Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

FlowDNN: a physics-informed deep neural network for fast and accurate flow prediction*

Donglin CHEN[†§1], Xiang GAO[†§1,2], Chuanfu XU[†‡1,2], Siqi WANG[1,2], Shizhao CHEN[1], Jianbin FANG[1], Zheng WANG[3]

[1]*College of Computer, National University of Defense Technology, Changsha 410073, China*

[2]*State Key Laboratory of High Performance Computing, National University of Defense Technology, Changsha 410073, China*

[3]*School of Computing, University of Leeds, Leeds LS29JT, UK*

[†]E-mail: chendonglin14@nudt.edu.cn; gaoxiang12@nudt.edu.cn; xuchuanfu@nudt.edu.cn

Received Aug. 28, 2020; Revision accepted May 4, 2021; Crosschecked Jan. 5, 2022

Abstract: For flow-related design optimization problems, e.g., aircraft and automobile aerodynamic design, computational fluid dynamics (CFD) simulations are commonly used to predict flow fields and analyze performance. While important, CFD simulations are a resource-demanding and time-consuming iterative process. The expensive simulation overhead limits the opportunities for large design space exploration and prevents interactive design. In this paper, we propose FlowDNN, a novel deep neural network (DNN) to efficiently learn flow representations from CFD results. FlowDNN saves computational time by directly predicting the expected flow fields based on given flow conditions and geometry shapes. FlowDNN is the first DNN that incorporates the underlying physical conservation laws of fluid dynamics with a carefully designed attention mechanism for steady flow prediction. This approach not only improves the prediction accuracy, but also preserves the physical consistency of the predicted flow fields, which is essential for CFD. Various metrics are derived to evaluate FlowDNN with respect to the whole flow fields or regions of interest (RoIs) (e.g., boundary layers where flow quantities change rapidly). Experiments show that FlowDNN significantly outperforms alternative methods with faster inference and more accurate results. It speeds up a graphics processing unit (GPU) accelerated CFD solver by more than 14 000×, while keeping the prediction error under 5%.

Key words: Deep neural network; Flow prediction; Attention mechanism; Physics-informed loss
https://doi.org/10.1631/FITEE.2000435 **CLC number:** TP391

1 Introduction

Fast and accurate determination of flow fields and performance is critical for flow-related design and optimization. The focus of our work is the analysis of incompressible steady flow, which is common in many industrial engineering applications such as automobiles, the environment, and architecture. Computational fluid dynamics (CFD) simulations are a vital methodology for analyzing and predicting flow fields and performance. Traditionally, CFD methods discretize governing equations of fluid dynamics, e.g., Navier-Stokes equations (Constantin and Foias, 1988), into a set of large-scale linear equations and then solve them iteratively (Blazek, 2015). While producing highly accurate results, CFD simulations are known for their high computational cost,

§ These two authors contributed equally to this work
‡ Corresponding author
* Project supported by the National Natural Science Foundation of China (Nos. 61772542, 61972408, and 12102467) and the Foundation of the State Key Laboratory of High Performance Computing, China (Nos. 201901-11 and 202001-03)
ORCID: Xiang GAO, https://orcid.org/0000-0002-8216-7482; Chuanfu XU, https://orcid.org/0000-0002-4876-2368

memory usage, and running time. This expensive simulation overhead inevitably prolongs the design process, increasing the cost and hampering exhaustive design space exploration and interactive design. This is a particular problem in the early design stages where designers would like to quickly estimate the potential benefits of numerous design choices.

Recent studies have taken a data-driven, supervised-learning-based approach to fast approximation of flow fields and performance by leveraging deep neural networks (DNNs) (Ronneberger et al., 2015; Guo et al., 2016; Thuerey et al., 2020). Such approaches work by learning, offline, a DNN from empirical information produced by a full-order CFD solver. The learned model can then be used to solve new, unseen flow problems, by taking as input a representation of the flow conditions and geometric shapes (e.g., usually projected as a two-dimensional (2D) matrix that can be visualized as an artificial image like other DNN-based image processing applications), and predicting a matrix of the flow fields or performance metrics (e.g., aerodynamic coefficients). By employing a model inference to substitute for the many computational iterations that a CFD solver entails, predictive modeling can essentially decrease the turnaround time of generating flow data.

While promising, the field of DNN-based flow approximation is still in its infancy. Existing approaches simply leverage established statistical models developed in the field of image processing. They have a fundamental flaw because they are not aware of the underlying physical principles of fluid flows. Unlike photo images, flow field images are visualizations of CFD numerical simulations that must satisfy the fundamental physical laws like mass and momentum conservation. Existing statistic-based flow models are trained to minimize the mean square errors (MSEs) of the prediction results, but a prediction with a low MSE does not guarantee the preservation of physical laws. As a result, prior methods often produce physically unsound (and thus unusable) data, which in turn discourage the adoption of the technique. Some of the most recent studies have attempted to incorporate knowledge of the physical system into deep learning (DL) (Geneva and Zabaras, 2019; Wang et al., 2020). These methods are tuned for modeling turbulence simulations, and the model architectures are tightly coupled with certain simulation methods. As a result, they do not generalize to steady flow prediction and other simulation methods. As we show later in this paper, existing approaches give large prediction errors in steady flow simulation scenarios.

In addition to ignoring the physical laws, prior work also fails to capitalize on the optimization opportunities of CFD workloads to improve prediction accuracy. Specifically, for flow field prediction, we want to direct the model to pay attention to regions like boundary layers, because CFD users are more interested in these complex regions with sharp gradients. This is different from classical image processing tasks, like object recognition, in which we want the model to pay more attention to the location and size of a target. Because prior DNN-based flow approximation methods simply adapt existing models developed for standard image processing, they are not tuned for CFD workloads and thus miss massive optimization opportunities. Moreover, evaluations for accuracy and physical characteristics of predicted flow fields are often deficient or not comprehensive in these emerging DNN-based flow prediction methods, making no distinction for the whole flow field or a specific region of interest (RoI), e.g., boundary layers.

In this paper, we propose FlowDNN, a physics-informed deep convolutional neural network with attention mechanisms and network pruning for highly accurate and fast steady flow prediction. FlowDNN is designed to produce predictions that obey the physical laws and use the CFD workload characteristics to improve the quality and accuracy of flow predictions. Unlike prior work in Wang et al. (2020), FlowDNN is not closely linked to a specific simulation method and can be applied to a wide range of simulation algorithms. To preserve the laws of conservation of mass and momentum of fluid dynamics, FlowDNN incorporates a novel physical loss function. This novel loss function allows FlowDNN to dramatically enhance the prediction accuracy while meeting the physical consistency of the predicted flow fields. To leverage the domain characteristics of CFD simulations, FlowDNN employs two new attention mechanisms to better extract knowledge from areas with sharp gradients.

