TUM School of Engineering and Design

# Physics field prediction using convolutional neural networks

Hao Ma

Vollständiger Abdruck der von der TUM School of Engineering and Design der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

genehmigten Dissertation.

| | |
|---|---|
| Vorsitz: | Prof. Dr.-Ing. Agnes Jocher |
| Prüfer der Dissertation: | 1. Prof. Dr.-Ing. Oskar J. Haidn |
| | 2. Priv.-Doz. Dr.-Ing. habil. Xiangyu Hu |
| | 3. Prof. Dr. Nils Thuerey |

Die Dissertation wurde am 04.10.2021 bei der Technischen Universität München eingereicht und durch die TUM School of Engineering and Design am 10.03.2022 angenommen.

# Declaration of Authorship

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Hao Ma
May 16, 2022


hao.ma@tum.de



Released May 16, 2022
Typesetting LaTeX

*To my family*
*To the human being*

*Hao Ma*

# Vier Jahre in meinem Leben

# 人生中四年

# Abstract

The previous research shows that the Computational Fluid Dynamics (CFD) simulation is able to acquire the accurate solutions of the physics field. However, the traditional discrete methods used in the numerical simulation are expensive and in some practical problems, they still remain great challenges. In this decade, machine learning has developed rapidly. From the initial computer vision field where so much research has been conducted, machine learning, especially the deep learning approach, has been applied to many scientific research fields, among which the physics prediction is an attractive choice.

This cumulative dissertation is devoted to predicting the physics field solutions in liquid rocket engines, where both fundamental scientific and practical engineering problems are involved, using the Convolutional Neural Networks (CNN) method. In particular, a data-driven CNN approach is developed for the characterization of film cooling in a combustor, a physics-driven method is proposed for field solution prediction such as steady-state heat conduction and fluid mechanics, a combined data-driven and physics-driven method is proposed using both reference samples and physics law in loss, a generative adversarial networks with physical evaluators framework is designed to simulate the instantaneous spray of a pintle injector.

The first part of this thesis contributes to introducing one convolutional neural networks architecture to directly predict the mixing characteristics between coolant and hot gas in a rocket combustion chamber. Based on a reference experiment, numerical solutions are obtained from a Reynolds-Averaged Navier–Stokes simulation campaign. And then the numerical results on finite volumes are interpolated on a rectangular finer grid for CNN. A U-net architecture is modified to encode and decode features of the mixing flow field. The influence of training data size and learning time with both normal and re-convolutional loss functions is illustrated. By conducting numerical experiments about test cases, the modified architecture and related learning settings are demonstrated with global errors less than 0.55%.

The second part of this thesis chooses the heat conduction problem as an example and compared the data- and physics-driven learning process with deep CNN. It shows that the convergence of the error towards a ground truth solution and the residual of the heat conduction equation exhibits remarkable differences. Based on this observation, a combined data- and physics-driven method for learning acceleration and more accurate solutions is proposed. With a weighted loss function, reference data and the physical equation are able to simultaneously drive the learning. Several numerical experiments are conducted to investigate the effectiveness of the combined method. For the data-driven based method, the introduction of the physical equation not only is able to speed up the convergence but also produces physically more consistent solutions. For the physics-driven based method, it is observed that the combined method is able to speed up the convergence up to about 50% even the target values are not very precise, which allows to incorporate data uncertainty matters.

The third part of this thesis targets the physics-driven learning of complex flow field solutions with high resolutions, especially to reduce the dependency on large

amounts of pre-computed training data. The use of CNN with a U-net architecture is able to efficiently represent and reconstruct the input and output fields, respectively. By introducing Navier-Stokes equations and boundary conditions into loss functions, the physics-driven CNN is designed to predict corresponding steady flow fields directly. In particular, this prevents many of the difficulties associated with approaches employing fully connected neural networks. Several numerical experiments are conducted to investigate the behavior of the CNN approach, and the results indicate that a first-order accuracy has been achieved. Specifically for the case of the flow around a cylinder, different flow regimes can be learned and the adhered “twin-vortices" are predicted correctly. The numerical results also show that the training for multiple cases is accelerated significantly, especially for the difficult cases at low Reynolds numbers, when limited reference solutions are used as supplementary learning targets.

The fourth part of this thesis aims at a pintle injector with adjustable geometry in liquid rocket engines. A novel deep learning approach used to simulate instantaneous spray fields under continuous operating conditions is explored. Based on one specific type of neural networks and the idea of physics constraint, a Generative Adversarial Networks with Physics Evaluators (GAN-PE) framework is proposed. The geometry design and mass flux information are embedded as inputs. After the adversarial training between the generator and discriminator, the generated field solutions are fed into the two physics evaluators. In this framework, a mass conversation evaluator is designed to improve the training robustness and convergence. And the spray angle evaluator, which is composed of a down-sampling CNN and a theoretical model, compares the generated spray angle with the reliable one according to injection conditions. The characteristics of the simulated spray, including the spray morphology, droplet distribution, and spray angle, are well predicted. The work suggests the great potential for prior physics knowledge employment in the simulation of instantaneous flow fields.

# Acknowledgements

First and foremost, I would like to express my sincere gratitude to Prof. Oskar J. Haidn for his guidance, patience, encouragement, and suggestions through my Ph.D. period. I would never forget the kindness from the other end of the line when I was requesting an invitation letter with reverence and unease. The free environment in our chair is the premise of my involvement with desired research topics. I am grateful to PD Dr. Xiangyu Hu for his guidance in this work. What he provides is much more than a typical mentor's job, including but not limited to novel ideas, beneficial discussions, and systematic paper writing advice. He is a true researcher dedicated to the advancement of science. I'd like to express my gratitude to Prof. Nils Thuerey who has been extremely helpful in offering feedback on any research progress through regular group meetings and email exchanges, revising my articles meticulously, and acquainting me to the machine learning society. I also want to thank Dr. Yuxuan Zhang for his assistance. He is the one who leads me to discover this unique and fascinating area when I was still clueless about artificial intelligence.

I had a fantastic time as a Ph.D. candidate at the Chair of Space Propulsion, Technical University of Munich. Dr. Christoph Kirchberger, Dr. Meng Luo, Dr. Ye Hong, Zhendong Yu, Julian Pauw, Zijie Li, Nikolaos Perakis, Andrej Sternin, Fernanda Winter, Jianing Liu, Hongxin Wang, Christoph von Sethe, Dr. Silong Zhang, Yongchuan Yu, Dr. Jingying Zuo, Prof. Volker Gümmer, Mattia Straccia, and Ilaria De Dominicis are among my colleagues who have shown strong passions into scientific research and inspired my Ph.D. studies. It is always a pleasure to work with my collaborators in Hu group, including Dr. Chi Zhang, Dr. Massoud Rezavand, Yujie Zhu, and Thuerey group, including Dr. Liwei Chen and You Xie.

My gratitude also goes to many researchers around the world, especially to Prof. George Karniadakis, Prof. Bernd R. Noack, Prof. Weiwei Zhang. As the forerunners of this specific research field, they still provide feedback to the queries via conferences and online approaches. Thanks for the discussions with my friends Botao Zhang, Wei Wang, Xiaobin Hu, Zhengkui Wang, Lujun Li, Shiyu Li, and Yang Sun. The advice inspired this work a lot.

I am deeply indebted to the Chinese Scholarship Council for providing the financial support for my research in Germany. Grateful acknowledgment also goes to the National University of Defense Technology and the Dalian University of Technology. Only to have them as the origin can I have the opportunity to see a wider world.

Finally, but certainly not least, I want to express my gratitude and love for my family. My parents are always there to back me up. My sister's family assists me in sharing the burden of responsibility. To my beloved wife Yabo Song, the happiness from your company is a source of strength facing frustrations. It is lucky to have you in these two Chinese-zodiac-cycle student life. Now let's move on to more cycles.

# Contents

# List of Symbols

| | |
|---|---|
| $CFD$ | computational fluid dynamics |
| $HPC$ | high-performance computing |
| $N-S$ | Navier–Stokes |
| $POD$ | proper orthogonal decomposition |
| $DMD$ | dynamic mode decomposition |
| $AI$ | artificial intelligence |
| $ML$ | machine learning |
| $ANN$ | artificial neural networks |
| $NN$ | neural networks |
| $SVM$ | support vector machine |
| $GPU$ | graphics processing unit |
| $MPP$ | massively parallel processing |
| $PDE$ | partial differential equations |
| $RANS$ | Reynolds-averaged Navier–Stokes |
| $CNN$ | convolutional neural networks |
| $CV$ | computer vision |
| $GAN$ | generative adversarial networks |
| $MLP$ | multilayer perceptron |
| $PINN$ | physics-informed neural networks |
| $PD-CNN$ | physics-driven convolutional neural networks |
| $GAN-PE$ | generative adversarial networks with physical evaluators |
| $BGD$ | batch gradient descent |
| $SGD$ | stochastic gradient descent |
| $MBGD$ | mini-batch gradient descent |
| $Adam$ | adaptive moment estimation |
| $MAE$ | mean absolute error |
| $MSE$ | mean square error |
| $CIFAR$ | canadian institute for advanced research data set |
| $MNIST$ | modified national institute of standards and technology database |
| $LSTM$ | long-short term memory |
| $AIOps$ | artificial intelligence for information technology operations |
| $K$ | kernel |
| $k$ | convolutional kernel size |
| $c$ | channel factor |
| $s$ | stride |
| $T$ | temperature |
| $p$ | pressure |
| $\mu$ | viscosity |
| $\rho$ | mass density |
| $\mathbf{u}$ | velocity vector |

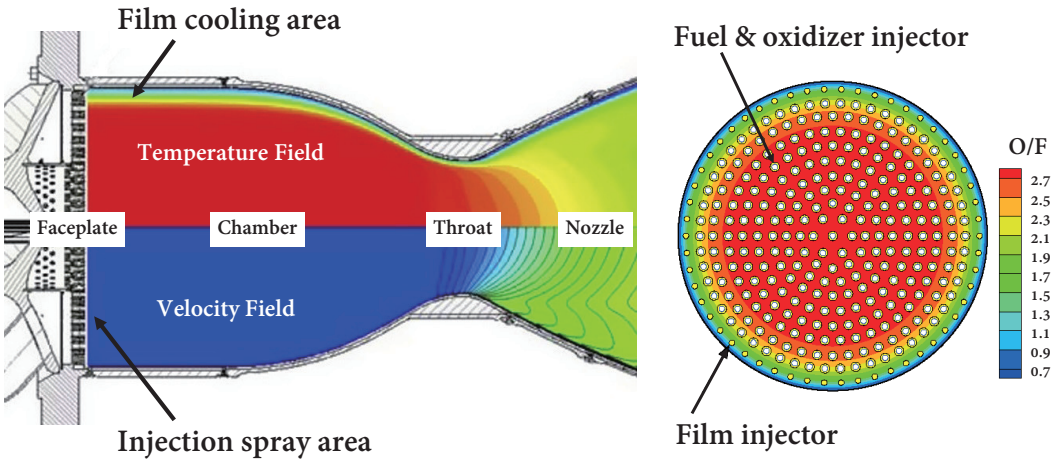
# Chapter 1

# Introduction

## 1.1 Traditional methods for physics field prediction

Future generations of space transportation systems require flexible launch capabilities and reusability. For the foreseeable future, they will rely on chemical propulsion systems as primary engines, as this type of propulsion offers the best compromise between development and efficiency. Particularly, civilian launchers and spacecraft rely on liquid-propellant engines. In terms of cost, efficiency, reliability, and environmental compatibility, this type of engine needs more technological improvement to reach the highest payload while keeping aerodynamic and thermal loads at reasonable levels [1]. As a precondition for novel technical solutions, extensive basic study of the enormous thermal and mechanical loads of liquid engines are necessary.

There are two key technological challenges in the development of liquid rocket engines: film cooling and injection spray, as illustrated in Figure 1.1. They are essential for launcher systems to operate efficiently and safely. The former is a thermal protection technique that can be used in combustion chambers by introducing a portion of fuel along the wall [2]. This introduction in the combustion chamber not only protects the wall from high thermal loads, but also from chemical impact. The latter is the injection spray. Because transient start-up is one of the most challenging operations for a rocket engine, and the combustion performance of the propellant combination is strongly reliant on the injection and mixing procedures, it is critical to comprehend the spray phenomena involved.




(A) Engine configuration and physics field

(B) Injector face plate

FIGURE 1.1: Film cooling and injection spray in a liquid rocket thrust chamber (adapted from Ref. [3]).

For the aforementioned practical engineering tasks, obtaining the physics field is an important aspect in order to obtain dominating parameters. These parameters and other obtained characteristics are crucial for engine design such as a blowing ratio, injector distribution, and nozzle geometry, etc. The physics field is traditionally obtained by solving a sequence of governing equations on a computational mesh with proper boundary conditions. One of the approaches is known as *Computational Fluid Dynamics* (CFD) simulation. It bases on the fact that physical phenomena can be described by partial differential equations, such as the Laplace equation and *Navier-Stokes* (N-S) equations, which can usually be solved analytically or approximated within small domains of space and time [4]. With the development of the CFD methods and high-performance computing resources, the accuracy of the numerical simulation has been greatly improved. From the early studies which are mainly carried out with experimental techniques [5, 6, 7], the numerical investigations were widely conducted for almost all engine components nowadays.

However, despite the impressive advances in the field of *High-Performance Computing* (HPC), the computational resources needed for complicated full-resolution applications requiring many iterations of physics solutions, such as aerodynamic optimization, are still out of reach. The physical field solving is computationally expensive in many circumstances, slowing down the entire design process. As a result, a method that is both economical and precise is necessary [8].

Besides the discrete methods, there are several dimensionality-reduction strategies for difficult prediction tasks, such as *Proper Orthogonal Decomposition* (POD) or *Dynamic Mode Decomposition* (DMD). By which, the low-rank modes and subspaces that characterize spatial-temporal flow data are computed and interpreted [9, 10]. POD and DMD are based on the singular value decomposition which is widely used in the dimensionality reduction of physical systems. These reduction methods constitute the mathematical foundation of reduced-order modeling and realize the computation of high-dimensional discretizations of complicated processes [11]. However, the employment of the reduced-order modeling to compress and reconstruct the physics field not only is complicated but also may introduce additional errors from the projection onto reduced space [12, 13, 14].

Since the development of computational tools, researchers have employed CFD approaches extensively for rocket engine design and analysis. As a result, the amount of data gathered from numerical simulations has increased dramatically. Consequently, new methods for processing and analyzing these data are needed. On the other hand, data science has grown significantly in recent years, resulting in the dramatic blossom of *Machine Learning* (ML) techniques for big data. As a result of the advancement of effective ML algorithms, data-driven modeling has arisen as a new modeling pattern for physics prediction.

