievable as observed by the results of the oracle or BM that is fully trained at 256×256 and 512×512 resolutions. By incrementally transferring with a step size of 2 up to 2048×2048, the performance gain drops to $1.7 - 1.8\times$. So, we conclude that to achieve oracle-level accuracy and performance, the ideal step size for ITL is 1.

C. SURFNet vs. Oracle

We now compare SURFNet with the oracle or baseline model (BM) at 256×256 and 512×512 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 II collected at the tiny resolution (64×256) 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 1×10^{-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 IV 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 $3.6\times$ and $10.2\times$, 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 impracticability 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 IV: Comparison of the time (in hours) spent by I-SURFNet and 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	1	3.86
Data collection time for TL	0.06	0.41	-	-
Training time	2.5	2.5	9.25	38
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 [36]. Thuerey et al. use an encoder-decoder type of network but their approach does not account for the eddy viscosity field [25]. 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 [24] maintains accuracy up to 1024×1024 resolution. However, Eulerian fluid simulations ignore the viscous terms in the Navier-Stokes equations – which are critical for several 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 physics laws described by general nonlinear partial differential equations. These methods substitute traditional solvers [32, 33, 34]. However, this approach has several constraints. First, recent works have proven successful in canonical 1-D laminar flows. For complex turbulent flow problems, including the conservation laws in the loss function may lead to stiffer optimizations [51]. 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 [14, 16, 17] 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 can apply this information (via transfer learning) to high-resolution inputs. We proposed two variations for transfer learning in SURFNet: 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 $256\times$ larger than the tiny discretization used in training and a uniform $2 - 2.1\times$ speedup across target resolutions and test geometries compared to the OpenFOAM CFD solver. SURFNet is able to recover high-resolution flow features with $15\times$ less data at high resolutions and reducing the combined data collection and training time by $3.6\times$ and $10.2\times$ at 256×256 and 512×512 grid sizes, respectively.

Future work includes expanding SURFNet to a variety of flow conditions, including different freestream boundary conditions, Reynolds numbers, and/or Mach numbers. An area of further research is to consider SURFNet across additional physical domains. dd As SURFNet is computationally inexpensive, and data from low-resolution CFD simulations are widely available, further explorations of the generalization are warranted. SURFNet's underlying CNN architecture is unconstrained, and therefore areas such as solid mechanics or molecular dynamics could leverage the same architecture. SURFNet could also be beneficial for transfer learning across physical domains, with predictions from a coarse resolution simulation to a finer resolution.

APPENDIX

A. Abstract

The artifact contains the source code for SURFNet. This code provides dataset generation, model training, model transfer, and model prediction, together with the OpenFOAM physics solver setup and acceleration through SURFNet's inference. The artifact is well-documented and is designed to be re-used and reproduced at any academic or industry research level since it is amenable to any Computational Fluid Dynamics (CFD) solver that targets steady simulations.

B. Artifact check-list (meta-information)

- **Program: The open-source OpenFOAM physics solver is required for speedup evaluation.**
- **Data set: training and validation data sets to obtain the coarse model, and transfer and validation data sets for transfer learning.**
- **Hardware: 4 Tesla V100 NVIDIA GPUs**
- **Software required: tensorflow, conda, docker, python, and OpenFOAM**
- **Runtime Environment: CentOS 7.0 with conda and docker installed and available.**
- **How much disk space required (approximately)?: 50 GB**
- **How much time is needed to prepare workflow (approximately)?: 30 minutes**
- **How much time is needed to complete experiments (approximately)?: 4 hours if hardware available is the specified before**
- **Publicly available? Yes, on Zenodo**

C. Description

1) Source code: The source code of our work is hosted in Zenodo and can be downloaded from:

```
https://doi.org/10.5281/zenodo.5139858
```

2) Hardware dependencies: We developed and tested our code on a dual-socket Intel Xeon Gold 6148, totalling 40 cores, and 4 V100 Tesla NVIDIA GPUs. We expect that our code will run on GPUs with a compute capability no less than 5.0

3) Software dependencies: Our work requires both tensorflow and the OpenFOAM physics solver. Tensorflow is required for model training, model transfer, and model prediction (*inference*). The OpenFOAM physics solver is required to evaluate the speedups provided by *inference*. We use Python 3.8 for our python scripts, that require `numpy`, `glob`, `openfoamparser`, `h5py`, and `gdown`. They all can be installed through `pip`.