We apply FlowDNN to a real-life flow dataset and derive various metrics for the whole flow fields or specific RoIs to evaluate the accuracy and physical consistency of FlowDNN prediction. Compared

with three state-of-the-art DL-based CFD approximation methods, our approach significantly outperforms competitive methods with faster prediction and smaller prediction error for steady flow prediction. When compared to a state-of-the-art graphics processing unit (GPU) accelerated parallel CFD solver, our approach speeds up the simulation by orders of magnitude (more than 14 000×). The key contribution of this paper is a general DNN for fast steady flow simulations that can preserve physical principles. Our approach not only delivers fast and more accurate simulation predictions, but also ensures that the prediction outcomes obey desirable physical characteristics, filling the gap in learning-based steady flow prediction.

2 Related work

2.1 Data-driven flow field modeling

Data-driven methods have been used to accelerate aerodynamic simulations. Early work in the areas adopts classical machine learning methods like polynomial regression, support vector machines, and artificial neural networks (Daberkow and Mavris, 1998; Balabanov et al., 1999; Ahmed and Qin, 2009; Raissi et al., 2017). These strategies work in small-scale settings but cannot scale to the whole flow field.

In recent years, efforts have been devoted to applying DL to fluid dynamics. For example, Ling et al. (2016) and Geneva and Zabaras (2019) constructed customized neural networks for turbulence modeling. Wang et al. (2020) presented a novel hybrid DL model that unifies representation learning and turbulence simulation techniques, achieving improvement in both the prediction error and desired physical quantities. However, these works target turbulence modeling and are tightly coupled to a specific simulation algorithm. Guo et al. (2016) and Bhatnagar et al. (2019) were among the first attempts at predicting steady flow fields, but their models do not guarantee that the prediction outcome will obey the fundamental physical laws. For the prediction of unsteady flow, Lee and You (2019) predicted the flow fields over a circular cylinder using diverse DL networks to refine both spatial and temporal features of the input flow field. Thuerey et al. (2020) focused on investigating the accuracy of a modernized U-Net model for steady flow field prediction, but the model lacks physical constraints. To assess the capabilities of neural networks to predict temporally evolving turbulent flows, Srinivasan et al. (2019) proposed two types of neural networks, multi-layer perceptron (MLP) and long short-term memory (LSTM), to predict turbulent shear flows, and LSTM led to excellent results. To further reduce the amount of data and time required for training, Guastoni et al. (2020) assessed the feasibility of performing transfer learning for the fully convolutional network (FCN) model between different Reynolds numbers. The results show great potential to exploit initial training in a certain flow condition and transfer this knowledge to another condition. This paper presents the first generalized prediction framework for steady flow simulations, which incorporates physical principles in the design, training, and inference of the model. Our work extends the U-Net architecture to model complex flow datasets.

2.2 Image-to-image mapping

Our work converts the flow field prediction problem to an image-to-image regression. It employs DNN models to find the right mapping from given inputs to the expected simulation outcomes based on the assigned tasks. There is an extensive body of work on image-to-image processing tasks, including image segmentation (Long et al., 2015; Ronneberger et al., 2015; Zhou et al., 2018) and image translation (Isola et al., 2017; Kim et al., 2017; Zhu et al., 2017; Amodio and Krishnaswamy, 2019). Prior works in these areas are concerned mainly with medical or natural images from an unknown physical process and do not incorporate physical principles to guide network training. The flow prediction problem targeted in this work is different from the conventional image-to-image translation task, because the flow data are generated by solving specific governing equations and must satisfy the physical laws. Our work contributes by introducing a novel physical loss function to ensure the physical consistency of the predictions.

3 Our approach

3.1 Problem definition

In this study, we train and test our DNN model using mainly steady flow data and problems. Our

model can also be evaluated with more complicated flow problems in the future. Steady flows are very common in many industrial applications when dealing with low-speed flow motion, where the fluid properties at a point in the system do not change over time. In many situations, such as the flow condition set in this study, the changes in pressure and temperature are sufficiently small so that the changes in density are negligible. In this case, the flow can be modeled as an incompressible flow (Constantin and Foias, 1988). For 2D incompressible steady situations, the macroscopic governing equations can be expressed as follows:

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0, \tag{1}$$

$$\frac{\partial(uu)}{\partial x}+\frac{\partial(uv)}{\partial y}=\frac{\partial\tau_{xx}}{\partial x}+\frac{\partial\tau_{yx}}{\partial y}-\frac{\partial p}{\partial x}, \tag{2}$$

$$\frac{\partial(vu)}{\partial x}+\frac{\partial(vv)}{\partial y}=\frac{\partial\tau_{xy}}{\partial x}+\frac{\partial\tau_{yy}}{\partial y}-\frac{\partial p}{\partial y}, \tag{3}$$

$$e_{\mathrm{in}}+\frac{u^2+v^2}{2}=e, \tag{4}$$

where Eq. (1) defines the conservation of mass, Eqs. (2) and (3) define the conservation of momentum, and Eq. (4) comes from the conservation of energy. Specifically, u and v stand for the predicted flow components. τ_{xx}, τ_{yx}, τ_{xy}, and τ_{yy} are the components of the viscous stress tensor, p stands for the pressure, e_{in} is the internal energy per unit mass, and e is the total energy per unit mass. The physical loss function presented in this work is derived from the first three equations, but the effect of pressure p is omitted. Because the proposed FlowDNN model currently predicts velocity vectors only for steady flows, the input data and the training dataset do not include pressure fields. The solver implemented to generate the ground-truth flow fields is based on the lattice Boltzmann method (LBM). LBM considers the macroscopic motion of the fluid as the average result of the microscopic motion of the particles, where the microscopic motion is based on the molecular kinematic theory and statistical mechanics, and its particle distribution function satisfies the Boltzmann equation, which is more basic than Eqs. (1)–(4). The LBM solver has many advantages. For example, it can directly solve the flow fields on Cartesian grids, and the algorithm is parallel, which enables us to efficiently simulate many training samples on parallel computers (Li et al., 2016).

In this work, we apply our techniques to the 2D velocity simulation, but our approach is equally applicable to other fluid flows, including three-dimensional (3D) steady flow simulations.

3.2 Data representation

To predict flow fields over different objects with deep networks, we first need to have an appropriate way to represent the object's geometric and domain boundaries. In this study, we use LBM simulation results as our training CFD data for deep networks and divide fluid domains into Cartesian grids. For each lattice cell of the grid, there is an identifier to define whether it is the solid part of the fluid domain, and macroscopic physical quantities of the flow field simulated using LBM are stored in the cell center. As shown in Fig. 1, the blue cells are those solid parts that represent the geometry of the 2D illustrated airplane. This image-like array storage inspires us to use artificial images to represent flow fields and boundaries, and to transform the flow field prediction into an image-to-image regression problem.

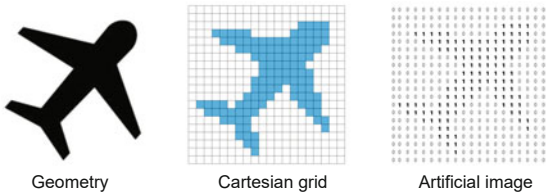


Fig. 1 Converting a 2D domain boundary to a Cartesian grid to generate a matrix input for our model

References to color refer to the online version of this figure

In this work, we use a binary representation to characterize artificial input images and recognize object boundaries in fluid domains. In Fig. 1, pixels with value 1 indicate the object boundaries, other pixels with value 0 demonstrate the fluid domain, and the corresponding pixels of the artificial output images represent the approximation of steady flow quantities after end-to-end learning.