## 1.2 Machine learning

Over the last decade, machine learning-based approaches have become increasingly popular and affect a wide range of industries, including autonomous driving, health care, banking, manufacturing, and more. Like computers and information technology at the turn of the century, machine learning is frequently recognized as one of the most cutting-edge technologies of our era. Machine learning's overall goal is to find patterns in data that can be used to guide how problems are solved. In an autonomous car, for example, massive amount of data from sensors must be turned

into judgments on how to manage the car in a very intricate environment. The inside machine should has learned to recognize danger and make decision. Major advances in the aforementioned range, such as the objective recognition, have emphasized machine learning's recent success.

In particular, deep learning technology, to a significant extent, has recently allowed machines to access applications that were previously unavailable. AlphaGo's demonstration of deep reinforcement learning, for example, has had a substantial impact on the perception that the whole *Artificial Intelligence* (AI) field is moving closer to what was expected. Among the advanced methods, machine learning, deep learning, and neural networks are examples of artificial intelligence subfields. Deep learning, on the other hand, is a branch of machine learning, and neural networks is a branch of deep learning.

*Artificial Neural Networks* (ANN) or simply *Neural Networks* (NN), are a core subset of the current machine learning methods. The name and structure are inspired by the human brain, and they function similarly to biological neurons in terms of communication.

In the middle 1980s, neural networks were extremely popular due to their parallel and distributed processing capabilities. However, the lack of availability of backpropagation training, which is often used to alter the parameters, has hampered advancement in this field. At that time, neural networks have been progressively replaced by *Support Vector Machines* (SVM) [15] and other simple models that can be easily trained to address convex optimization problems. Over the past decade, along with the increased computational capabilities, such as *Graphics Processing Unit* (GPU) and *Massively Parallel Processing* (MPP), the use of neural networks has a resurgence. Meanwhile, the proposal of ResNet [16] helps deep neural networks gaining a lot of success in supervised learning and the networks with hundreds of layers have been developed to replace the shallow neural networks. When it comes to speech and image recognition, deep learning performs as well as, or even better than, the human being. Also, when utilized for unsupervised learning tasks like feature extraction, deep neural networks have more powerful capability to extract information from raw audio or images with considerably little interaction. In recent years, some novel and better training procedures such as unsupervised pre-training and automated machine learning have sparked new interest in neural networks.

As Figure 1.2 shows, NN is comprised of some node layers, containing an input layer, one or more hidden layers, and an output layer. In the numerous deep learning tasks, the output layer defines the task form. For example, when the output layer is a categorical variable, the neural network is a way to address classification problems. When the output layer is a continuous variable, the network can be used to conduct regression tasks. When the output layer is the same as the input layer, the network can be used to extract intrinsic features. Each node, or artificial neuron, is connected to the others and has a weight and threshold linked. If a node's output exceeds a certain threshold value, the node is activated, and data is sent to the next layer of the network. Otherwise, no data is sent. Because data is transmitted from one layer to the next, this neural network is referred to as a feedforward network.

The layers are made of nodes that are illustrated in Figure 1.3. A node is a computational unit modeling a neuron in the human brain that will activate when it gets enough stimuli. A node combines data input with a set of coefficients, or weights, that either amplify or dampen that input, providing relevance to inputs for the algorithm's learning purpose. Larger weights indicate that some variables are more critical to the choice or result. The inputs are then multiplied by their corresponding

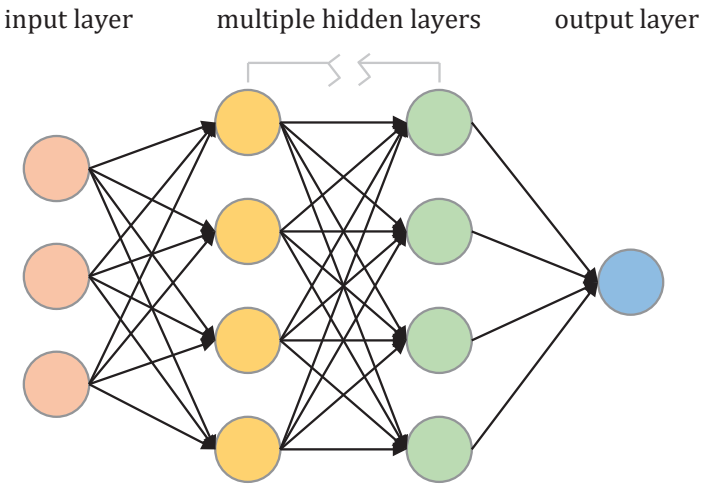



FIGURE 1.2: A neural network consists of three parts: input layer, hidden layers, and output layer. The number of hidden layers defines the model complexity and modeling capacity.

weights before being added together. The sum is then passed to the activation function of a node, which determines whether and how far the signal should go across the network to accomplish the desired goal. The node activates when the sum exceeds a specified threshold, transferring data to the next layer of the network. One node's output becomes the input of the next node. The model would generate different results if the weights or the threshold were altered. As a result, a neural network might make increasingly sophisticated judgments depending on the output of previous layers.

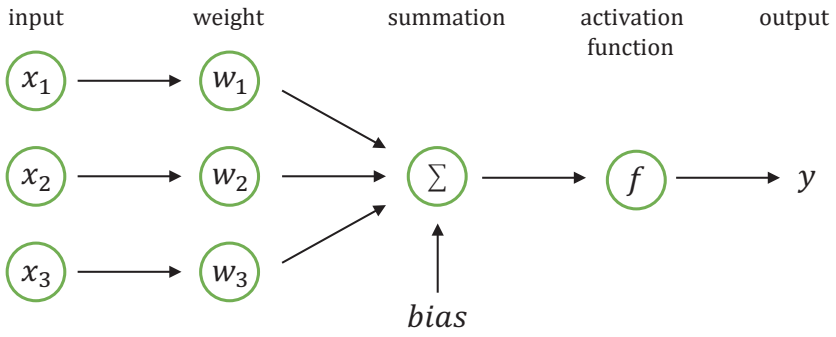



FIGURE 1.3: Diagram of a node in layers. Each individual node has its own regression model, composed of the input data, weights, a bias, a activation function, and an output.

## 1.3 Machine learning for physics prediction

Parallel to the rise of machine learning techniques in industrial applications, scientists, notable physicists, have become more interested in machine learning's potential in fundamental research. This is unsurprising to some extent, considering that ML and physics both utilize comparable methods and have similar goals. Both professions are interested in gathering and interpreting data in order to develop models

that can predict how complex systems will behave. Though the core goals are fulfilled, the ways are noticeably different. On the one hand, physicists strive to understand nature's systems and employ their own knowledge, intelligence, and intuition into the models. Machine learning, on the other hand, does the inverse: models are agnostic, and the machine draws "intelligence" from data. The obtained models are opaque to humans' understanding. But in certain contexts, they provide remarkably good results [17].

With abundant training methods and high-performance computing resources, machine learning has been applied for many scientific research fields, including computational physics where modeling [18, 19], optimization [20, 21], control [22] and other critical tasks [23] have been carried out. A specific application of machine learning is to predict the physics field for reducing or avoiding the large computational cost of the traditional discrete, finite volume/element/difference, methods which solve *Partial Differential Equations* (PDEs) numerically.

Machine learning methods for physics field prediction can be classified into two distinct types: data-driven relying on training data and physics-driven using physics law. In previous research, usually, a large amount of labeled training data are used to train the model, which is called the data-driven method. The modern data-driven methods can be roughly identified as the direct way using neural networks and the indirect way using closure models. In fact, in order to reduce the restriction of training data, the physics law presented as PDEs, which is unknown in the data-driven learning methods, could be explicitly employed in the machine learning process [24, 25]. Raissi introduced this decades-old idea into actual machine learning algorithms and named it as *Physics Informed Neural Networks* (PINN), which were presented as two distinct types. For solving high-dimensional inverse problems, scatted and noisy training data were utilized to train the PINN and eventually acquire accurate coefficients of PDE [26]. This type of PINN is actually still a kind of data-driven method. By substitution of training data values into the integer-order PDEs, the neural networks are able to obtain the physics information. Similar works were shown in Ref. [27, 28]. In the other type, by constraining nonlinear PDEs into the loss function, PINN is able to obtain the solution corresponding with the initial and boundary training samples. The effectiveness of the PDE-driving-learning framework was demonstrated through a collection of physical solutions [26, 29, 30]. This kind of physics-constrained, data-free learning method appeared only recently and I name this machine learning approach as the physics-driven method.

Machine learning has presented good characteristics to be an alternative to carry out the physics field prediction compared with traditional discrete methods [31]. Because of the relatively long-term development, the data-driven method has been applied in some practical physics field inference. However, in some difficult cases, though a large number of training samples are input, data-driven methods still have difficulties to obtain solutions which accurately obey the underlying physical laws. Choosing the work in Ref. [32] as an example, though the number of training samples is 12k, the errors still manifest themselves in the inferred shapes of the flow structure behind the airfoil, and the outputs cannot fully reproduce the real fluid field because of lacking new information which the model extracts from the existing training data set. A similar phenomenon also happens in Ref. [33]. It requires more research to eliminate this lack rather than simply utilizing an even larger training data set. For the physics-driven method, it is a new machine learning approach to conduct the physics field prediction work without any training data, which has a broad prospect but still lacks research about the learning process and performance. Meanwhile, although the machine learning practice presented above has shown the

capability to capture the characteristics of the basic fluid mechanics, the application of the machine learning methods in practical fluid dynamical problems is rare, especially for liquid rocket engines.

## 1.4 Aims and objectives

The main objective of the present thesis is to explore and validate the potential of the *Convolutional Neural Networks* (CNN) method for physics field prediction in liquid rocket engines. In practical liquid rocket design work, film cooling is an effective cooling approach for wall protection, and the spray pattern of an injector significantly influences the combustion efficiency. In addition, heat conduction and fluid mechanics are the two key fundamental scientific problems involved. As shown in Figure 1.4, for the two technical challenges and two scientific problems, the thesis aims to propose a systematic framework which consists of the four parts listed below.

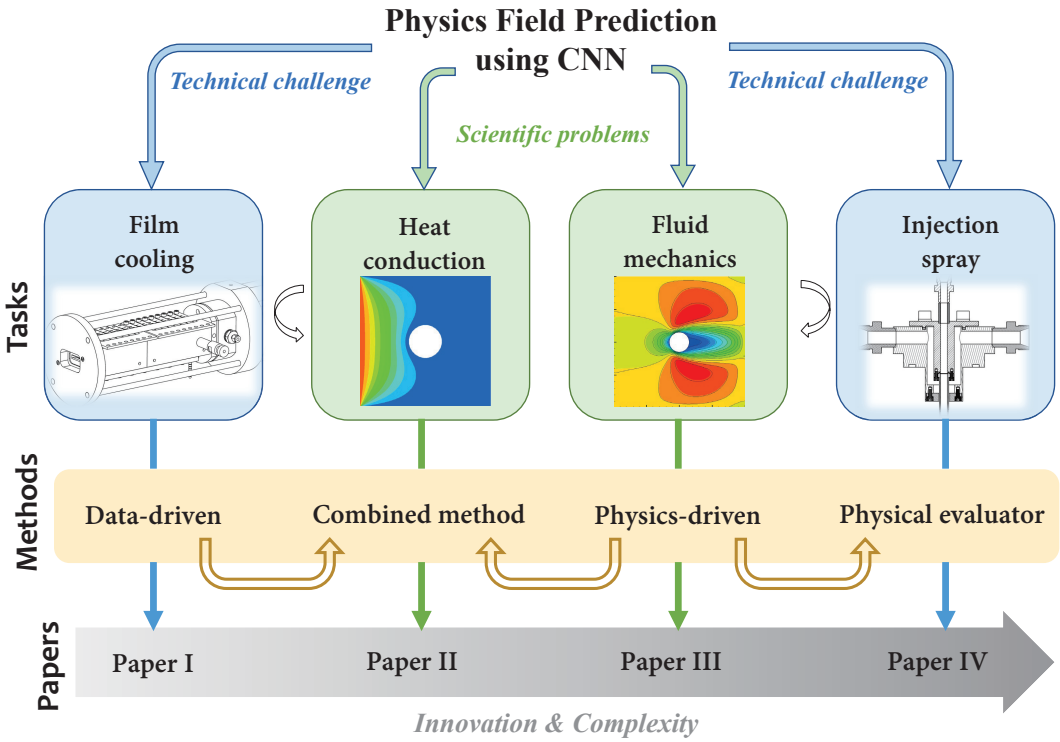



FIGURE 1.4: The framework of this cumulative dissertation.

Concerning the simulation of fluid dynamics in a rocket combustor, in particular the mixing characteristics between coolant film and hot gas, a data-driven framework for flow field prediction based on U-net CNN is presented. The key idea is to modify a U-net architecture that encodes and decodes features of the mixing flow field. In the proposed method, given the inlet flow conditions, the flow field in the subscale gaseous methane/oxygen combustion chamber can be directly predicted. The influence of training data size and learning time with both normal and re-convolutional loss functions is also explored. The proposed method is expected to verify the capability of CNN to capture the characteristics in the practical fluid dynamical problems in liquid rocket engines. This work is detailed in Paper I [34],

- H. Ma, Y. X Zhang, O. J. Haidn, N. Thuerey, X. Y Hu, Supervised learning mixing characteristics of film cooling in a rocket combustor using convolutional neural networks. *Acta Astronautica*. **175**, 11-18, 2020,

which has been attached in Appendix A.1.

The work in Paper I [34] presents the capability of CNN with a data-driven approach. However, in some difficult cases, the data-driven method may still not be able to obtain sufficiently accurate solutions from a large number of training samples. There is a physics-driven method that employing PDEs as the loss function. In order to remedy the shortcomings of the two aforementioned methods, we propose an idea that combines the data- and physics-driven perspectives. Choosing the heat conduction problem which is common in rocket engine analysis work as an example, we first realize the original methods based on a deep CNN respectively and compare their learning progresses for a single case training. A weighted loss function combining the effects of reference target and physics law (given as Laplace equation) is proposed for training the CNN to predict temperature fields. This work is detailed in Paper II [35],

- H. Ma, X. Y. Hu, Y. X. Zhang, N. Thuerey, O. J. Haidn, A combined data-driven and physics-driven method for steady heat conduction prediction using deep convolutional neural networks. *arXiv preprint arXiv:2005.08119*. 2020,

which has been attached in Appendix A.2.