If using `conda`, run the following command to install and use tensorflow:

```
conda create -n tf-gpu tensorflow-gpu
```

For OpenFOAM, we recommend using `docker`:

```
docker pull openfoam/openfoam6-paraview56
```

4) Data sets: We provide the command to download the data sets to (1) train the coarse model as described in Section IV-D and (2) perform transfer learning to a specific target discretization. Hence, results from Figure 6 and Table IV can be reproduced. Refer to the `READMEdatasets` file inside the SURFNET directory to find a detailed explanation of the composition of the datasets.

5) Models: Models are not given and will be generated through the experimental workflow so the user can reproduce results from the paper, or generate any desired model with their own data sets.

D. Installation

Download the source code in the zip file hosted on Zenodo and `cd` to the main directory named SURFNET. Assuming `conda` and `docker` are installed and available:

1) Install tensorflow with the command provided before:

```
conda create -n tf-gpu tensorflow-gpu
```

2) Download the OpenFOAM docker image as stated before:

```
docker pull openfoam/openfoam6-paraview56
```

3) Install all the python dependencies required for the project:

```
pip install -r requirements.txt
```

E. Experiment workflow

A step-by-step guide to carry an experiment workflow is found in the `READMEworkflow` file in the main directory. We detail the steps for the user to generate required data sets (train, validation, etc.), train the coarse model, transfer learn to a desired discretization, do model prediction, and accelerate the OpenFOAM physics solver.

F. Evaluation and expected results

Because the validation of the results requires interaction between tensorflow and OpenFOAM, a dedicated README file can be found in the directory. In the `READMEresults` file we provide the experimental workflow executed to obtain results of this paper. User can validate coarse model setup, training, and losses, model transfer losses and OpenFOAM acceleration. To summarize, users should observe:

- A 2.5 hours long training of the coarse model (resolution 64x256) leading to training and validation MSE losses as described in IV-D
- A 4-minute long model transfer to a higher discretization.
- A $2\times$ speedup over the OpenFOAM physics solver for the NACA 1412 airfoil simulation when using SURFNet's inference as initial condition to OpenFOAM.

REFERENCES

[1] M. Lee, N. Malaya, and R. D. Moser, "Petascale direct numerical simulation of turbulent channel flow on up to 786k cores," pp. 1–11, 2013.

[2] J. Casacuberta, K. Groot, Q. Ye, and S. Hickel, "Transitional flow dynamics behind a micro-ramp," *Flow, Turbulence and Combustion*, vol. 104, no. 2, pp. 533–552, 2020.

[3] J. Blazek, *Computational fluid dynamics: principles and applications*. Butterworth-Heinemann, 2015.

[4] H. Jasak, A. Jemcov, Z. Tukovic *et al.*, "Openfoam: A c++ library for complex physics simulations," in *International workshop on coupled methods in numerical dynamics*, vol. 1000. IUC Dubrovnik Croatia, 2007, pp. 1–20.

[5] O. Reynolds, "Iv. on the dynamical theory of incompressible viscous fluids and the determination of the criterion," *Philosophical transactions of the royal society of london.(a.)*, no. 186, pp. 123–164, 1895.

[6] B. Mostafazadeh Davani, F. Marti, B. Pourghassemi, F. Liu, and A. Chandramowlishwaran, "Unsteady navier-stokes computations on gpu architectures," in *23rd AIAA Computational Fluid Dynamics Conference*, 2017, p. 4508.

[7] B. Mostafazadeh, F. Marti, F. Liu, and A. Chandramowlishwaran, "Roofline guided design and analysis of a multi-stencil CFD solver for multicore performance," in *Proc. 32nd IEEE Int'l. Parallel and Distributed Processing Symp. (IPDPS)*, Vancouver, British Columbia, Canada, May 2018, pp. 753–762.

[8] National Advisory Committee for Aeronautics airfoils, "NACA Family of Airfoils," https://www.nasa.gov/image-feature/langley/100/naca-airfoils, 1929.