With this kind of data representation, we can express different flow field quantities as artificial images. For example, a 2D velocity field can be expressed as an image with two channels indicating the velocity components in the x- and y-direction, and this representation can be easily extended to 3D problems. Our methods are extensible to deal with training data generated from CFD solvers using

structured/unstructured grids (Farrashkhalvat and Miles, 2003), because we can map the domain boundaries and ground-truth flow fields onto a Cartesian grid.

3.3 Network architecture

Fig. 2 illustrates the overall architecture of FlowDNN. Our model takes input as a matrix that describes a 2D geometry domain (of size 128×256 in this work) of the target object. To generate the input matrix, we first divide the fluid domains into Cartesian grids from which we map the input fluid domain image to a matrix of 0 and 1. This process is illustrated in Fig. 1. Our model predicts the steady flows around the object given an artificial image that represents flow fields and boundaries. The model produces two matrices of size 128×256 with numerical values, where a matrix represents the velocity field for the x- or y-direction.

At the core of FlowDNN is a U-Net (Ronneberger et al., 2015) architecture for steady flow prediction around arbitrary objects. U-Net was traditionally used in image segmentation to determine the area to which a pixel belongs. In this work, we extend U-Net to predict flow quantities with physical consistency for each pixel. This is achieved by using a physical loss function to constrain the training process (Section 3.4). Unlike classical U-Net for image processing that uses a pooling layer for downsampling, we adopt a transposed convolutional kernel with a stride of 2 for downsampling. This allows the network to adjust the filter weights used for each pixel to enable a more accurate prediction at the pixel level.

FlowDNN has a typical U-shaped structure and comprises mainly two parts (Fig. 2): the left part includes seven encoder blocks and the right part has seven decoder blocks. Each encoder block is followed by a convolutional layer, an activation unit, and a batch normalization layer. The convolutional kernels have a size of 4×4, except for the one in the last encoder block because the size of its input feature map is only 1×2. For each decoder block, we set up an upsampling layer followed by an activation unit.

The encoder and the decoder are connected through a skip architecture, which concatenates all down-sampled feature maps from the encoder blocks to the corresponding maps in decoder blocks and doubles the number of channels. We also extend the canonical U-Net architecture by introducing attention modules (AMs), including a channel attention module (CAM) at the bottom and six spatial attention modules (SAMs) at all other skip connections. These AMs help the skip architecture integrate the fine-grained and coarse-grained information more effectively.

3.4 Physical loss functions

Our approach explicitly provides prior physical conservation law information to the network

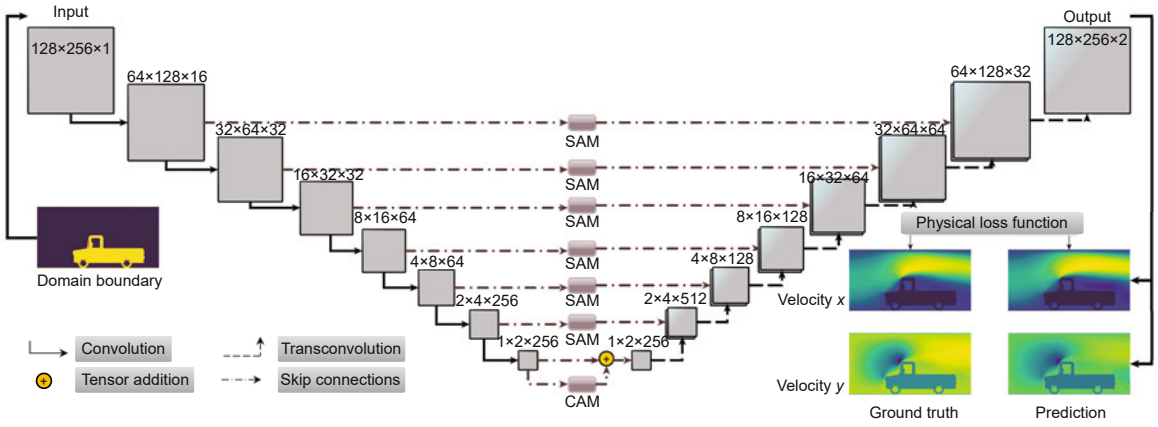


Fig. 2 The architecture of FlowDNN

The black arrows denote convolutional layers and transposed convolutional layers, while the brown arrows indicate the skip connections with the attention module (AM). The artificial image of the domain boundary is passed to the network as input. The output is the prediction of a 2D velocity field and is compared to the ground-truth data with the physical loss function. References to color refer to the online version of this figure

to enable it to extract features that satisfy physical consistency. To this end, we design two loss functions, L_{mass} and L_{momentum}, for the laws of conservation of mass and momentum, respectively, and combine them with the traditional L_1 loss function to formulate the physical loss function, L_{physical}, as

$$L_{\text{physical}} = \alpha_1 L_1 + \alpha_2 L_{\text{mass}} + \alpha_3 L_{\text{momentum}}, \quad (5)$$

where the weights (α_1, α_2, and α_3) of L_{physical} are set to ensure an equal contribution of the three terms (L_1, L_{mass}, and L_{momentum}) to the total loss. For 2D geometries, L_1 and L_{mass} are defined as follows:

$$L_1 = \frac{1}{2mn_xn_y}\sum_{l=1}^{m}\sum_{i=1}^{n_x}\sum_{j=1}^{n_y}\left(\left|u_{ij}^l-\overline{u}_{ij}^l\right|+\left|v_{ij}^l-\overline{v}_{ij}^l\right|\right), \quad (6)$$

$$L_{\text{mass}} = \frac{1}{m\left(n_x-2\right)\left(n_y-2\right)} \cdot\sum_{l=1}^{m}\sum_{i=2}^{n_x-1}\sum_{j=2}^{n_y-1}\left|\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)_{ij}^{l}-\left(\frac{\partial \overline{u}}{\partial x}+\frac{\partial \overline{v}}{\partial y}\right)_{ij}^{l}\right|. \quad (7)$$

L_{momentum} is defined in Eq. (8), given at the bottom of this page. In Eqs. (6)–(8), m is the batch size, l denotes a certain sample, and n_x and n_y are the numbers of cells (pixels) along the x- and y-direction, respectively. u and v are the flow components of the x- and y-direction, respectively, and $\overline{u}$ and $\overline{v}$ stand for the predicted flow components. L_{mass} is the loss function based on the law of conservation of mass, which evaluates the difference between predicted and ground-truth mass flowing through each cell (the density is typically assumed to be constant for an incompressible steady flow). L_{momentum} is the loss function based on the law of conservation of momentum, which compares the difference of momentum in the x- and y-direction. Re represents the fixed Reynolds number. Note that the fluid condition is typically quantified by a dimensionless Reynolds number (Blazek, 2015) that describes the ratio of inertial force to viscous force in a flowing fluid. Here the first- and second-order partial derivatives are calculated using the first- and second-order central difference schemes, respectively (Blazek, 2015). Taking variable u as an example, we have

$$\begin{cases}\dfrac{\partial u_{i,j}}{\partial x} = \dfrac{1}{2}\left(u_{i+1,j}-u_{i-1,j}\right),\\ \dfrac{\partial^2 u_{i,j}}{\partial x^2} = u_{i+1,j}-2u_{i,j}+u_{i-1,j}.\end{cases} \quad (9)$$

Note that we remove both the ground truth and predicted pressure terms in L_{momentum}, so the momentum equation is still conserved (if the predicted velocities equal the ground truth, L_{momentum} will be 0). In future work, it would be worth trying to use the automatic differentiation technique for the physical loss function, like the physics-informed neural networks proposed by Raissi et al. (2019).