In the physics field, PDEs are usually applied to describe the physical phenomena and predict local quantities such as temperature, pressure, and velocities. In physics-driven NNs, it is postulated that the optical relationship between the local quantities is dominating. However, the previous research on physics-driven method usually focuses on solving simple physics problems using *Multilayer Perceptron* (MLP) which takes a vector as input [36]. Based on the idea of physics constraints and a specific CNN architecture, a *Physics-driven Convolutional Neural Networks* (PD-CNN) method is proposed. In addition, in the physics-driven learning process, the weights of CNN are adapted only to minimize the residuals of PDEs, the solution itself is not constrained, which means there is no target being offered for reference. In order to accelerate the convergence and eventually improve the training performance, besides the physical laws, additional reference targets are provided for constraining the network. This work is detailed in Paper III [37],

- H. Ma, Y. X. Zhang, N. Thuerey, X. Y. Hu, O. J. Haidn, Physics-driven learning of the steady Navier-Stokes equations using deep convolutional neural networks. *arXiv preprint arXiv:2106.09301*. 2021,

which has been attached in Appendix A.3.

After the physics-informed methods research on fundamental scientific problems in Paper II [35] and Paper III [37], the last objective of the thesis is the application of physics-informed CNN on practical engineering problems. Due to the adjustable geometry, pintle injectors are especially suitable for liquid rocket engines which require a wide throttleable range. However, applying the conventional computational fluid dynamics approaches to simulate the complex spray phenomena in the whole range still remains a great challenge. In this work, a novel deep learning approach used to simulate instantaneous spray fields under continuous operating conditions is explored. In this framework, GAN is used to generate the instantaneous spray field. A mass conversation evaluator is designed to improve the training robustness and convergence. And a spray angle evaluator guides the networks

generating the spray solutions that is more consistent with the injecting conditions. This work is detailed in Paper IV [38],

- H. Ma, B. T. Zhang, C. Zhang, O. J. Haidn, Generative adversarial networks with physical evaluators for spray simulation of pintle injector. *AIP Advances*. **11**, 075007, 2021,

which has been attached in Appendix A.4.

## 1.5 Outline

The following is the structure of the remainder of the present thesis. Chapter 2 provides an introduction to the methodology used, including the governing equations and data acquisition, the universal theory and fundamentals of CNN, the specific methods for the two practical technical challenges and two fundamental scientific problems. Particularly, as listed in Chapter 3: data-driven CNN is applied to characterize film cooling flow in a rocket combustor. A combined data- and physics-driven method is proposed for heat conduction prediction. A physics-driven approach is developed and discussed for fluid mechanics problems which obey N-S equations. The *Generative Adversarial Networks with physical evaluators* (GAN-PE) is proposed to explore the potential for prior physics knowledge employment in the instantaneous flow simulation. Finally, Chapter 4 discusses the existing literature and gives recommendations for future work.

# Chapter 2

# Methodology

The aim of this chapter is to introduce the methodology used for solving the two technical challenges and two scientific problems. Emphasis is placed on the approaches which are used to obtain the results shown in Appendix A.1 - A.4. There are two parts in this chapter: the first contains two sections describing the universal methods this destination involves, including the governing equations, training data acquisition approach, and the CNN's fundamentals. The second one includes four sections introducing the specific methods for the different physics field prediction tasks.

## 2.1 Governing equations and data acquisition

### 2.1.1 Governing equations

This subsection is aiming to introduce the basic governing equations used to numerically describe heat conduction, fluid motion, and turbulent flows. These governing equations, represented as the partial differential equations, are employed for the training samples generation in the data-driven approach, or as the loss function in the physics-driven approach.

*Laplace equation*

Based on the law of conservation of energy and Fourier's law of heat conduction, the partial differential equation of heat conduction in three-dimensional Cartesian coordinates can be obtained as

$$\rho c \frac{\partial T}{\partial \tau} = \frac{\partial}{\partial x}(\lambda \frac{\partial T}{\partial x}) + \frac{\partial}{\partial y}(\lambda \frac{\partial T}{\partial y}) + \frac{\partial}{\partial z}(\lambda \frac{\partial T}{\partial z}) + \dot{\Phi}. \tag{2.1}$$

In cases of constant properties, without an inner thermal source, the steady heat conduction governing equation could be written as a two-dimensional Laplace equation

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0, \tag{2.2}$$

which is a typical second-order partial differential equation whose solution is important in many branches of physics.

*Navier–Stokes equations*

In flow field prediction work, the PDEs which describe the motion of viscous fluid substances are Navier-Stokes (N-S) equations. In our study, the steady and incompressible form of N-S equations is chosen as follows:

$$\nabla \cdot \boldsymbol{U} = 0, \tag{2.3}$$

$$\boldsymbol{U} \cdot \nabla \boldsymbol{U} + \nabla P - \mu \nabla^2 \boldsymbol{U} = 0, \tag{2.4}$$

where $\boldsymbol{U} \equiv \boldsymbol{U}(u, v)$, $P$, $\mu$ are velocity, pressure and viscosity respectively. Equation (2.3) is the continuity equation, which imposes the incompressibilities of the fluid. Equation (2.4) is the momentum conservation equation, in which the first term represents the momentum convection, $\nabla P$ the pressure gradient and $\mu \nabla \boldsymbol{U}$ the viscous dissipation.

The 2-dimensional form of momentum Equations is

$$\begin{cases} \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \\ u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} - \mu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = 0 \\ u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} - \mu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) = 0 \end{cases} \tag{2.5}$$

*Reynolds-averaged Navier–Stokes equations*

For the flows of constant-property Newtonian fluids, the instantaneous momentum equation can be written as

$$\frac{\mathrm{D}\boldsymbol{u}}{\mathrm{D}t} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \boldsymbol{u} - \frac{\rho}{\rho_0} g \boldsymbol{k}. \tag{2.6}$$

Since the other terms are linear in $\boldsymbol{u}$ and $p$, the mean of the 3-D momentum equation is

$$\frac{\partial \overline{u}_i}{\partial t} + \frac{\partial\left(\overline{u}_i \overline{u}_j\right)}{\partial x_j} = -\frac{1}{\rho_0}\frac{\partial \overline{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\nu \frac{\partial \overline{u}_i}{\partial x_j} - \overline{u_i' u_j'}\right) - \left[1 - \beta(\overline{T} - T_0)\right] g \delta_{i3}. \tag{2.7}$$

The Reynolds equations and the Navier-Stokes equations are the same, except for the term in the Reynolds stresses which are defined as

$$\tau_{ij}' = -\rho_0 \overline{u_i' u_j'}. \tag{2.8}$$

According to the turbulent-viscosity hypothesis, the Reynolds stresses are given by

$$-\rho_0 \overline{u_i' u_j'} = -\frac{2}{3}\rho \delta_{ij} k + \rho \nu_\mathrm{t}\left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i}\right), \tag{2.9}$$

where $\nu_\mathrm{t}$ is turbulent viscosity.

Given the turbulent viscosity field, Eq. 2.9 provides a most convenient closure to the Reynolds equations. The $k - \varepsilon$ model belongs to the class of two-equation models, in which model transport equations are solved for two turbulence quantities. The specification of the turbulent viscosity as

$$\nu_\mathrm{t} = C_\mu \frac{k^2}{\varepsilon}. \tag{2.10}$$

The $k - \varepsilon$ model is the most widely used complete turbulence model, and it is incorporated in most commercial CFD codes [39]. Just like other turbulence models, the concepts and details of this model evolved over time. Jones and Launder developed the standard $k - \varepsilon$ model [40]. The model transport equation for $k$ is

$$\frac{\partial k}{\partial t} + \overline{u}_i \frac{\partial k}{\partial x_i} = \frac{\partial}{\partial x_i}\left[\left(C_k \frac{k^2}{\varepsilon} + \nu\right)\frac{\partial K}{\partial x_i}\right] + P - \varepsilon, \tag{2.11}$$

where $C_k = 0.09 \sim 0.11$. And $P$ is the production term and can be obtained by

$$P = \nu_t \left(\frac{\partial \overline{u}_i}{\partial x_k} + \frac{\partial \overline{u}_k}{\partial x_i}\right) \frac{\partial \overline{u}_i}{\partial x_k}. \tag{2.12}$$

The model transport equation for $\varepsilon$ is

$$\frac{\partial \varepsilon}{\partial t} + \overline{u}_i \frac{\partial \varepsilon}{\partial x_i} = \frac{\partial}{\partial x_i}\left[\left(C_\varepsilon \frac{k^2}{\varepsilon} + \nu\right)\frac{\partial \varepsilon}{\partial x_i}\right] + C_{\varepsilon 1}\frac{\varepsilon}{k}P - C_{\varepsilon 2}\frac{\varepsilon^2}{k}, \tag{2.13}$$

where $C_\varepsilon = 0.07 \sim 0.09$, $C_{\varepsilon 1} = 1.41 \sim 1.45$,$C_{\varepsilon 2} = 1.91 \sim 1.92$.

### 2.1.2 Data acquisition

The machine learning approach for physics field prediction is a combination of data-driven and physics-driven methods. The former usually requires a large number of samples as training targets. For physics prediction tasks involving fluids mechanics, the training targets are usually from two sources: a traditional numerical simulation and experimental results. These approaches are also employed in the flow field prediction tasks in liquid rocket engines. Here, these two distinct data acquisition approaches for the film cooling flow and injection spray are introduced respectively.

*Numerical simulation for film cooling flow*

The training data used for the deep learning task in Paper I [34] are extracted from the numerical simulation results of the film cooling in a rocket combustor [41]. For the numerical simulation campaign, the mass fluxes of mainstream and coolant film are generated as random values which are beneficial for the following training work. And the temperatures of the coolant are evenly distributed as six values in the range between 200 K and 300 K; the temperature of the mainstream is 3326.54 K, and the amount of simulation cases is 1176.

As the performance of the film cooling far downstream is not as remarkable as that in the vicinity of the applicator, the forepart of the simulation domain was chosen as the learning domain for machine learning work; the length of the learning region is 150 mm, while the width is 6 mm, which is the half of the chamber height. Deep learning research in the machine vision field usually uses square-resolution images to adapt to the convolutional neural network architectures. In order to indicate the characteristics of mixing in both dimensions, the learning domain is resampled onto a rectangular $64 \times 256$ grid to obtain a data set including inputs and targets. The three input channels, including mass fluxes of main flow and coolant and coolant temperature, are rearranged as three initial fields, meanwhile, the simulation results of each case are interpolated into a set of Cartesian grids with the same size of $64 \times 256$. The values in every row of the input initial fields describe the mass fluxes and temperature. In order to precisely describe the geometry characteristic, there is an assumed boundary row in the input grid whose values indicate the interface between the main flow and the coolant film. By this calculation, the values in the mixing shear layer are accurately obtained and the geometry characteristic of the film injecting slot is precisely defined in the grid.

Nondimensionalization is a normal data processing method in the numerical calculation of fluid mechanics by which the features with different properties can be compared. The involved quantities are usually normalized with respect to the magnitude of the flow, i.e., make them dimensionless. For convenience, the corresponding characteristic quantity of inlet flow is used to represent this magnitude.

Thus, the maximum film velocity is chosen to be the characteristic quantity $|\mathrm{v}_i|$. So, the dimensionless velocities can be calculated by $\tilde{u}_0 = u_0/|\mathrm{v}_i|$ and $\tilde{v}_0 = v_0/|\mathrm{v}_i|$. According to the energy equation, the nondimensionalization of pressure could be $\tilde{p}_0 = p_0/(|\mathrm{v}_i|)^2$ in order to remove the quadratic scaling of the values from the target data. The input coolant temperature is dimensionless by $\tilde{T}_\mathrm{c} = T_\mathrm{c}/T_\mathrm{m}$, where $T_\mathrm{m} = 3326.54$ K denotes the temperature of the main flow.

In addition, directly using the pressure as a target is an improper choice. It is the pressure gradient rather than the pressure which is involved in the *Reynolds-averaged Navier–Stokes* (RANS) calculation. If the pressure is used as one of the targets directly, the CNN will map the relationship between inputs and the pressure which is less correlated. Thus, mean pressure is subtracted from pressure solution and defined as $\hat{p}_0 = \tilde{p}_0 - p_\mathrm{mean}$ , where $p_\mathrm{mean}$ denotes the mean pressure of all individual pressure samples. With this removal of mean pressure, the learning performance increase by a factor of ca. 4 according to the previous research [42]. Lastly, the values of the quantities in each channel are normalized to the [-1,1] range in order to minimize errors in the training phase. The maximum absolute values of each quantity are found throughout the entire training set, and then the quantities are divided by these maximum values respectively. Before the quotients are transferred into CNN, both inputs and targets are processed in this way. This pre-processing method can flatten the data space and simplify the training task of the deep neural network, eventually accelerating the convergence.

*Experiments for injection spray*

The training data used for the deep learning task in Paper IV [38] are extracted from the spray experimental results of a pintle injector. As detailed in Appendix A.4, the non-reactive cold experiments were conducted at atmospheric pressure. Dry air is used for axial flows and filtered water for radial flows. A back-lighting photography technique is used for instantaneous spray image visualization. The image acquisition system consists of an *Light-emitting Diode* (LED) light source, a high-speed camera, and a computer. The exposure time is 10 µs and the frame rate is 50k fps.

The resolution of the images is $640 \times 480$. The raw monochrome pictures are processed as 8-bit gray images in which every pixel has a gray value and the range is $0 \sim 255$. So, the images are regarded as the 2-D matrices whose dimensions are $640 \times 480$. Eventually, 35k raw images were captured and 29k of them were used for training and the others for validation.

The spray angles of the time-averaged spray images are manually measured. Since the raw images are all captured in the steady injection stage and have no temporal fluctuations, the measured angle value is unique per operating condition. Note that, the average images and the corresponding manually measured angles are only used to train the spray angle estimator, i.e., the down-sampling CNN in the spray angle evaluator. The raw images are used in the training of GAN-PE.

The resolutions of the instantaneous spray image are $640 \times 480$. In order to reduce the training cost, the images are interpolated to the images with a resolution of $128 \times 128$. While the measured angle values, which represented the nature of the spray phenomenon, are fixed in spite of the image scaling.

*Uncertainty*

The data set obtained above is for training for the two technical challenges. The uncertainty associated with the numerical simulation or experiments will influence the accuracy of the learning models which are based on the training samples. Here, the uncertainty from the settings and measurements are discussed.

In numerical simulations for rocket engines, a common approach is to use the temperature field at one certain moment as the boundary condition. Paper I [34] utilizes the in-house tool "$Ro\dot{q}FITT$" to calculate the continuous temperature and heat flux field [43]. The data used for the calculation is from the separated temperature data measured by thermocouples arranged in the chamber wall. The errors in this process are introduced by the accuracy of the thermocouples. Due to the large thermal gradients in the vicinity of the mainstream boundary, even the small temperature error will be amplified and resulting in a large uncertainty for the heat flux calculation. Besides, the placement of the thermocouples which are influenced by the heat contact will also result in a systematic error [44].