[9] S. T. Bose and G. I. Park, "Wall-modeled large-eddy simulation for complex turbulent flows," *Annual review of fluid mechanics*, vol. 50, pp. 535–561, 2018.

[10] E. Nehme, L. E. Weiss, T. Michaeli, and Y. Shechtman, "Deep-storm: super-resolution single-molecule microscopy by deep learning," *Optica*, vol. 5, no. 4, pp. 458–464, 2018.

[11] W. Yang, X. Zhang, Y. Tian, W. Wang, J.-H. Xue, and Q. Liao, "Deep learning for single image super-resolution: A brief review," *IEEE Transactions on Multimedia*, vol. 21, no. 12, pp. 3106–3121, 2019.

[12] Z. Wang, J. Chen, and S. C. Hoi, "Deep learning for image super-resolution: A survey," *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[13] C. M. Jiang, S. Esmaeilzadeh, K. Azizzadenesheli, K. Kashinath, M. Mustafa, H. A. Tchelepi, P. Marcus, A. Anandkumar *et al.*, "Meshfreeflownet: A physics-constrained deep continuous space-time super-resolution framework," *arXiv preprint arXiv:2005.01463*, 2020.

[14] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Fourier neural operator for parametric partial differential equations," *arXiv preprint arXiv:2010.08895*, 2020.

[15] H. Gao, L. Sun, and J.-X. Wang, "Super-resolution and denoising of fluid flow using physics-informed convolutional neural networks without high-resolution labels," *arXiv preprint arXiv:2011.02364*, 2020.

[16] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Neural operator: Graph kernel network for partial differential equations," *arXiv preprint arXiv:2003.03485*, 2020.

[17] K. Bhattacharya, B. Hosseini, N. B. Kovachki, and A. M. Stuart, "Model reduction and neural networks for parametric pdes," *arXiv preprint arXiv:2005.03180*, 2020.

[18] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, "Huggingface's transformers: State-of-the-art natural language processing," *arXiv preprint arXiv:1910.03771*, 2019.

[19] Y. You, Z. Zhang, C.-J. Hsieh, J. Demmel, and K. Keutzer, "Imagenet training in minutes," in *Proceedings of the 47th International Conference on Parallel Processing*, 2018, pp. 1–10.

[20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR 2015*, 2015. [Online]. Available: http://arxiv.org/abs/1409.4842

[21] R. Wang, K. Kashinath, M. Mustafa, A. Albert, and R. Yu, "Towards physics-informed deep learning for turbulent flow prediction," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1457–1466.

[22] X. Guo, W. Li, and F. Iorio, "Convolutional neural networks for steady flow approximation," pp. 481–490, 2016.

[23] O. Obiols-Sales, A. Vishnu, N. Malaya, and A. Chandramowlishwaran, "Cfdnet: a deep learning-based accelerator for fluid simulations," *arXiv preprint arXiv:2005.04485*, 2020.

[24] W. Dong, J. Liu, Z. Xie, and D. Li, "Adaptive neural network-based approximation to accelerate eulerian fluid simulation," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2019, pp. 1–22.

[25] N. Thuerey, K. Weißenow, L. Prantl, and X. Hu, "Deep learning methods for reynolds-averaged navier–stokes simulations of airfoil flows," *AIAA Journal*, pp. 1–12, 2019.

[26] L. Zhu, W. Zhang, J. Kou, and Y. Liu, "Machine learning methods for turbulence modeling in subsonic flows around airfoils," *Physics of Fluids*, vol. 31, no. 1, p. 015105, 2019.

[27] "NACA0012 grids," https://turbmodels.larc.nasa.gov/naca0012numerics_grids.html.

[28] J. Boussinesq, *Essai sur la théorie des eaux courantes*. Impr. nationale, 1877.

[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] G. Kalitzin, G. Medic, G. Iaccarino, and P. Durbin, "Near-wall behavior of rans turbulence models and implications for wall functions," *Journal of Computational Physics*, vol. 204, no. 1, pp. 265–291, 2005.

[44] S. B. Pope, "Turbulent flows," 2001.

[45] 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.

[46] 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.

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

[48] "EarlyStopping keras callback," https://keras.io/api/callbacks/early_stopping/.

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

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

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

[52] 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.