3.5 Channel and spatial attention modules

As a departure from all prior work on DL-based CFD approximation, we introduce attention mechanisms to our learning framework. This is motivated by the observation that some RoIs in fluid flows often contain more important and complicated information than others as flow quantities change rapidly. To achieve accurate predictions in these areas, we introduce the self-attention mechanism (Hu et al., 2018; Park et al., 2018) to direct the networks to focus on RoI areas. Specifically, FlowDNN adopts two lightweight AMs, CAM and SAM, which are extended from Woo et al. (2018). CAM and

$$\begin{aligned}&L_{\text{momentum}}\\ &=\frac{1}{m\left(n_x-2\right)\left(n_y-2\right)}\sum_{l=1}^{m}\sum_{i=2}^{n_x-1}\sum_{j=2}^{n_y-1}\left\{\left|\left[\left(\frac{\partial(uu)}{\partial x}+\frac{\partial\left(uv\right)}{\partial y}\right)_{ij}^{l}-\frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)_{ij}^{l}\right]\right.\right.\\ &\quad-\left[\left(\frac{\partial(\overline{uu})}{\partial x}+\frac{\partial\left(\overline{uv}\right)}{\partial y}\right)_{ij}^{l}-\frac{1}{Re}\left(\frac{\partial^2 \overline{u}}{\partial x^2}+\frac{\partial^2 \overline{u}}{\partial y^2}\right)_{ij}^{l}\right]\Bigg|+\left|\left[\left(\frac{\partial(vu)}{\partial x}+\frac{\partial\left(vv\right)}{\partial y}\right)_{ij}^{l}-\frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right)_{ij}^{l}\right]\right.\\ &\quad\left.\left.-\left[\left(\frac{\partial(\overline{vu})}{\partial x}+\frac{\partial\left(\overline{vv}\right)}{\partial y}\right)_{ij}^{l}-\frac{1}{Re}\left(\frac{\partial^2 \overline{v}}{\partial x^2}+\frac{\partial^2 \overline{v}}{\partial y^2}\right)_{ij}^{l}\right]\right|\right\}.\end{aligned} \quad (8)$$

SAM can extract the discriminative features from the channel and the spatial domains respectively to facilitate FlowDNN by learning which information (e.g., boundary information) to emphasize. The following equations show how these two AMs work:

$$\boldsymbol{F}_{\mathrm{c}} = \boldsymbol{M}_{\mathrm{c}}(\mathbb{F}) \otimes \mathbb{F}, \tag{10}$$

$$\boldsymbol{F}_{\mathrm{s}} = \boldsymbol{M}_{\mathrm{s}}(\mathbb{F}) \otimes \mathbb{F}, \tag{11}$$

$$\boldsymbol{M}_{\mathrm{c}}(\mathbb{F}) = \sigma(\mathrm{MLP}(\mathrm{GAP}(\mathbb{F})) + \mathrm{MLP}(\mathrm{GMP}(\mathbb{F}))), \tag{12}$$

$$\boldsymbol{M}_{\mathrm{s}}(\mathbb{F}) = \sigma\left(\mathrm{Conv}\left(\mathrm{GAP}_{\mathrm{c}}(\mathbb{F}) \oplus \mathrm{GMP}_{\mathrm{c}}(\mathbb{F})\right)\right), \tag{13}$$

where $\otimes$ and $\oplus$ denote element-wise multiplication and channel-wise concatenation, respectively. $\mathbb{F} \in \mathbb{R}^{C\times H\times W}$ indicates the input feature map, while $\boldsymbol{M}_{\mathrm{c}} \in \mathbb{R}^{C\times 1\times 1}$ and $\boldsymbol{M}_{\mathrm{s}} \in \mathbb{R}^{1\times H\times W}$ represent the CAM and SAM, respectively. The intermediate results $\boldsymbol{M}_{\mathrm{c}}(\mathbb{F})$ and $\boldsymbol{M}_{\mathrm{s}}(\mathbb{F})$ need $\otimes$ with $\mathbb{F}$ itself, matching the dimension of the original input and obtaining the outputs $\boldsymbol{F}_{\mathrm{c}}$ and $\boldsymbol{F}_{\mathrm{s}}$. Eqs. (12) and (13) show the details of operations in CAM and SAM. CAM first creates a global average pooling (GAP) and a global max pooling (GMP) along the spatial axis on the input feature map, producing a channel vector. The vector is then sent to an MLP with one hidden layer to estimate attention across channels. SAM also includes global pooling operations, but they are performed along the channel axis, $\mathrm{GAP}_{\mathrm{c}}$ and $\mathrm{GMP}_{\mathrm{c}}$. The results from $\mathrm{GAP}_{\mathrm{c}}$ and $\mathrm{GMP}_{\mathrm{c}}$ are concatenated and sent to a convolution operation to generate a spatial attention map with one channel. Both CAM and SAM are followed by the sigmoid function σ for normalization.

3.6 Network pruning

We also apply network pruning to speed up the inference of a trained model. Network pruning (Liu et al., 2019) is also used to verify that the improvement of FlowDNN is not simply due to the introduction of more learning parameters.

To this end, we use a Taylor expansion based criterion from Molchanov et al. (2017) to rank the neurons in the network and iteratively remove the least important one. Pruning is performed iteratively. We first train the network until it reaches the convergence criteria. We then evaluate the importance of neurons using the Taylor expansion based criterion and remove the least important neuron. Next, we fine-tune the pruned model and re-evaluate the neurons' importance to remove the next least important neuron until reaching the target trade-off between accuracy and efficiency. We note that pruning is done offline and is a one-off cost.

4 Experiment setup

4.1 Data preparation

We evaluate FlowDNN by applying it to 2D flow field velocity predictions. Our training dataset includes 3000 samples that are a combination of simple 2D geometric primitives of ovals and rectangles, with different positions and sizes. Our validation and test datasets include 22 and 44 types of car prototypes, respectively. Fig. 3 illustrates some of our training and test samples. We use the parallel GPU implementation of the LBM (Ernst, 1981) provided by an open-source CFD solver (http://mechsys.nongnu.org) to generate our training CFD data (i.e., the ground-truth velocities after performing CFD simulations on the input data). We choose LBM because it is a widely used CFD simulation method and can be parallelized to run on GPU. For LBM simulations, the Reynolds number is set to 400, and the airflow blows toward the 2D object parallel to the x-direction. The Cartesian grid size and input artificial image size are both 256×128. For flow fields simulated by CFD methods like finite volume and spectral method, an extra step is required to first interpolate the result onto a Cartesian grid.