In the spray experiments, there are different error sources. One is the mass flow metering in the experimental operations. The mass flow rates listed in Table I in Paper IV [38] are the typical values. Despite the mass flows of the liquid and gaseous propellants being measured in the steady injection stage, they still have slight variations along with the time promotion. In addition, the measurement of the spray angle will also introduce the error. In order to alleviate this, the spray images obtained in the experiment are post-processed to clarify the spray boundary. By calculating the mean gray value, the average of 10 raw images with the same time interval is used to measure the spray angle manually. Every operation condition has 1k raw images, so, the $100^{\text{th}}$, $200^{\text{th}}$, ..., $900^{\text{th}}$ and $1000^{\text{th}}$ images are averaged.

## 2.2 Theory and fundamentals of CNN

Artificial intelligence is aiming to make considerable strides towards bridging the gap between human and computer capabilities. To realize this ambitious goal, researchers focus on a number of themes in this discipline. *Computer Vision* (CV) is one of the numerous research fields in this category. The goal of this field is to enable machines to see and understand the world in the same way that humans do, and to use this learned knowledge for tasks like image and video recognition, image analysis and classification, and so on. Deep learning breakthroughs in CV have been built and developed throughout time, mostly using one method – the convolutional neural networks. To some extent, the solving physics field task can be regarded as the media generation/reconstruction work which is one of the important sub-fields of CV. Hence, employing CNN for physics field generation tasks became attractive. Some general expressions about the CNN method in this section are adapted from the widely circulated books, lectures, and reviews that are cited in the text.

### 2.2.1 Convolution and motivation

In its most general form, convolution is an operation on two functions of a real-valued argument. There is an example of a new function $s$ providing a smoothed estimate of the position in Ref. [45]:

$$s(t) = \int x(a)w(t-a)da, \tag{2.14}$$

where $x(a)$ denotes a position function which has a single output, $w(t-a)$ is a weighted function, and $a$ is the age of a measurement. This operation is called convolution and the convolution operation is typically denoted with an asterisk:

$$s(t) = (x * w)(t). \tag{2.15}$$

If the $x$ and $w$ are defined only on integer $t$, the discrete convolution can be defined as:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a). \tag{2.16}$$

Here the two-dimensional matrix $I$ is the input and a two-dimensional kernel $K$ is assumed:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n), \tag{2.17}$$

where $i$, $j$, $m$, and $n$ are coordinates. Convolution is commutative and Equation (2.17) could be rewritten as

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n). \tag{2.18}$$

There is an example of convolution applied to a 2-D tensor in Figure2.1.

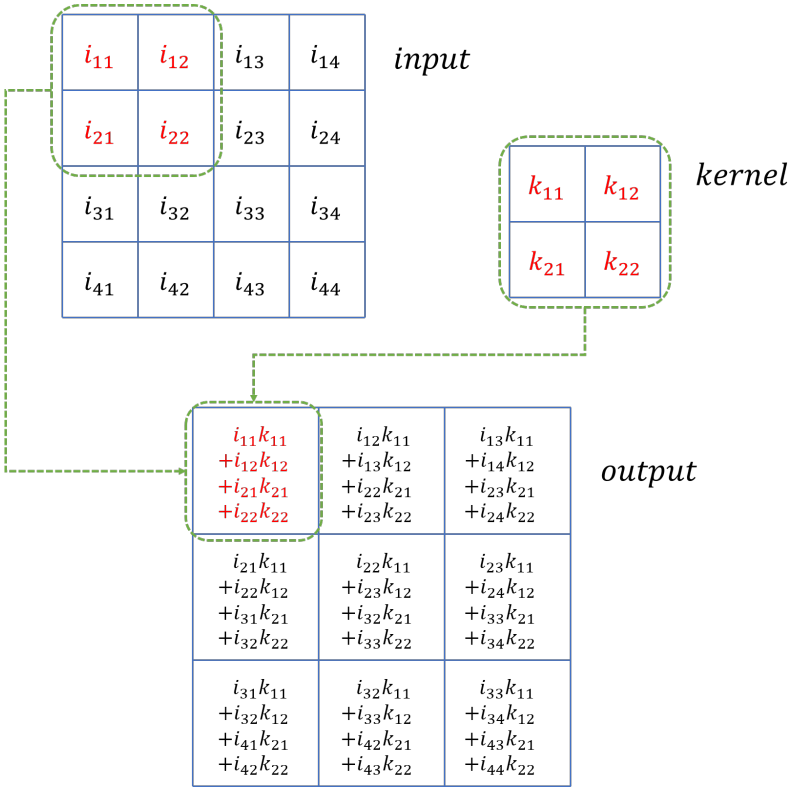



FIGURE 2.1: An example of 2-D convolution. The boxes with arrows indicate how the upper-left element of the output tensor is formed by applying the kernel to the corresponding upper-left region of the input tensor (adapted from Ref. [45]).

CNN's architectures mimic the visual cortex's organization and are similar to the connecting pattern of neurons in the human brain. Individual neurons can only respond to stimuli in the receptive field, a small portion of the visual field. To span the entire visual field, a number of similar fields can be stacked on the top of another. Based on the shared-weight kernels that slide along the whole input fields, CNNs are able to provide translation equivalent responses called feature maps.

Convolutional neural networks are MLP's variations. By utilizing the significant spatially local correlation found in natural images, CNNs minimize the constraints

brought by the MLP design. Compared with MLPs, CNNs have the following characteristics:

3-D volumes of neurons: The neurons in the layers of a CNN are arranged in three dimensions: width, height, and depth [46]. A convolutional layer's receptive field is a region of the layer where each neuron is only coupled to a limited portion of the layer before it. To build a CNN architecture, several layers are stacked, both locally and globally connected, which not only benefits the spatial connection between the positions but also achieves the correlation in the dimension of channel and depth easier.

Local connectivity: Similar to how receptive fields work, CNNs take use of spatial locality by generating a local connection pattern between neurons in nearby layers [47]. The learned filters produce the strongest sensitivity to a spatially constrained input pattern. By stacking many of these layers, non-linear filters become progressively global, allowing the network to first build representations of small parts of the input, then assemble representations of larger areas from them.

In contrast, MLP architecture by itself does not take into account the spatial structure of data. The data points in the learning domain are treated the same way irrespective of their distance and as such their correlation is omitted [48]. For purposes like image reconstruction, which are dominated by spatially input patterns, the complete connection of neurons is thus wasteful. In addition, the quantities of different positions in the physics field, represented as the pixel values in images, have a strong correlation with the adjacent position, while the fully connected neural networks are not able to represent that.

Shared weights: Each filter in a CNN is duplicated throughout the whole visual field. These units with constant values build a feature map with the same parameterization (weights and bias). This means that within their specific response area, all of the neurons in a convolutional layer react to the same feature [49]. In MLP models, the full connection between nodes produces the curse of dimensionality, and higher resolution images become computationally intractable.

For example, in CIFAR-10 data set [50], the size of images are only $32 \times 32 \times 3$ (32 wide, 32 high, 3 color channels), so a single fully connected neuron in the first hidden layer of a regular neural network would have $32 \times 32 \times 3 = 3,072$ weights. A $200 \times 200$ image, however, would lead to neurons that have $200 \times 200 \times 3 = 120,000$ weights. A $1000 \times 1000$-pixel image with RGB color channels has 3 million weights, which is too high to feasibly process efficiently at scale with full connectivity.

LeNet [51] and the other early CNN architectures exploited the underlying basis of the image that the neighboring pixels are correlated to each other and feature motifs are distributed across the entire image [52]. As a result, convolution with learnable parameters is a powerful tool for extracting similar features from several locations with a small number of parameters.

Also, CNN represents a specialized and well-established type of NN to tackle the challenges existing in the conventional physics prediction filed. CNNs have presented a good performance to predict high-fidelity physics solutions. E.g., without an extra reduced-order modeling step, the CNN can directly compress and reconstruct high-fidelity flow fields with a series of convolutional calculations [42]. CNNs succeeded in solving simple physics problems which obey a single PDE, such as Laplace equation [53] and Darcy's law [54], achieving high computational efficiency in capturing multi-scale features of the physics fields. Meanwhile, CNNs also show the capacity to learn spatial connections between the adjacent data points [35], or the long-term control of fluids with physical losses [55].

### 2.2.2 Basic CNN components

*Convolutional layer*

When it comes to the two-dimensional tensors that consist of different channels in the multi-layer convolutional neural networks, the convolution operation can be expressed as follows:

$$s_l^k(p,q) = \sum_c \sum_{x,y} i_c(x,y) \cdot e_l^k(m,n). \tag{2.19}$$

The mathematical symbols used are defined in Table 2.1.

TABLE 2.1: Convolution symbols.

| Symbol | Description |
|---|---|
| $x$ | $x^{\text{th}}$ coordinate under consideration of a tensor |
| $y$ | $y^{\text{th}}$ coordinate under consideration of a tensor |
| $m$ | $m^{\text{th}}$ row under consideration |
| $n$ | $n^{\text{th}}$ column under consideration |
| $c$ | channel index |
| $s_l^k(p,q)$ | $(p,q)$ element of feature matrix |
| $i_c(x,y)$ | $(x,y)$ element of $c^{\text{th}}$ channel of a tensor |
| $e_l^k(m,n)$ | $(m,n)$ element of $k^{\text{th}}$ kernel of $l^{\text{th}}$ |
| $l$ | layer number |

A CNN's main building component is the convolutional layer. The parameters of the layer are made up of a series of learnable convolutional kernels with a narrow receptive field. But the receptive fields span along with the depth. Each filter is convolved across the width and height of the input volume during the forward pass, computing the dot product between the filter and the input to produce a 2-dimensional feature map of that filter. As a consequence, when the network detects a particular sort of feature at a particular spatial location in the input, it will activate the kernel [56].

The whole output volume of the convolution layer is produced by stacking the activation maps for all filters along the depth dimension. Every matrix in the output 3-D tensor can thus be regarded as the output of a neuron that examines a small part of the input and shares parameters with other neurons in the same input. The extent of this connectivity between the kernels and inputs is called the receptive field. The connections are local in space (along width and height) but always extend along with the entire depth of the input volume.

Three hyperparameters control the size of the output volume of the convolutional layer: the depth, stride, and padding size. The depth of the output volume controls the number of neurons in a layer that connects to the same region of the input tensors. Stride determines how depth columns are distributed around the width and height. A larger stride means fewer receptive field overlaps and lower spatial dimensions of output volume [57]. Padding allows modifying the spatial size of the output volume. In particular, it is occasionally advantageous to keep the input volume's spatial size exactly the same, which is referred to as "same" padding. In the CNN architecture used in this dissertation, there is no padding operation and the kernels are able to consider the boundary conditions embedded in the input volumes without noise.

*Fully connected layer*

A fully connected layer is mostly used at the end of the network for classification, for example, the spray angle estimation in Paper IV [38]. Unlike pooling and convolution, it is a global operation that collects data from the feature extraction stages and analyzes the output of all the layers [58]. As a result, it creates a non-linear combination of selected features that are used to classify data [59].

*Pooling layer*

In a CNN's pooling layers, feature maps are divided into rectangular sub-regions, and the features in each rectangle are down-sampled to a single value, usually by taking the average or maximum value. In addition to reducing the size of feature maps, the pooling method provides local translational invariance to the features contained, allowing the CNN to be more resilient to changes in their placements [60].

Pooling can be defined as follows:

$$\mathbf{Z}_l^k = g_p(\mathbf{F}_l^k), \tag{2.20}$$

where the pooling operation in which $\mathbf{Z}_l^k$ represents the pooled feature-map of $l^{\text{th}}$ layer for $k^{\text{th}}$ input feature-map $\mathbf{F}_l^k$, whereas $g_p$ defines the type of pooling operation.

The use of pooling operation helps to extract a combination of features, which are invariant to translational shifts and small distortions [61, 62]. Feature motifs, which appear as a result of the convolution technique, can appear in the image at various positions. Once features have been extracted, their precise location becomes less crucial as long as their relative position to others is kept. Pooling is a fascinating local activity and it gathers similar data in the immediate vicinity of the receptive field and produces the dominant response in this area [63].

Decreasing the size of the feature map to an invariant feature set not only regulates the network's complexity but also aids generalization by reducing overfitting. In CNN, several pooling formulations are employed, such as max, average, L2, overlapping, and spatial pyramid pooling [64, 65].

*Activation function*

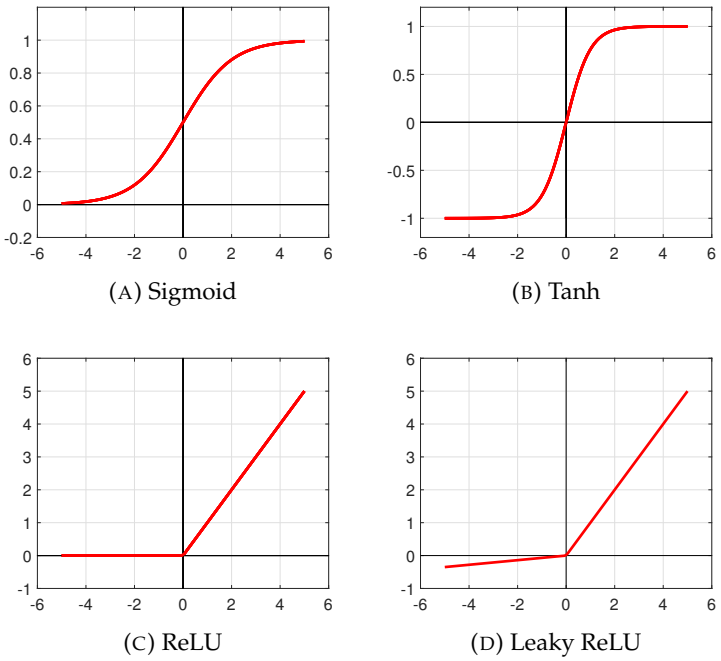
An activation function is a unit that determines which information should be transferred to the next neuron, which is similar to the function of the neuron model of the human brain. Each neuron in the neural network takes the previous layer's output value as input and processes it before passing it on to the next layer. This operation exists between the two layers in a multi-layer neural network and is called the activation function [66].

If no activation function is used or a linear function is utilized, each layer's input will be a linear function of the previous layer's output. In that case, no matter how many layers the neural network has, the output is always a linear combination of the input, indicating that hidden layers have no influence [67]. The primordial perceptron, in Ref. [68], is like this condition and has limited learning capabilities. Nonlinear functions are introduced as activation functions to solve this problem. Deep neural networks with nonlinear activation functions can theoretically approximate any function, considerably improving neural networks' capacity to fit data.