4.2 Implementation and training details

Our models are built on an NVIDIA Tesla V100 GPU with PyTorch 1.1.0. We train the model with the adaptive moment estimation (Adam) optimizer. To converge to stable results without overfitting, the training proceeds up to 400 epochs. Table 1 shows the hyper-parameter setting and tuning range. We set the initial learning rate at 4×10^{-4} and decay it every 25 epochs by a factor of 0.9. The batch size is set to 16. Note that transposed convolutions may cause checkerboard artifacts (Odena et al., 2016). Thus, we set the kernel size divisible by the stride to avoid this drawback. For each network pruning step, we remove only the lowest ranking neuron, namely the filter, from the network because pruning too

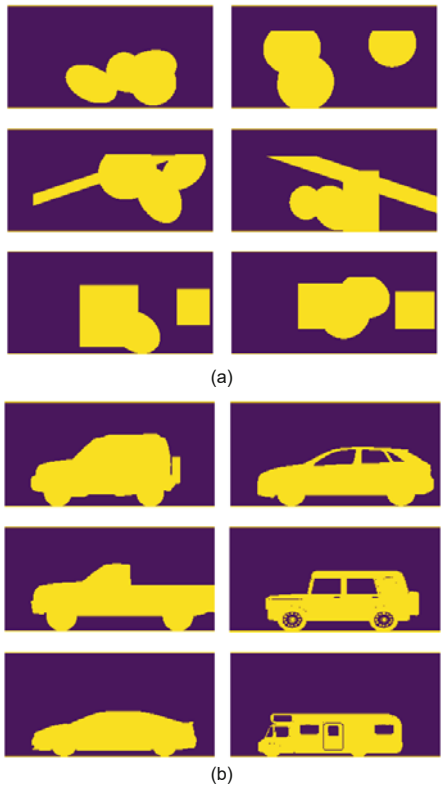


Fig. 3 Visualization examples from our training (a) and test (b) datasets

Table 1 Hyper-parameter setting

Parameter	Optimum	Tuning range
Lr	4×10^{-4}	10^{-5} – 10^{-3}
Lr decay interval	25	20 – 50
Batch size	16	4 – 64
Filter size	4	2 – 8
Pruning number	1	1 – 5
Weight of L_{physical}	(1, 5, 25)/3	–

Pruning number: number of filters for each pruning step;
Lr: learning rate

much at each step may lead to a damaged network. We then fine-tune the pruned network by training for 400 epochs to converge to a stable result. As for activation functions, we use the conventional rectified linear units (ReLUs) function, which performs better than the exponential linear units (ELUs) function as recommended in Hamdan et al. (2019). For the weights of our physical loss function, we first keep α_1, the distribution of L_1 loss, equal to 1. Then α_2 and α_3 are set to make the items contribute equally to the total loss. Because there are three components in L_{physical}, we divide all three parameters by 3.

4.3 Baselines

We compare our model with three state-of-the-art baselines for steady flow field prediction:

1. C-Net (Guo et al., 2016): an encoder-decoder model with three convolutional layers and three deconvolutional layers, providing lightweight interior and exterior flow performance feedback.

2. T-Net (Thuerey et al., 2020): a modernized U-Net structure aiming at the inference of pressure and velocity distributions for Reynolds-averaged Navier-Stokes solutions.

3. U-Net (Ronneberger et al., 2015): conventional convolutional neural networks originally developed for image segmentation, also popularly used for flow field prediction.

4.4 Evaluation metrics

We use the mean relative error (MRE) to evaluate the overall prediction accuracy for all flow fields. Because CFD users often pay particular attention to specific regions, we also evaluate the MRE for RoIs, which we call MRE_{RoI}. For example, Fig. 4 shows an institution of RoI defined by the box, including the area of the car and the boundary layers in the experiments. Note that the box is not formalized, and the size and location of the box will adaptively change to totally encompass the object geometry in the flow field. Because it is important to ensure that the predictions are physically sound, we define MRE_{ma} and MRE_{mo} for the laws of conservation of mass and momentum, respectively. These metrics are defined as follows:

1. MRE: This is computed using the predicted velocity and the ground truth for the whole 2D flow field of all the N test samples:

$$\frac{1}{N}\sum_{l=1}^{N}\Big(\sum_{i=1}^{n_x}\sum_{j=1}^{n_y}\big(\,|u_{ij}^{l}-\overline{u}_{ij}^{l}|+|v_{ij}^{l}-\overline{v}_{ij}^{l}|\,\big)\Big/\sum_{i=1}^{n_x}\sum_{j=1}^{n_y}\big(|u_{ij}^{l}|+|v_{ij}^{l}|\big)\Big).$$

2. MRE_{RoI}: This is computed as

$$\frac{1}{N}\sum_{l=1}^{N}\left(\sum_{i=1}^{n_{\text{s}}}\big(|u_i^l-\overline{u}_i^l|+|v_i^l-\overline{v}_i^l|\big)\Big/\sum_{i=1}^{n_{\text{s}}}\big(|u_i^l|+|v_i^l|\big)\right),$$

where n_{s} is the number of cells in the RoI.

3. $\mathrm{MRE}_{\mathrm{ma}}$ and $\mathrm{MRE}_{\mathrm{mo}}$: These are calculated as

$$\frac{1}{N}\sum_{l=1}^{N}\left(\sum_{i=1}^{n_x}\sum_{j=1}^{n_y}|g_{ij}^{l}-\overline{g}_{ij}^{l}|\Big/\sum_{i=1}^{n_x}\sum_{j=1}^{n_y}|g_{ij}^{l}|\right),$$

where g and $\overline{g}$ are the net change of mass or momentum on each lattice between ground truths and predictions, respectively.

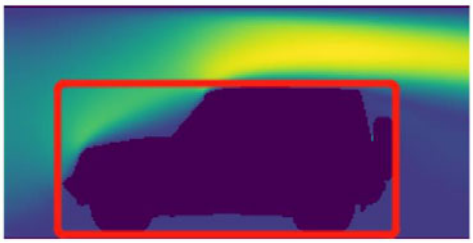


Fig. 4 An example of the region of interest (RoI) defined by the box

5 Experimental results

5.1 Overall results

Table 2 compares the baselines to FlowDNN in terms of accuracy, inference runtime, and the parameter size. Without any further optimization, FlowDNN with our physical loss function greatly outperforms its counterparts for MRE, $\mathrm{MRE}_{\mathrm{ma}}$, and $\mathrm{MRE}_{\mathrm{mo}}$. We also present MRE for the boundary domain ($\mathrm{MRE}_{\mathrm{RoI}}$), which is a specific RoI for CFD experts in our test. Our attention mechanisms are important for improving accuracy at the boundary layer; without AM, FlowDNN gives a higher $\mathrm{MRE}_{\mathrm{RoI}}$ than U-Net. We also observe that there is a slight increase in the runtime when introducing the attention mechanism, but the overhead is negligible (<1 ms).