An activation function is defined as

$$\mathbf{T}_l^k = g_a(\mathbf{F}_l^k), \tag{2.21}$$

where $\mathbf{F}_l^k$ is an output of a convolution, which is assigned to activation function $g_a$ that adds non-linearity and returns a transformed output $\mathbf{F}_l^k$ for $l^{\text{th}}$ layer.



(A) Sigmoid (B) Tanh

(C) ReLU (D) Leaky ReLU

FIGURE 2.2: Four typical activation functions.

In this dissertation, four frequently-used activation functions are mainly focused on. To begin, the sigmoid function, which has an overall S-shape, is one of the most common non-linear activation functions (see Figure 2.2(A)). With the x value approaching 0, the gradient becomes steeper. In some tasks, the output value needs to be transformed from a real number to (0, 1), and sigmoid is a good way to implement for binary classification problems. Different from sigmoid, tanh function (see Figure 2.2(B)), can map a real number to (-1, 1) [69]. Since the mean value of the output of tanh is 0, it can achieve a kind of normalization, which makes the next layer easier to learn.

In addition, *Rectified Linear Unit* (ReLU) (see Figure 2.2(C)) is another effective activation function [70]. The function value for x less than 0 is 0; the function value is x itself for x higher than or equal to 0. A notable advantage of employing the ReLU function over the sigmoid and tanh functions is that it can speed up learning. When computing derivatives, sigmoid and tanh use exponential operations that require division, but ReLU's derivative is a constant. Furthermore, if the value of x is too large or too little in the sigmoid and tanh functions, the gradient of the function is relatively tiny, causing the function to converge slowly. Since the derivative of ReLU is 0 when x is less than 0, and 1 when x is greater than 0, it can achieve an ideal convergence effect [71]. However, the ReLU gradient is 0 when x is less than 0. Then the back-propagated error will be multiplied by 0 and no error will be transferred to the preceding layer. The neurons will be considered inactive in this case. As a result, certain enhanced versions are proposed. Leaky ReLU (see Figure 2.2(D)) can reduce neuron inactivation. When x is less than 0, the output of Leaky ReLU is $x/a$. Instead of zero, $a$ is a fixed parameter in the range $(1, +\infty)$ [66].

*Batch normalization*

Batch normalization is used to address the issues related to the internal covariance shift within feature-maps. Batch normalization for a transformed feature-map $\mathbf{F}_l^k$ is shown as

$$\mathbf{N}_l^k = \frac{\mathbf{F}_l^k - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}. \tag{2.22}$$

In Eq. 2.22, $\mathbf{N}_l^k$ represents the normalized feature-map, $\mathbf{F}_l^k$ is the input feature-map, $\mu_B$ and $\sigma_B^2$ depict mean and variance of a feature-map for the batch, respectively. In order to avoid division by zero, $\varepsilon$ is added for numerical stability. By adjusting the distribution of feature-map values to zero mean and unit variance, batch normalization unifies the distribution [72]. It also works as a regulatory factor, smoothing the gradient flow and assisting with network generalization.

*Dropout*

Dropout introduces regularization to the network, which enhances generalization by ignoring randomly some units or connections with a specific probability. In NNs, the non-linear relation is learned by multiple connections that are occasionally co-adapted, it may results in the overfitting problem [73]. The random dropping of some connections or units brings some thinning network topologies, from which the network with minimal parameters is chosen. This chosen design is then used as a rough approximation for the proposed networks [74].

*Loss function*

The loss function, also known as the cost function, is used to calculate the difference between the predicted and actual values. The loss function is commonly used as a learning criterion for the optimization problems [66]. The loss function can be used with convolutional neural networks to solve problems like image generation, with the goal of minimizing the difference between generations and training samples. Common loss functions include *Mean Absolute Error* (MAE, also known as L1 Loss), *Mean Square Error* (MSE, also known as L2 loss), cross entropy, etc.

When dealing with regression tasks in convolutional neural networks, researchers are likely to employ MAE or MSE. The mean of the absolute error between the predicted and actual values is calculated by MAE, whereas the mean of square error is calculated by MSE. For many training cases, researchers prefer using L2 loss instead of L1 since the convergence of the former is faster. But when the training samples contain a number of outliers, L2 error will be much larger compared to L1. In those cases, the difference between an incorrectly predicted target value and original target value will be quite large and squaring it will make it even larger. As a result, L1 loss function is more robust and is generally not affected by outliers. On the contrary L2 loss function will try to adjust the model according to these outlier values, even at the expense of other samples. Hence, L2 loss function is more sensitive to outliers in the dataset [75].

When it comes to classification jobs, convolutional neural networks have a lot of loss functions to deal with. Cross-entropy loss is the most common one to compare the predicting probability distribution with the real one. This function calculates the error depending on the difference between the predicted probability and the real value in each class. Because the error is logarithmic, the function gives minor differences a lower score and larger differences a higher score. Softmax loss is another name for cross-entropy loss, and is usually utilized in networks representing a "softmax" layer.

*Optimizer*

In the training process, the algorithm of CNN is usually required to optimize a non-convex loss function. So, optimizers are employed to reduce the loss and achieve the best network parameters in a reasonable amount of time.

There are three kinds of gradient descent approaches to train CNN models: *Batch Gradient Descent* (BGD), *Stochastic Gradient Descent* (SGD), and *Mini-Batch Gradient Descent* (MBGD). The BGD signifies that an entire batch of data must be calculated to provide a gradient for each update, ensuring convergence to the convex plane's global optimum and the non-convex plane's local optimum. However, using BGD takes a long time because the average gradient of the entire or full batch of samples must be calculated. It can also be difficult with the data that isn't suitable for in-memory calculations. As a result, BGD is rarely used in practice to train CNN-based models. SGD, on the other hand, just uses one sample for every update. Because just one sample's gradient needs to be calculated, SGD takes significantly less time for each update than BGD. SGD is appropriate for online learning in some scenarios. It is updated quickly and with a lot of variation, causing the objective function to oscillate a lot. On the one hand, oscillation can lead the gradient calculation to jump out of the local optimum and eventually find a better position; on the other hand, due to endless oscillation, SGD may never converge. Based on BGD and SGD, MBGD was proposed, which combines the advantages of both. For each update, MBGD uses a small batch of samples, allowing it to not only execute a more efficient gradient descent than BGD but also to reduce variance, resulting in more stable convergence. The most popular of these three approaches is MBGD. Lots of classic CNN models use it to train their networks in original papers. Note that in the algorithms using open-source machine learning libraries, the batch size usually refers to the number of samples in a mini-batch [66].

On the basis of MBGD, a series of effective algorithms for optimization are proposed to accelerate the model training process. Momentum, RMSprop, Adam, and other optimization techniques are typical. Qian et al. proposed the Momentum algorithm [76]. It simulates physical momentum by updating weights with an exponentially weighted average of the gradient. However, the learning process will become unbalanced if the gradient in one dimension is substantially larger than the gradient in another dimension. Another frequently-used optimizer is *Adaptive Moment Estimation* (Adam) [77]. It is essentially an algorithm formed by combining the Momentum and the RMSProp [78]. The Adam method has been proven to function effectively in a variety of situations and convolutional neural network architectures [79, 80].

At the last of this section, the parameters and hyperparameters used in different CNN components are summarized in Table 2.2.

### 2.2.3 Architectures of CNN

Since the proposal of LeNet in 1994, researchers have invented a variety of CNN architectures to make CNN realizable to large, complex, and multi-class problems. Innovations include different aspects such as modification of processing units, design patterns, and connectivity of layers, etc. The work presented in this cumulative dissertation is based on U-Net. Here, U-Net and the earlier dependent architecture are presented and discussed.

*LeNet*

TABLE 2.2: Parameters and hyperparameters in the components of convolutional neural networks. A parameter is the variable that is automatically optimized during the training process and a hyperparameter is the variable that needs to be set beforehand [81].

| Components | Parameters | Hyperparameters |
|---|---|---|
| convolutional layer | kernels | kernel size, number of kernels, stride, padding, activation function |
| pooling layer | none | pooling method, filter size, stride, padding |
| fully connected layer | weights | number of weights, activation function |
| others | | model architecture, optimizer, learning rate, loss function, batch size, epochs, regularization, weight initialization, dataset splitting |

LeNet is one of the first convolutional neural networks which was proposed in 1994. After some revisions, LeCun et al. introduced LeNet-5, a groundbreaking architecture based on gradient back-propagation [69]. The goal of LeNet-5 is to automatically identify hand-written digits on bank checks on a wide scale, with a recognition accuracy of 99.2% on *Modified National Institute of Standards and Technology Database* (MNIST).

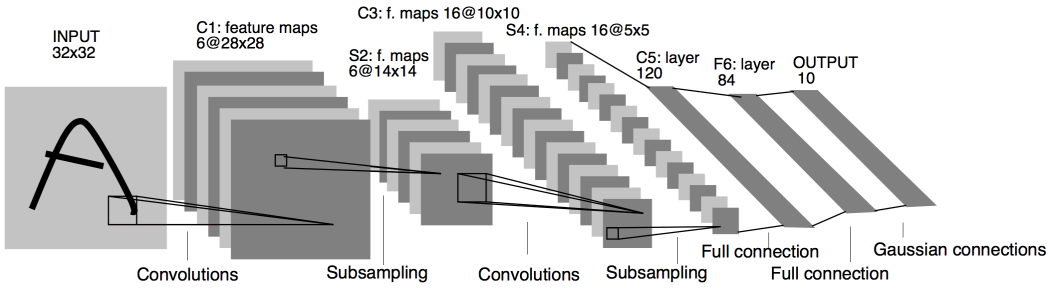



FIGURE 2.3: Architecture of LeNet-5. Each plane is a feature map [69].

As shown in Figure 2.3, two convolutional layers, three fully-connected layers, and two pooling layers make up LeNet-5. Because LeNet-5 combines local receptive fields, shared weights, and spatial and temporal subsampling, it can assure shift, scale, and distortion invariance to some extent. LeNets are regarded as the foundation of modern CNN architecture. However, it still does not exceed the traditional *Support Vector Machine* (SVM) and boosting algorithms [66].

*AlexNet*

Krizhevsky expands on LeCun's theories by applying the LeNet-5 core principles to a wider and deeper architecture [46]. Meantime, for the first time, AlexNet successfully uses the ReLU activation function, dropout, and local response normalization. During the time of LeNet-5, hardware limited deep CNN architectures to tiny sizes and restricted the learning capacity. However, AlexNet was trained in parallel on two NVIDIA GTX 580 GPUs to make use of deep CNNs' representational capacity.

As shown in Figure 2.4, AlexNet has eight layers. The depth increase compared with LeNet enhances generalization for varied image resolutions and makes CNN

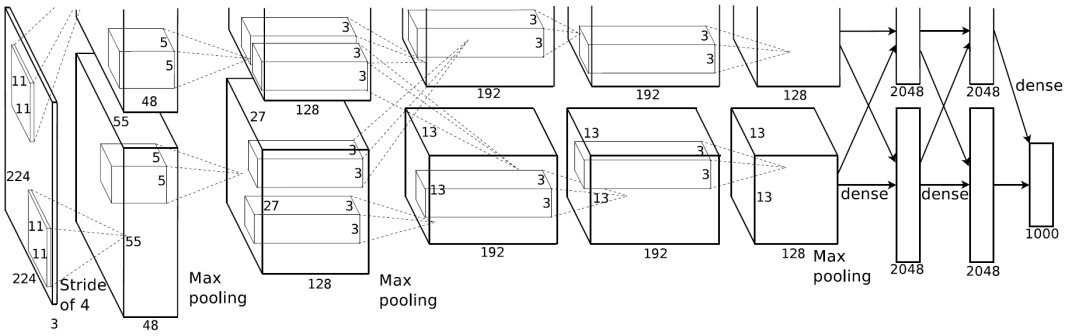



FIGURE 2.4: Architecture of AlexNet, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers [46].

suitable for a broader range of image categories. However, it will be accompanied by overfitting. To overcome this limitation and learn more robust features, AlexNet employs dropout to randomly discard some neurons during training. Following that, dropout is commonly employed in the final few fully-connected layers [82]. Furthermore, AlexNet employs ReLU [83] as the activation function, which mitigates the issue of gradient vanishing problem in deep networks and so improves the convergence speed to some extent. In comparison to previously proposed networks, AlexNet's efficient learning technique is critical in the development of CNNs, and it has ushered CNN architecture into a new era of development.

*Skip Connection*

Deep networks naturally combine low/mid/high-level features and classifiers in an end-to-end multi-layer fashion, with the number of stacked layers enhancing the depth of features. To solve the degradation problem from deep structure, He et al. proposed *Residual Neural Network* (ResNet) [16]. As shown in 2.5, the building block

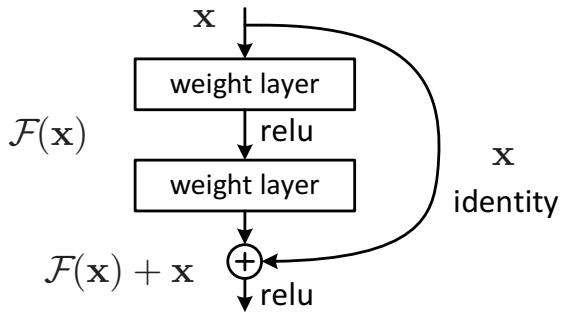



FIGURE 2.5: Building block in ResNet. Compared with MLP, ResNet has an additional shortcut connection between the input layer and the output layer, shown by the curve, circumventing all the hidden layers. In other words, ResNet consists of a fully connected neural network and a shortcut connection [16].

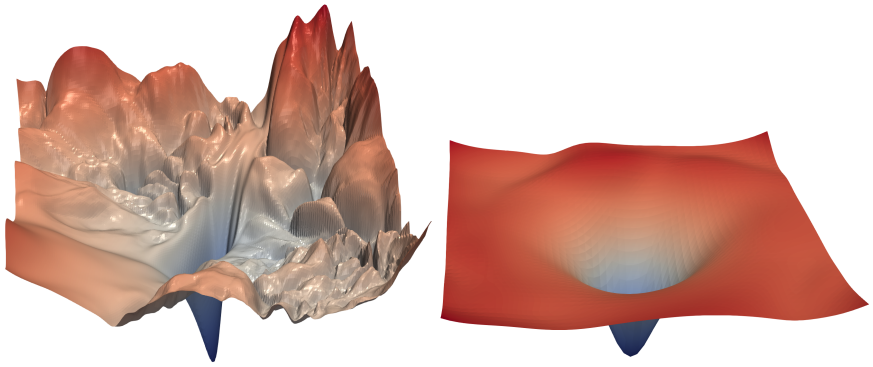
adopts identity mapping by shortcuts to every few stacked layers. It is formally defined as:

$$y = \mathcal{F}(x, \{W_i\}) + x, \tag{2.23}$$

where x and y are the input and output vectors of the layers considered. The function $\mathcal{F}(x, \{W_i\})$ represents the residual mapping to be learned. For the example in Fig.