The last row of Table 2 shows the results of network pruning (detailed in Section 5.5). We reduce the prediction time to 3.62 ms and remove almost half of the parameters (from 13.74×10^6 to 7.40×10^6), validating the effectiveness of our method compared to the three baselines with more parameters (180.25×10^6, 7.45×10^6, and 32.96×10^6). Interestingly, pruning actually improves network performance. We believe this is because having fewer parameters reduces the chance of overfitting (Frankle and Carbin, 2019). Overall, the full implementation of FlowDNN greatly improves the accuracy at less inference runtime compared to alternative methods. Because the weight parameters of the neural network are different after each training, we train each neural network three times and compare the predicted values. The prediction results show that the difference between each trained network is almost negligible, so we use the average value as the final result.

We further compare the feedback speed between our DL method and the traditional LBM solver on the GPU platform. As we can see from Table 2, because the batch size is set to 16, FlowDNN can predict the fluid velocity fields for 16 different prototypes in less than 4 ms, whereas the LBM solver needs about 3.3 s to simulate the result for only one prototype, indicating an improvement of speed by more than 1.45×10^4 times.

Table 3 quantifies the differences of some of the model prediction visualization samples against the full-order CFD simulation results, indicating that our predictions are visually closer to the results given by the full-order CFD solver.

For more performance details on four metrics of all models, Fig. 5 describes the statistical distribution of four evaluation metrics for different models on the test dataset. The three baseline models perform well on some specific samples, but FlowDNN can achieve high predictive performance in almost all cases.

5.2 Loss function

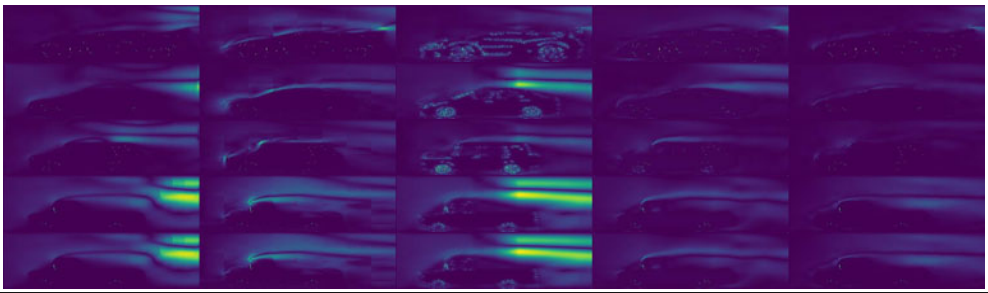
Fig. 6 reports the impact of L_1 and L_{physical} on FlowDNN without AM or network pruning. Fig. 6a compares the two loss functions for MRE, $\mathrm{MRE}_{\mathrm{RoI}}$, $\mathrm{MRE}_{\mathrm{ma}}$, and $\mathrm{MRE}_{\mathrm{mo}}$ on the test dataset. In addition to the improvement in the accuracy of all flow fields, L_{physical} reduces the prediction error dramatically at the complicated boundary layer: $\mathrm{MRE}_{\mathrm{RoI}}$ drops from 42.23% to 23.56%. Moreover, L_{physical} can provide predictions with higher physical consistency compared to the conventional L_1 with no physical constraint. Fig. 6b visualizes the absolute error for mass (Δmass) and momentum (Δmomentum) between the ground truth and predictions using different loss functions. Compared to L_1, the predictions of L_{physical} contain more fine features and are more consistent with the ground truth in the mass and momentum change of flow quantities.

Table 2 Comparing FlowDNN with different baseline models

Method	MRE	MRE_{RoI}	MRE_{ma}	MRE_{mo}	Runtime (ms)	Parameter ($\times 10^6$)
LBM	–	–	–	–	3300×16	–
C-Net	14.31%	30.99%	37.44%	45.60%	9.29	180.25
T-Net	24.65%	59.71%	79.09%	82.38%	8.47	7.45
U-Net	14.74%	13.14%	30.78%	44.42%	16.15	32.96
FlowDNN	7.91%	23.56%	18.28%	22.46%	**3.52**	13.70
FlowDNN w/ AM	5.34%	9.16%	12.34%	15.69%	4.51	13.74
FlowDNN w/ AM and P	**4.77%**	**8.87%**	**12.14%**	**14.63%**	**3.62**	**7.40**

All FlowDNN models are trained using the physical loss function with AM and network pruning. AM: attention module; LBM: lattice Boltzmann method; MRE: mean relative error; RoI: region of interest; P: network pruning

Table 3 The differences of baselines and FlowDNN compared with the ground truth

Vehicle type	U-Net	C-Net	T-Net	FlowDNN	FlowDNN AP
Racing					
Saloon					
Jeep					
Pickup					
Bus					

FlowDNN AP: FlowDNN with attention module and network pruning

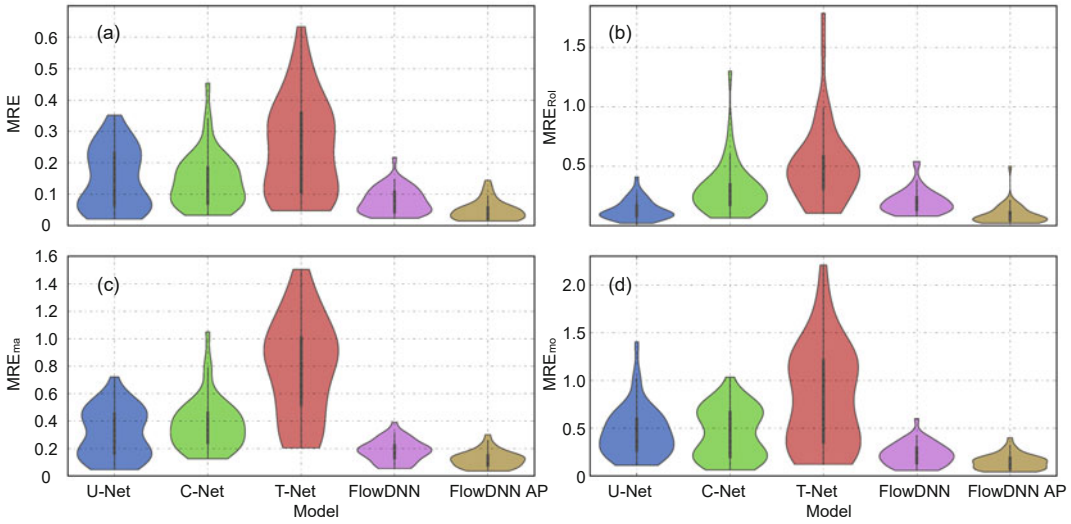


Fig. 5 Distribution for different models on the test dataset: (a) MRE; (b) MRE_{RoI}; (c) MRE_{ma}; (d) MRE_{mo}

5.3 Attention mechanisms

Fig. 7a shows the change of the validation loss for FlowDNN with and without AM. We can see that the model with AM quickly converges within 50 epochs and achieves a smaller overall validation loss. The results suggest that AM can boost FlowDNN in terms of accuracy and training efficiency.