2.5, if the biases are omitted for simplifying notations, the residual mapping could be written as $\mathcal{F} = W_2\sigma(W_1\mathrm{x})$ in which $\sigma$ denotes activation function $ReLU$. The operation $\mathcal{F} + \mathrm{x}$ is performed by element-wise addition. There are no additional parameters introduced by this shortcut connection since it is a simple addition, the increase in processing complexity is insignificant.

Although the notations above are about the fully-connected layers for simplicity in Ref. [16], they are also applicable to convolutional layers. The element-wise addition is done channel by channel on two feature maps, i.e., the output of convolutional blocks.



(A) without skip connections (B) with skip connections

FIGURE 2.6: The loss surfaces of ResNet-56 with/without skip connections [84].

It is observed in Figure 2.6 that skip connections promote a flat loss surface and prevent the transition to chaotic behavior, which helps explain why skip connections are necessary for training extremely deep networks.

ResNet realizes skip connection via addition, while in general, there are other ways that use skip connections through different non-sequential layers: concatenation in densely connected architectures. For this, the most famous architecture is DenseNet [85]. To enable maximal information to flow across layers in the network, as opposed to ResNets, this architecture connects all layers directly with each other via concatenation. This results in a) a massive number of feature channels on the network's last layers, b) more compact models, and c) extraordinary feature reuse [86].

*U-net*

For skip connections, there actually are two kinds of setups: a) short skip connections, b) long skip connections. Short skip connections are typically employed in consecutive convolutional layers that do not modify the input dimension (see ResNet), whereas long skip connections are more common in encoder-decoder layouts.

Long skip connections often exist in architectures that are symmetrical, where the spatial dimensionality is reduced in the encoder part and is gradually increased in the decoder part as illustrated in Figure 2.7. In the decoder part, the dimensionality of a feature map is increased via transpose convolutional layers. This transposed convolution process creates the same connectivity as the standard convolution, but in reverse. By introducing skip connections in the encoder-decoder architecture, fine-grained details can be recovered in the prediction. Symmetrical long skip connections work incredibly effectively in dense prediction tasks [87], such as image segmentation or the field solution generation.

As the pioneering architecture that utilizes long skip connection, U-net is a widely-used architecture of CNN which is originally designed for biomedical image segmentation [88] and has previously been used for flow field reconstruction with data-driven learning [42].

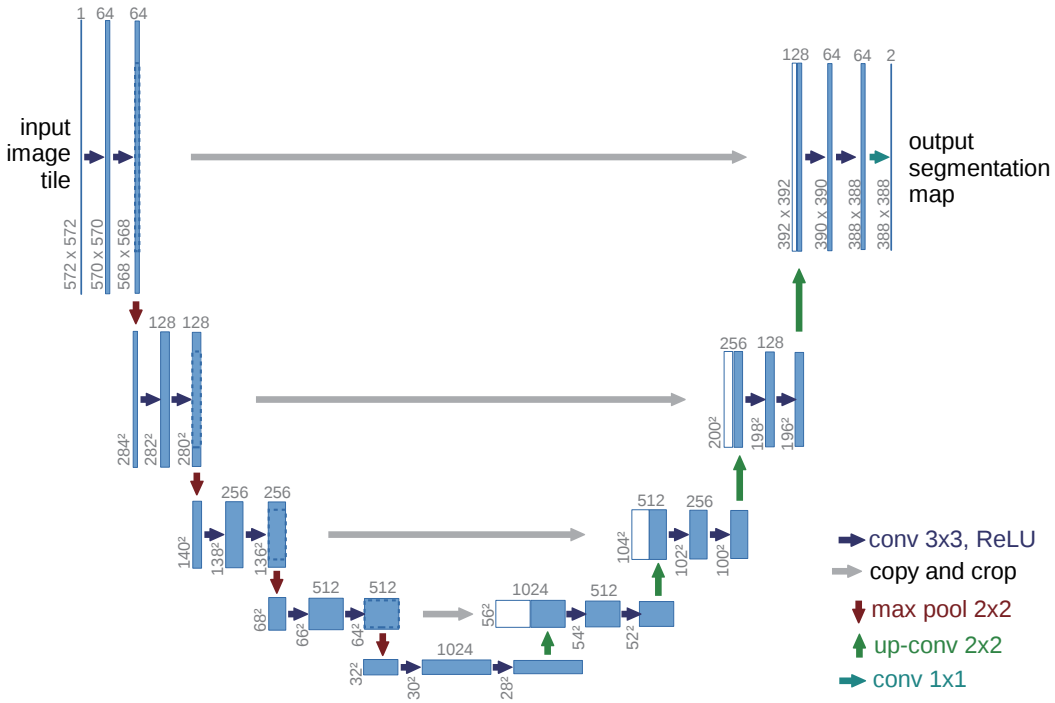



FIGURE 2.7: Architecture of U-Net, example for 32x32 pixels in the lowest resolution. Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower-left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations [88].

One important modification in the architecture is that the up-sampling part has also a large number of feature channels, which allow the network to propagate context information to higher resolution layers. As a consequence, the expansive path is more or less symmetric to the contracting path and yields a U-shaped architecture [89]. The network does not have any fully connected layers and only uses convolution so it belongs to the class of fully convolutional neural networks.

In our work, including the input and output layers, the U-net architecture consists of multiple layers and corresponding convolutional blocks. The input layer consists of a number of channels that contain the input conditions. In Paper III [37], for example, two of them, $u_0$ and $v_0$, are inflow velocities in $x$ and $y$ directions, which are uniform non-dimensional values in the whole learning domain. The geometry channel $G$ describes the shape of the object in the flow fields. When there is an object in the flow, all values inside it be marked as 1 and the other as 0. In this way, the geometry is embedded into the network and it is also used for evaluating the physics-based loss function. The output layer consists of a number of channels that contain the obtained physics quantities. For example, the velocities in both x and y directions and pressure respectively. These outputs are also non-dimensional values.

The core target of this dissertation is to generate physics fields constrained by

given physics laws or training data. For steady problems, the PDEs, i.e. the mathematical expressions of underlying physical laws, represent the spatial relationships of adjacent positions. Similarly, the convolutional kernels extract the spacial feature of the receptive field consisting of a group of adjacent pixels. In the U-net architecture used, the encoding part is responsible for recognizing the geometry and initial conditions of the physics field, in order to extract the necessary features representing the physics of the inputs using convolution operation layer by layer. These features are the basis for the subsequent decoding part. Here, the layers of the decoding process at different depths store the physical feature maps and the spacial relationship are recovered by the inverse convolutional calculation. Eventually, the decoding part is able to reconstruct the proper flow field under the constraints of PDEs or training data. In addition, there are the concatenations of the feature channels between encoding and decoding as the gray arrows denoted in Figure 2.7. Duplicating the feature channels from the encoding blocks to the corresponding decoding ones, the "skip connections" effectively double the number of feature channels in each decoding layer and enable the network to consider the information from the encoding layers, which are extracted from the geometry and initial conditions.

The architecture used in this dissertation is symmetrical, which means the encoding and decoding processes have the same depth, meanwhile, the amounts and dimensions of corresponding blocks are the same. However, the depths of the two processes are both adjustable. In some practical tasks, a coarse input compressed by fewer encoding layers is also able to generate a high-resolution solution reconstructed by more decoding layers. More details of the U-net architecture and convolutional block can be found in Appendices.

## 2.3 CNN for film cooling flow

This section mainly describes the data-driven method for film cooling process prediction in a rocket combustor. Emphasis is on the basic CNN architecture for physics field solution prediction with data-driven methods.

As the initial temperature field defined in Subsection 2.1.2 shows, the bottom three rows present the height of the film part of the inlet flow. Using this method, flow conditions of mainstream and coolant are encoded in a $64 \times 256 \times 3$ grid of values; the first two channels contain the mass flux of mainstream and coolant film while the last of the channels contain the temperature of the film. The targets, extracted from the RANS solution calculated by ANSYS Fluent, include 16 items: both x and y components of velocity, temperature, pressure, and concentrations of the 12 gaseous species from both hot gas mixture and coolant film. So, the data sets for supervised training have the same size with inputs but different channel numbers. The first four channels contain the flow field information including x and y velocity components, pressure, and temperature sequentially. The next twelve channels contain the concentration of every species. From the simulation domain with more than 20K cells, each target is interpolated into one $64 \times 256$ matrix, eventually obtaining a $64 \times 256 \times 16$ targets data grid.

The algorithms are based on the PyTorch platform [90]. The U-net CNN architecture is illustrated in Figure 2.8. In the beginning, the main-flow mass flux, coolant mass flux, and coolant temperature are introduced into the architecture as three rectangular matrices. After two convolutional calculation layers utilized with rectangular kernels, the original rectangular matrices are transferred into square matrices. Then the square kernels are utilized until the single-value vectors are obtained. In

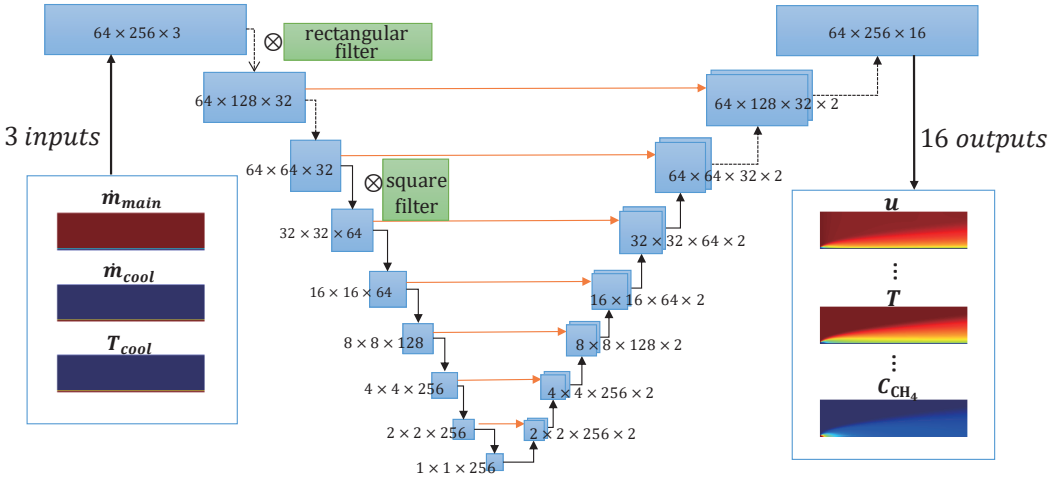



FIGURE 2.8: U-net architecture for film cooling flow filed prediction.

this encoding process, the matrices of mass fluxes and temperatures are progressively down-sampled by convolutional calculations. With the number of feature channels increasing, abstract and large-scale information is extracted by the convolutional neural networks. Then the decoding part, an inverse convolutional process, mirrors the behavior of the encoding part. The solutions are reconstructed in the up-sampling layers along with the increase of spatial resolution. Eventually, rectangular matrices with 16 channels can be obtained, which express the flow field information and concentration information of the gaseous species. It is noteworthy that there are concatenation operations between two different channels, which represents a "skip connection" function introduced in Subsection 2.2.3.

The U-net architecture for film cooling prediction consists of 17 layers and each layer has a number of convolutional blocks. Each convolutional block has a similar structure: batch normalization, activation function, convolutional calculation, and dropout. Convolutional blocks (represented by C) are usually parametrized by channel factor c, convolutional kernel size k, and stride s. cX shortly denotes c = X, the channel number is the product of c and a basic multiplier 32. kXY shortly denotes k =(X, Y) in two dimensions. In addition, r in the block shown below indicates activated by ReLU, and l indicates activated by a leaky ReLU. Batch normalization is indicated by b. In order to improve accuracy, up-sampling (up []) is used to substitute converse convolutional calculation which is widely used in the research of super-resolution. And "conc ()" denotes concatenation operation.

So, the convolutional blocks in the decoder can be summarized as

$$\begin{aligned} l_0 &\rightarrow C(k41, s21) \rightarrow l_1 \\ l_1 &\rightarrow C(c1, k41, s21) \rightarrow l_2 \\ l_2 &\rightarrow C(c1, k44, s22, l, b) \rightarrow l_3 \\ l_3 &\rightarrow C(c2, k44, s22, l, b) \rightarrow l_4 \\ l_4 &\rightarrow C(c2, k44, s22, l, b) \rightarrow l_5 \\ l_5 &\rightarrow C(c4, k22, s11, l, b) \rightarrow l_6 \\ l_6 &\rightarrow C(c8, k22, s11, l, b) \rightarrow l_7 \\ l_7 &\rightarrow C(c8, k22, s11, l, b) \rightarrow l_8. \end{aligned} \tag{2.24}$$

The convolutional blocks in the encoder can be summarized as

$$\begin{aligned} l_8 &\rightarrow \text{up}\left[C(c8, k22, s11, r, b)\right] \rightarrow l_9 \\ \text{conc}(l_9, l_7) &\rightarrow \text{up}\left[C(c16, k22, s11, r, b)\right] \rightarrow l_{10} \\ \text{conc}(l_{10}, l_6) &\rightarrow \text{up}\left[C(c16, k22, s11, r, b)\right] \rightarrow l_{11} \\ \text{conc}(l_{11}, l_5) &\rightarrow \text{up}\left[C(c8, k44, s22, r, b)\right] \rightarrow l_{12} \\ \text{conc}(l_{12}, l_4) &\rightarrow \text{up}\left[C(c4, k44, s22, r, b)\right] \rightarrow l_{13} \\ \text{conc}(l_{13}, l_3) &\rightarrow \text{up}\left[C(c4, k44, s22, r, b)\right] \rightarrow l_{14} \\ \text{conc}(l_{14}, l_3) &\rightarrow \text{up}\left[C(c2, k41, s21, r, b)\right] \rightarrow l_{15} \\ \text{conc}(l_{15}, l_2) &\rightarrow \text{up}\left[C(c2, k41, s21, r)\right] \rightarrow l_{16}. \end{aligned} \tag{2.25}$$

In the prediction of mixing shear layer between coolant and hot gas, the generated field solutions are then introduced into the loss function that compares the difference between outputs and targets obtained from RANS simulations. And the loss is utilized to update the CNN's parameters. After a number of epochs, the CNN model is able to generate the accurate filed solutions eventually.

The modified U-net CNN described here is the basic network architecture for the methodologies in all the four journal publications accumulated in this dissertation, see Appendix A.1 - A.4. In these papers, the CNN are modified to fit the specific tasks. The CNN output after the upsampling process is subsequently introduced to the different loss functions or evaluators.

## 2.4 Physics-driven method

Similar to the presented approach in Section 2.3, the previous research on flow field prediction using CNN is mainly focused on data-driven methods. However, for complex practical engineering problems, the training samples employed in the loss function may be hard to obtain. In the contrast, the physics-driven method trains the model to obtain the solution of the physics field without labeled data. In Paper II [35] and Paper III [37], two fundamental scientific problems were chosen to study the physics-driven method.

*Heat conduction*

In Paper II [35], choosing the heat conduction problem as an example, the two-dimensional Laplace equation is used to drive the learning process by the loss function

$$\mathcal{L}_{\text{Laplace}} = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = e_1. \tag{2.26}$$

The boundaries are constrained by Dirichlet boundary conditions by which the temperatures of the outer and inner boundaries are kept a constant. The boundary conditions are implemented differently for the outer and inner boundaries. While the temperatures at the outer boundaries are assigned as constants, their values at the inner boundary as well as the inside void region are constrained by a loss function as

$$\mathcal{L}_{\text{BC,in}} = T - T_{\text{BC,in}} = e_2. \tag{2.27}$$

Note that, for the physics-driven method, the second input channel of the U-net CNN not only describes the geometry but also functions as a mask, by which the Laplace equation is not effective in the void region.

*Fluid mechanics*

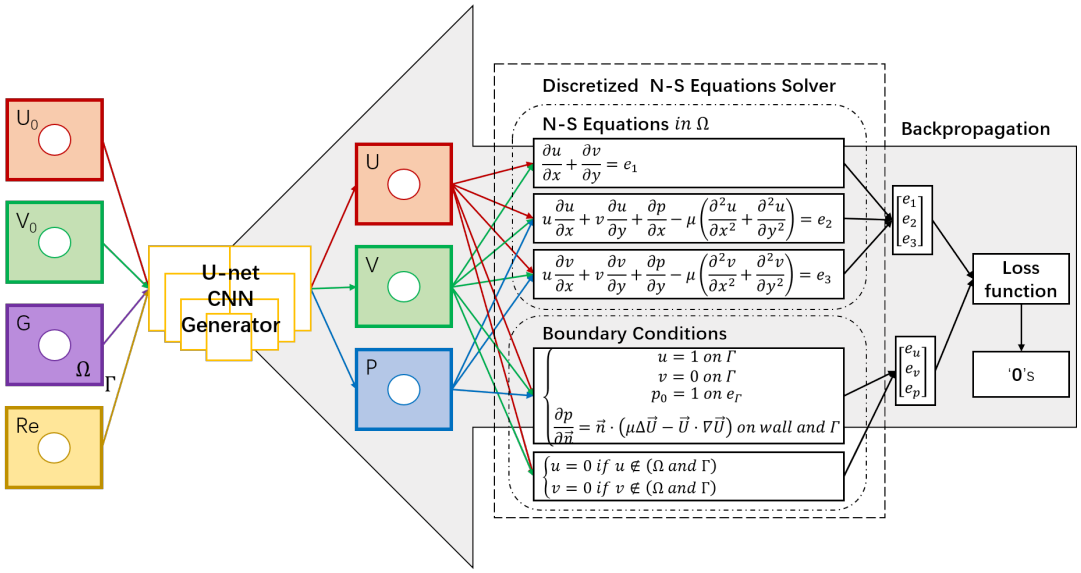



FIGURE 2.9: Physics-driven learning for solving the N-S equations. The U-net CNN generates the solution. The backpropagation computes the gradient of the loss function and updates the weights of the CNN to satisfy the discretized N-S equations and boundary conditions.

In Paper III [37], for flow around a cylinder problem, once the preliminary flow field is obtained from the U-net CNN generator, the physical constraints are applied to this field as shown in Figure 2.9. The learning domain is separated as inner domain and boundaries, which are represented by $\Omega$ and $\Gamma$ respectively. In the inner domain $\Omega$, the left-hand sides of N-S Equations are employed as parts of the loss function and 3 residuals can be obtained as

$$\begin{cases} \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = e_1 \\ u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} - \mu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = e_2 \\ u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} - \mu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) = e_3 \end{cases} \tag{2.28}$$

So the residuals of the N-S equations in $\Omega$ can be represented as $E_\Omega = [e_1, e_2, e_3]^{\mathrm{T}}$.

In order to obtain a differentiable formulation of the physics in the loss function, suitable convolutional filters are designed to compute the N-S equations via finite differences. Similar approaches have been proposed previously for simple PDEs [53], and the partial differential operators for two different dimensions are constructed separately [91, 92]. The construction via convolutions has the advantage that the backpropagation of a deep learning framework can be used, and the finite difference kernels yield well-controlled accuracies for the derivative calculations. Choosing the x direction as an example, the weights of the filters are represented as

$$\mathrm{W}_{\frac{\partial}{\partial x}} = \begin{bmatrix} 0 & -0.5 & 0 \\ 0 & 0 & 0 \\ 0 & 0.5 & 0 \end{bmatrix}, \mathrm{W}_{\frac{\partial^2}{\partial x^2}} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \tag{2.29}$$

After the rotating and moving operation through the matrix obtained from the last

layer, the first- or second-order partial derivatives of local quantities are calculated. Choosing $u$ as an example, this procedure can be written as:

$$g_{i,j} = \sum_{m=0}^{2} \sum_{n=0}^{2} u_{i+m-1,j+n-1} \cdot f_{m,n}, \tag{2.30}$$

where $f_{m,n}$ is the convolutional filter and the $g_{i,j}$ is the central difference of $u$ in each data point.

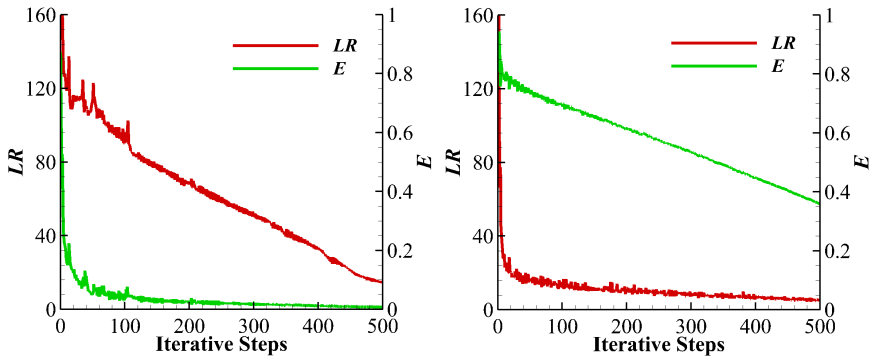
On the boundary $\Gamma$, including inflow and outflow side and walls, the Dirichlet and Neumann boundary conditions are considered as shown in Figure 2.9. The residuals of $u$, $v$ and $p$ are represented as $E_\Gamma = [e_u, e_v, e_p]^{\mathrm{T}}$. Combining both as $E = [E_\Gamma, E_\Omega]^{\mathrm{T}}$, the whole residual of the physics-driven method is obtained.

To reduce the residuals, the CNN is trained in an iterative manner using a stochastic gradient descent variant (here Adam [77] is employed). After the CNN generator, the preliminary flow fields are introduced in the loss function, and then the residuals $E$ are obtained. Once the backpropagation is applied, the weights and biases of CNN are adapted to minimize these physics-based residuals. Eventually, the high-resolution flow fields which obey physical laws and corresponding boundary conditions will be obtained.

## 2.5 Combined data-driven and physics-driven method

Machine learning methods can be classified into two distinct types: data-driven methods presented in Section 2.3 and physics-driven methods presented in Section 2.4.

In Paper II [35], the learning processes of these data- and physics-driven methods with deep CNN were compared. The results show that the convergence of the error towards a ground truth solution and the residual of heat conduction equation exhibit remarkable differences. For data-driven learning, the loss function only considers




(A) Data-driven method (B) Physics-driven method

FIGURE 2.10: Training history of $LR$ and $E$ (steps = $0 \sim 500$). The convergence behaviors of two learning methods exhibit significant differences.

the *Error* ($E$) between output and target rather than the *Residual of Laplace equation* ($LR$), so I call $E$ the loss term or explicit error and $LR$ the non-loss term or implicit error. Similarly in physics-driven learning, $LR$ is the explicit error, while $E$ represents the implicit error.

As shown in Figure 2.10, for both data-driven and physics-driven learning, $E$ and $LR$ drop dramatically in the beginning. However, after approximately 10 iterative steps, when the explicit errors have gradually approached an adequately small value, the implicit errors are still large. Finally, after a much larger number of iteration steps, both errors decrease to sufficiently small values, i.e. both the solution and its local structure are obtained.

*Heat conduction*

Based on the above observations, Paper II [35] proposed a novel combined data-driven and physics-driven method to improve the physical consistency and increase the convergence speed by combining both $E$ and $LR$ into the loss terms as explicit errors. It is observed that the scale of $LR$ is significantly larger than $E$. Utilizing a simple summation of these two errors as the total loss leads to a skewed optimization [93] with a dominance of the Laplace residual. In order to remedy this issue, I employ a weighted loss function that has been widely used in object detection [94] and audio detection [95]. The weighted loss function considering both target data and Laplace equation is written as

$$\mathcal{L}_{\text{heat}} = \mathcal{L}_{\text{data}} + \mathrm{R}_{\text{heat}} * \mathcal{L}_{\text{Laplace}}, \tag{2.31}$$

where $\mathrm{R}_{\text{heat}}$ is a constant hyperparameter, and the value of which can be speculated by the prior numerical experiments and manually-tuned to adapt the scales. With this weighted loss function, the different loss terms can be easily scaled to an equivalent magnitude. The combined method actually has two types: data-driven based and physics-driven based. The former one is aiming to improve the original data-driven method. For the data-driven based method, the loss function is Equation (2.31) and employed during the whole learning process. The latter is aiming to improve the performance of the original physics-driven method. The loss function is modified as

$$\mathcal{L}_{\text{heat}} = \begin{cases} \mathcal{L}_{\text{heat,ref}} + \mathrm{R}_{\text{heat}} * \mathcal{L}_{\text{Laplace}} & \mathcal{L}_{\text{heat}} \geq \mathcal{L}_{\text{thr}} \\ \mathcal{L}_{\text{Laplace}} & \mathcal{L}_{\text{heat}} < \mathcal{L}_{\text{thr}} \end{cases}, \tag{2.32}$$

where $\mathcal{L}_{\text{heat,ref}}$ is the error term with a given reference target depending on the different practical consideration. $\mathcal{L}_{\text{thr}}$ is the threshold value of the loss indicating that once the loss is less than the threshold, the loss function will only consist of the Laplace term.

*Fluid mechanics*

For the flow around a cylinder problem in Paper III [37], the weights of physics-driven CNN are adapted only to minimize the residuals of PDEs, the solution itself is not constrained, which means there is no target being offered for reference. In order to accelerate the convergence and eventually improve the training performance, besides the physical laws, additional reference targets are provided for constraining the network.

Similar to data-driven methods, there is a reference loss term comparing the difference between output and target, which is defined as

$$\mathcal{L}_{\text{fluid,ref}} = \sum_{i=1}^{\mathcal{I}} \sum_{n=1}^{\mathcal{N}} \left| \mathcal{X}_{\text{out}} - \mathcal{X}_{\text{tar}} \right|. \tag{2.33}$$

The subscript "ref" here denotes reference targets. Generally, $\mathcal{X}_{\text{out}}$ and $\mathcal{X}_{\text{tar}}$ are output quantities and corresponding targets, respectively, $\mathcal{I} = \textit{targets amount}$, meanwhile $\mathcal{N} = \textit{batch size}$ denotes the amount of training data in one batch operation. The total loss considering both reference targets and physics laws can be represented as

$$\mathcal{L}_{\text{fluid}} = \mathcal{L}_{\text{fluid,ref}} + \mathrm{R}_{\text{fluid}} * \mathcal{L}_{\text{NS}}, \tag{2.34}$$

where $\mathcal{L}_{\text{NS}}$ is the physical loss term considering the N-S equations and boundary conditions. Similar with Equation (2.31), $\mathrm{R}_{\text{fluid}}$ is a constant hyperparameter which is tuned to adapt the scales.

In the physics-driven training, a certain amount of randomly picked Reynolds numbers are input as one batch in each iterative step. In contrast, in the accelerating approach with reference targets, I also use some constant Reynolds numbers beside the variable ones. So, the new batch includes two groups as shown in Figure 2.11. The cases in the random group vary in every iterative step and are trained with

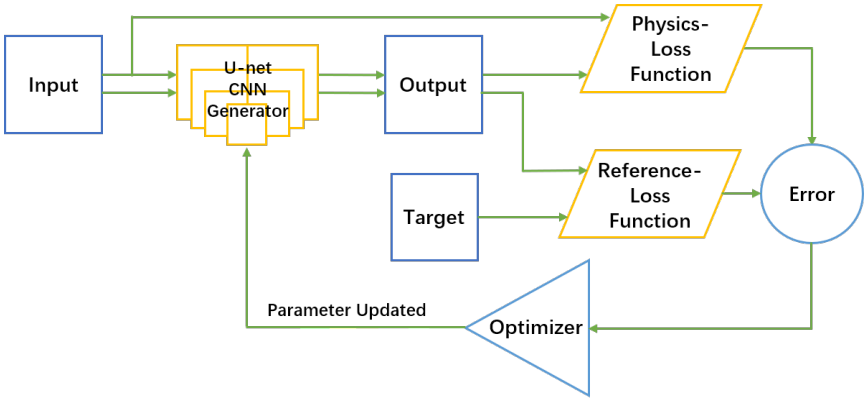



FIGURE 2.11: Acceleration with reference targets. One batch consists of the physics and reference groups. The outputs of physics group are introduced to physical loss function, while the outputs of reference group are introduced to the reference loss function.

the physical loss $\mathcal{L}_{\text{NS}}$. While the ones in the constant group are fixed in the whole iterative process and are trained with reference loss $\mathcal{L}_{\text{fluid,ref}}$.

In the numerical experiments of PD-CNN which are accelerated with reference targets, there are 18 cases in each batch. The first half batch consists of 9 random *Re* varied with epoch number, while the other half batch consists of 9 constant *Re* fixed in the whole training process. As discussed in Appendix A.3, the cases whose Reynolds numbers are near 1 are much more difficult to train. So, the manually defined 9 constant Reynolds numbers are 1.0, 1.5, 2.0, 3.0, 4.0, 6.0, 8.0, 12.0 and 18.0. In the whole training process, the numerical solutions obtained by FVM of these 9 constant Reynolds numbers are input as nine targets and the CNN is trained to minimize Equation (2.33).