The violin diagram in Fig. 7b shows the distribution of MRE_{RoI} on the test dataset. Here, the shape of the violin indicates the data distribution, and the thick black line shows where half of the data are located. AM helps reduce MRE_{RoI} from 23.56% to 9.16%. This is because CAM and SAM can improve the ability of the networks in learning boundary

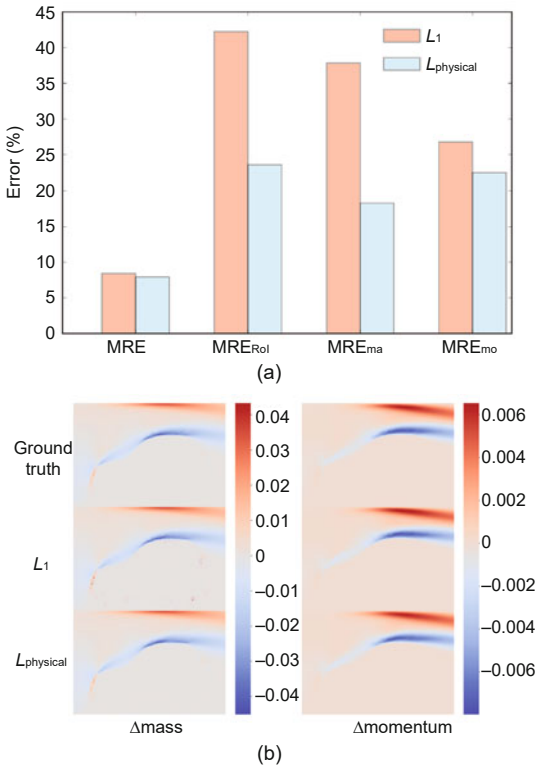


Fig. 6 Comparing MRE, MRE_{RoI}, MRE_{ma}, and MRE_{mo} between L_1 and $L_{physical}$ (a) and Δmass and Δmomentum for different loss functions (b)

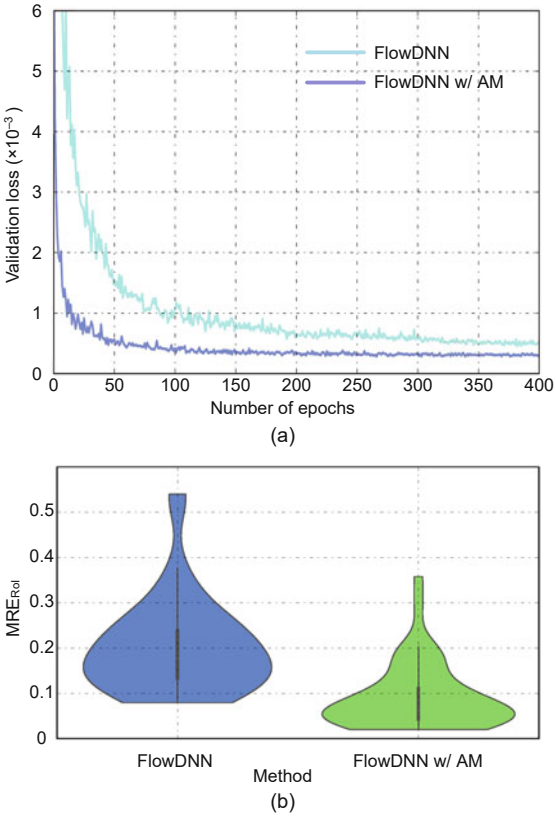


Fig. 7 The validation loss with and without the attention module (AM) (a) and distribution of MRE_{RoI} with and without AM on the test dataset (b)

information by extracting more discriminative features from the channel and spatial domain.

5.4 Activation function

Fig. 8a shows that the ReLU function, combined with batch normalization, converges faster and delivers fewer errors than the ELU function on the validation dataset. Because the ELU function allows the network to push the mean activation closer to zero and thus helps normalization, we argue that the relatively poor performance of the ELU function is due to the repeated normalization. To demystify this, we further experiment on the ELU function without batch normalization. The results show that the ReLU function still yields better performance in terms of accuracy, although the ELU function without batch normalization gains faster convergence and more stable results than before. We can conclude that batch normalization does not always boost the performance of neural networks. We do additional experiments comparing the ReLU function and its variants like the leaky ReLU and PReLU functions. However, we find that differences between these variants are relatively small. Considering that the ReLU function can increase the sparsity of the neural network and hence help network pruning, we choose ReLU as the activation function.

5.5 Network pruning

Fig. 8b shows how the model scores with network pruning. MRE fluctuates up and down in a small range when the number of pruned filters is smaller than about 500. As pruning continues, the damage to the network structure is beyond tolerance, so the prediction error rises sharply. Therefore, the model achieving the smallest prediction error is determined as the final predictive model.

6 Conclusions

We have presented FlowDNN, a novel DNN-based framework for predicting steady flow fields. FlowDNN is designed to speed up full-order CFD simulations while preserving the physical

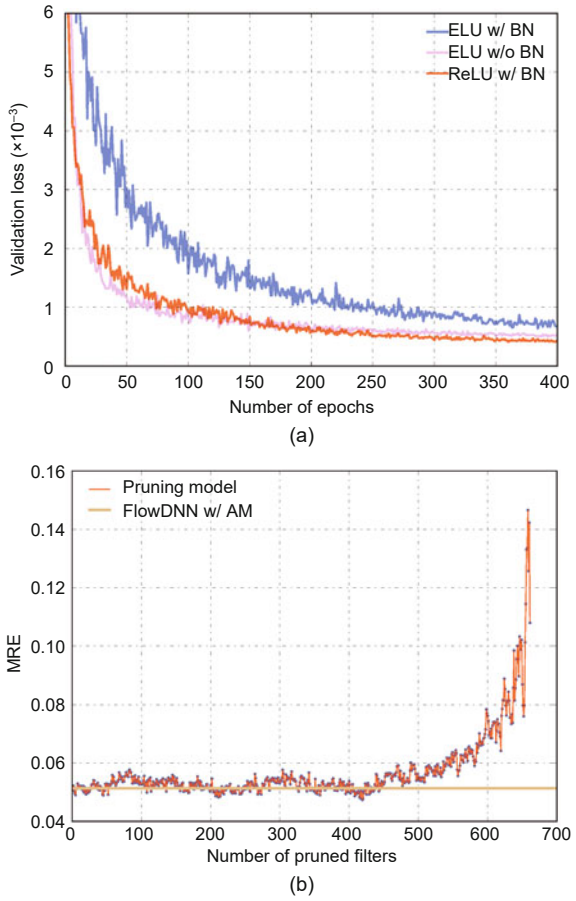


Fig. 8 The impact of ReLU and ELU activation functions (a) and MRE of velocity with and without pruning (b) (ELU: exponential linear unit; ReLU: rectified linear unit)

conservation laws. Unlike prior work, FlowDNN employs attention mechanisms to learn better from the boundary layers. Experimental results show that FlowDNN significantly outperforms prior CFD approximation methods by delivering faster inference and more accurate prediction results. It speeds up a GPU-accelerated CFD solver by more than 14 000×.

Contributors

Donglin CHEN and Xiang GAO designed the research. Siqi WANG and Shizhao CHEN processed the data. Chuanfu XU drafted the paper. Jianbin FANG and Zheng WANG helped organize the paper. Donglin CHEN and Xiang GAO revised and finalized the paper.

Compliance with ethics guidelines

Donglin CHEN, Xiang GAO, Chuanfu XU, Siqi WANG, Shizhao CHEN, Jianbin FANG, and Zheng WANG declare that they have no conflict of interest.

References

Ahmed MYM, Qin N, 2009. Surrogate-based aerodynamic design optimization: use of surrogates in aerodynamic design optimization. Int Conf on Aerospace Sciences & Aviation Technology, p.1-26. https://doi.org/10.21608/ASAT.2009.23442

Amodio M, Krishnaswamy S, 2019. TraVeLGAN: image-to-image translation by transformation vector learning. IEEE/CVF Conf on Computer Vision and Pattern Recognition, p.8975-8984. https://doi.org/10.1109/CVPR.2019.00919

Balabanov VO, Giunta AA, Golovidov O, et al., 1999. Reasonable design space approach to response surface approximation. *J Aircr*, 36(1):308-315. https://doi.org/10.2514/2.2438

Bhatnagar S, Afshar Y, Pan S, et al., 2019. Prediction of aerodynamic flow fields using convolutional neural networks. *Comput Mech*, 64(2):525-545. https://doi.org/10.1007/s00466-019-01740-0

Blazek J, 2015. Computational Fluid Dynamics: Principles and Applications (3rd Ed.). Butterworth-Heinemann, Oxford, UK, p.466.

Constantin P, Foias C, 1988. Navier–Stokes Equations. The University of Chicago Press, Chicago, IL, USA, p.199.

Daberkow DD, Mavris DN, 1998. New approaches to conceptual and preliminary aircraft design: a comparative assessment of a neural network formulation and a response surface methodology. World Aviation Congress & Exposition, Article 15. https://doi.org/10.4271/985509

Ernst MH, 1981. Nonlinear model-Boltzmann equations and exact solutions. *Phys Rep*, 78(1):1-171. https://doi.org/10.1016/0370-1573(81)90002-8

Farrashkhalvat M, Miles JP, 2003. Basic Structured Grid Generation: with an Introduction to Unstructured Grid Generation. Elsevier, Amsterdam, the Netherlands, p.190-226. https://doi.org/10.1016/B978-075065058-8/50008-3

Frankle J, Carbin M, 2019. The lottery ticket hypothesis: finding sparse, trainable neural networks. https://arxiv.org/abs/1803.03635v5

Geneva N, Zabaras N, 2019. Quantifying model form uncertainty in Reynolds-averaged turbulence models with Bayesian deep neural networks. *J Comput Phys*, 383:125-147. https://doi.org/10.1016/j.jcp.2019.01.021

Guastoni L, Guemes A, Ianiro A, et al., 2020. Convolutional-network models to predict wall-bounded turbulence from wall quantities. https://arxiv.org/abs/2006.12483

Guo XX, Li W, Iorio F, 2016. Convolutional neural networks for steady flow approximation. Proc 22nd ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining, p.481-490. https://doi.org/10.1145/2939672.2939738

Hamdan MKA, Rover DT, Darr MJ, et al., 2019. Mass estimation from images using deep neural network and sparse ground truth. http://arxiv.org/abs/1908.04387

Hu J, Shen L, Sun G, 2018. Squeeze-and-excitation networks. IEEE/CVF Conf on Computer Vision and Pattern Recognition, p.7132-7141. https://doi.org/10.1109/CVPR.2018.00745

Isola P, Zhu JY, Zhou TH, et al., 2017. Image-to-image translation with conditional adversarial networks. IEEE Conf on Computer Vision and Pattern Recognition, p.5967-5976. https://doi.org/10.1109/CVPR.2017.632

Kim T, Cha M, Kim H, et al., 2017. Learning to discover cross-domain relations with generative adversarial networks. Proc 34th Int Conf on Machine Learning, p.1857-1865.

Lee S, You D, 2019. Data-driven prediction of unsteady flow over a circular cylinder using deep learning. *J Fluid Mech*, 879:217-254. https://doi.org/10.1017/jfm.2019.700

Li DL, Xu CF, Wang YX, et al., 2016. Parallelizing and optimizing large-scale 3D multi-phase flow simulations on the Tianhe-2 supercomputer. *Concurr Comput*, 28(5):1678-1692. https://doi.org/10.1002/cpe.3717

Ling JL, Kurzawski A, Templeton J, 2016. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *J Fluid Mech*, 807:155-166. https://doi.org/10.1017/jfm.2016.615

Liu Z, Sun MJ, Zhou TH, et al., 2019. Rethinking the value of network pruning. https://arxiv.org/abs/1810.05270

Long J, Shelhamer E, Darrell T, 2015. Fully convolutional networks for semantic segmentation. IEEE Conf on Computer Vision and Pattern Recognition, p.3431-3440. https://doi.org/10.1109/CVPR.2015.7298965

Molchanov P, Tyree S, Karras T, et al., 2017. Pruning convolutional neural networks for resource efficient inference. Int Conf on Learning Representations.

Odena A, Dumoulin V, Olah C, 2016. Deconvolution and checkerboard artifacts. *Distill*, 1(10):e3. https://doi.org/10.23915/distill.00003

Park J, Woo S, Lee JY, et al., 2018. BAM: bottleneck attention module. https://arxiv.org/abs/1807.06514v1

Raissi M, Perdikaris P, Karniadakis GE, 2017. Physics informed deep learning (part I): data-driven solutions of nonlinear partial differential equations. https://arxiv.org/abs/1711.10561

Raissi M, Perdikaris P, Karniadakis GE, 2019. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J Comput Phys*, 378:686-707. https://doi.org/10.1016/j.jcp.2018.10.045

Ronneberger O, Fischer P, Brox T, 2015. U-Net: convolutional networks for biomedical image segmentation. Medical Image Computing and Computer-Assisted Intervention, p.234-241. https://doi.org/10.1007/978-3-319-24574-4_28

Srinivasan PA, Guastoni L, Azizpour H, et al., 2019. Predictions of turbulent shear flows using deep neural networks. *Phys Rev Fluids*, 4:054603. https://doi.org/10.1103/PhysRevFluids.4.054603

Thuerey N, Weissenow K, Prantl L, et al., 2020. Deep learning methods for Reynolds-averaged Navier-ÍCStokes simulations of airfoil flows. *AIAA J*, 58(1):25-36. https://doi.org/10.2514/1.J058291

Wang R, Kashinath K, Mustafa M, et al., 2020. Towards physics-informed deep learning for turbulent flow prediction. Proc 26th ACM SIGKDD Int Conf on Knowledge Discovery & Data Mining, p.1457-1466. https://doi.org/10.1145/3394486.3403198

Woo S, Park J, Lee JY, et al., 2018. CBAM: convolutional block attention module. European Conf on Computer Vision, p.3-9. https://doi.org/10.1007/978-3-030-01234-2_1

Zhou ZW, Siddiquee MMR, Tajbakhsh N, et al., 2018. UNet++: a nested U-Net architecture for medical image segmentation. 4th Int Workshop on Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support, p.3-11. https://doi.org/10.1007/978-3-030-00889-5_1

Zhu JY, Park T, Isola P, et al., 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. IEEE Int Conf on Computer Vision, p.2242-2251. https://doi.org/10.1109/ICCV.2017.244