In this way, the reference targets restrain the results generated by the physics-driven method to approximate real solutions faster. Since the targets only include a limiting number of cases and are used through the whole training process, this approach reduces the expensive data generation cost of the traditional data-driven methods. In practical engineering applications, the reference targets can be easily picked from the existing data, e.g., experimental and numerical results.

## 2.6 GAN with physical evaluators

In the state-of-the-art neural networks methods, *Generative Adversarial Networks* (GANs) proposed by Goodfellow et al. [96], are efficient to generate the instantaneous flow fields [97, 98]. In spite of the impressive performance for unsupervised learning tasks, the quality of generated solutions by GANs is still limited for some realistic tasks [99]. In Paper IV [38], based on the CNN architecture presented in previous sections, a novel GAN is explored to simulate instantaneous spray fields under continuous operating conditions.

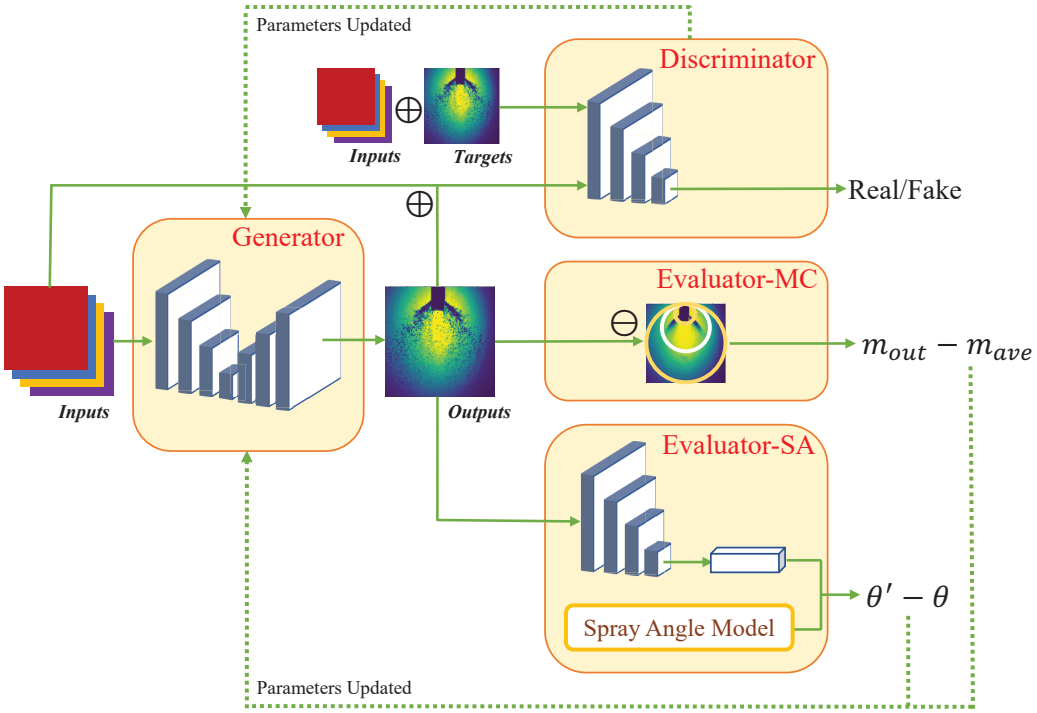



FIGURE 2.12: Schematic of the proposed networks framework. With the operating conditions, the U-net generator outputs a field solution of the spray. Then the outputs will be transferred to the discriminator, mass conservation evaluator, and spray angle evaluator. With the input-target pair and input-output pair, the discriminator is trained to distinguish the real and fake images. The mass conservation evaluator calculates the ring error between output and the corresponding average target. The spray angle estimator judges the angle value from the output and then compares it with the theoretical one. Eventually, the discriminator, mass conservation, and spray angle losses are utilized to update the generator by backpropagation.

The *Generative Adversarial Networks with Physical Evaluators* (GAN-PE) framework is shown in Figure 2.12, The GAN-PE is composed of 4 parts, *Generator* ($G$), *Discriminator* ($D$) and two physical evaluators.

The field solutions are generated by $G$, and the other three parts are employed to guarantee the outputs catch the spray morphology and obey the operating conditions. A GAN is the base of the proposed networks framework, the $G$ captures the real spray data distribution which is corresponding to the operation conditions, and the $D$ estimates the probability that a condition-sample pair came from the training data rather than $G$. There are also two evaluators designed to improve the performance of GANs. The first, *Mass Conservation Evaluator* ($E_{\mathrm{MC}}$), is used to improve

the generation robustness by calculating the ring error between output and the corresponding average target. The second, *Spray Angle Evaluator* ($E_{\mathrm{SA}}$), is used to improve the predicting accuracy in the specific operating conditions. Fed with the outputs from $G$, the losses of $D$, $E_{\mathrm{MC}}$ and $E_{\mathrm{SA}}$ are calculated respectively. After that, the backpropagation is applied to adjust the U-net CNN of $G$ to generate a new spray field that more satisfies the conditions and prior physics knowledge. After enough iterations, the network will be able to generate a reliable spray field solution.

### 2.6.1 GAN

From inputs to outputs, the network of $G$ consists of two parts: encoding and decoding [88]. In the encoding process, the operating conditions $L_{\mathrm{open}}$, $T_{\mathrm{gs}}$, $m_{\mathrm{g}}$ and $m_{\mathrm{l}}$ are resized as four feature channels of the input tensor for convolutional down-sampling with corresponding kernels [66, 52]. After that, the matrices with a size of $128 \times 128$ are progressively reduced into 512 single-value vectors. Each layer of the network consists of a convolution operation, batch normalization, and a non-linear activation function. By the convolutional calculation, along with the number of feature channels increasing, the matrices size is down-sampled by a factor of 2. In this way, the information of operating conditions is translated into the extracted features in the next layer. In addition, skip-concatenations from input to output feature channels are introduced to ensure operating conditions information is available in the following up-sampling process for inferring the solution. Then the decoding part works in an opposite way, which can be regarded as an inverse convolutional process mirroring the behavior of the encoding part. Along with the increase in spatial resolution, the spray fields are reconstructed based on the single-value vectors by up-sampling operations.

The weighted loss function considering the following discriminator and evaluators is written as

$$\mathcal{L}(D, E_{\mathrm{MC}}, E_{\mathrm{SA}}) = \mathcal{L}_D + \alpha \mathcal{L}_{E_{\mathrm{MC}}} + \beta \mathcal{L}_{E_{\mathrm{SA}}}, \tag{2.35}$$

where $\mathcal{L}_D$, $\mathcal{L}_{E_{\mathrm{MC}}}$ and $\mathcal{L}_{E_{\mathrm{SA}}}$ are the loss terms calculated by $D$, $E_{\mathrm{MC}}$ and $E_{\mathrm{SA}}$ respectively. Also, $\alpha$ and $\beta$ are the constant hyperparameters that are manually tuned before training to adapt the scales of these loss terms. After proper training, both generator and discriminator losses keep stable and the generator is able to map a spray sample from a random distribution to the desired one which obeys the physical knowledge and conditions.

The discriminator in a GAN is simply a classifier. It tries to distinguish real samples from the data created by the generator, i.e., fake data. The discriminator's training data comes from two sources. One consists of the real data instances, here are the real experimental images. $D$ uses these instances as positive examples during training. The other is the fake data instances created by the generator. The discriminator uses these instances as negative examples during training. Then $D$ is used to feed the possibility that samples come from the targets rather than generation distribution back to $G$. The settings of *Least Squares Generative Adversarial Networks* (LSGANs) are utilized to train the $D$ and the $G$ simultaneously [100]. This special type of GANs helps to remedy the gradients vanishing by using the least square loss function instead of the sigmoid cross-entropy loss function [101].

Here, $D$ is modified by the encoder of the $G$, which means the generated solutions from $G$ are down-sampled by the re-convolutional calculation so that the spray field information is transferred into the linear 1-D tensor. And then this 1-D tensor will be used to be trained to maximize the probability of assigning the correct label,

real or fake, to both training targets and generated solutions. Similar to the work in Ref. [102], I use the input-output and input-target pairs to feed $D$ instead of only output and target in the random image generation tasks. The operating conditions and the outputs/targets are concatenated as the different feature channels in a 4-D data tensor. In this way, the $D$ not only discriminates between real and fake values but also helps to judge whether the outputs match the corresponding conditions.

The loss functions for LSGANs are defined as

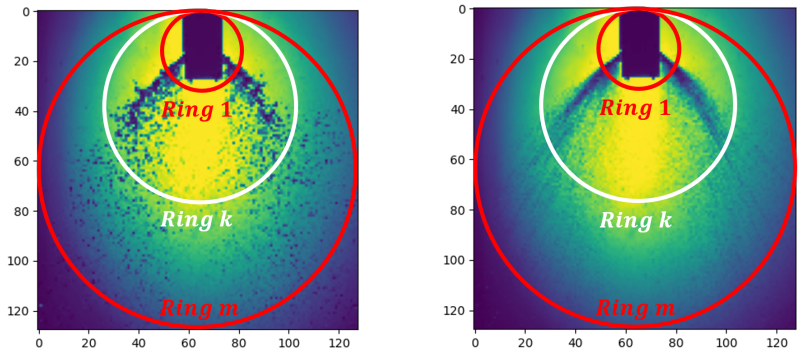
$$\begin{aligned} \min_D V_{\text{GAN}}(D) &= \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[(D(\boldsymbol{x})-b)^2] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[(D(G(\boldsymbol{z}))-a)^2] \\ \min_G V_{\text{GAN}}(G) &= \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[(D(G(\boldsymbol{z}))-c)^2], \end{aligned} \tag{2.36}$$

where $\boldsymbol{x}$ is the training data and $\boldsymbol{z}$ is the input variables. Also, $a$ and $b$ are the labels for fake data and real data respectively, and $c$ denotes the value that $G$ wants $D$ to believe for the generated solutions. Here, I apply $a = 0$ and $b = c = 1$. So, $\mathcal{L}_D$ is equal to the second part of Eq. (2.36).

### 2.6.2 Physical evaluators

*Mass conservation evaluator*

Originated from the idea that the spray phenomenon obeys the mass conservation law, a mass conservation evaluator is designed. As shown in Figure 2.13, there are a few spherical volumes with different diameters that are tangent at the middle point of the upper boundary in both generating images and the average images. The mass fluxes of droplets through one specific ring in every instantaneous frame are equivalent. The spray field generation is indeed a 2-D image reconstruction task. Following the definition of "L1loss" which is widely used in the machine learning community, a mass conservation loss is defined here. The difference is that the former measures the sum of absolute error between each element in the generation and target [90], but mine is calculating first a gray value summation of every element in one concerning ring, then comparing absolute error between the corresponding rings in the generation and target, and finally the summation of the errors.




(A) Generated image (B) Average target

FIGURE 2.13: Mass Conservation. The resolution of the images is 128 × 128. The rings remain tangent at the central point of the top edge.

The mass conservation error, i.e. the loss term from $E_{\mathrm{MC}}$, is defined as

$$\mathcal{L}_{E_{\mathrm{MC}}} = \begin{cases} \sum_{k=1}^{m} \left| \sum_{i=1}^{n} x_i - \sum_{j=1}^{n} y_j \right|_k & \mathcal{E} \geq \mathcal{E}_{\mathrm{thr}} \\ 0 & \mathcal{E} < \mathcal{E}_{\mathrm{thr}} \end{cases}, \tag{2.37}$$

where $x$ and $y$ are the gray values in generated images and average targets respectively. Also, $m$ is the number of the concerning rings and $n$ is the number of data points in one concerning ring in the matrix. $\mathcal{E}$ is the error between the output and average matrix and is defined as

$$\mathcal{E} = \sum_{i=1}^{N} |x_i - y_i| , \tag{2.38}$$

where $N$ is the number of all the data points in the matrix. To improve the generation randomness and in view of the error caused by light transmission and reflection, loss threshold $\mathcal{E}_{\mathrm{thr}}$ is introduced herein. Once the loss is less than the threshold, this loss term will be ignored.

*Spray angle evaluator*

The present evaluator is composed of two parts, one is the theoretical model of spray angle, and the other is a CNN encoder to estimate the spray angles from the generated field solutions. The schematic diagram of the theoretical model of spray angle is shown in Figure 2.14. Before carrying out the theoretical analysis, several basic hypotheses are declared (see Appendix A.1).

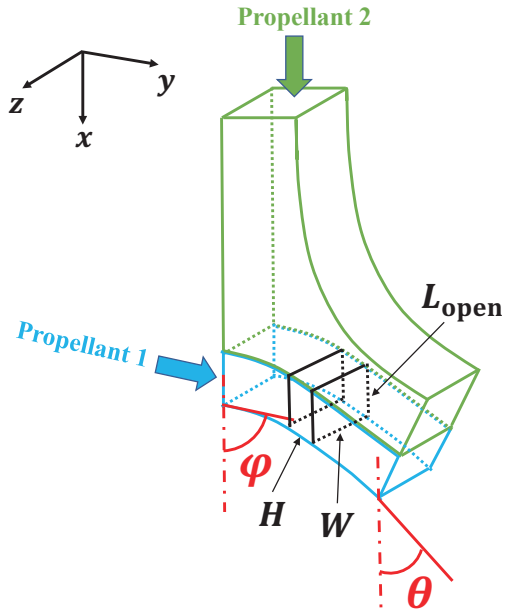



FIGURE 2.14: Schematic of the considering volume in the spray angle model. $L_{\mathrm{open}}$ and $W$ is the length and width of the radical rectangular section, respectively. $H$ is the axial height of the liquid element.

Following the definition in Ref. [103] and Newton's second law, the aerodynamic drag $F_{\mathrm{d}}$ of the liquid element in x-direction is

$$\frac{1}{2}C_{\mathrm{d}}\rho_{\mathrm{g}}\left(v_{\mathrm{g}}-v_{\mathrm{l}}\cos\varphi\right)^{2}WH=\rho_{\mathrm{l}}WL_{\mathrm{open}}H\frac{dv_{\mathrm{x}}}{dt}, \tag{2.39}$$

where $C_{\mathrm{d}}$ is drag coefficient. $v_{\mathrm{g}}$ and $v_{\mathrm{l}}$ are gas velocity and initial liquid element velocity, respectively. $m_l$ and $a$ are the mass and acceleration of liquid element, respectively. $v_{\mathrm{x}}$ is the x-component of the liquid element velocity. After the formula derivation which is detailed in Appendix A.4, the x-coordinate of the element's trajectory is expressed in terms of the momentum ratio $C_{\mathrm{TMR}}$ as follows

$$x=\left[\frac{C_{\mathrm{d}}}{4C_{\mathrm{TMR}}\sin^{2}\varphi}\left(1-\frac{v_{\mathrm{l}}\cos\varphi}{v_{\mathrm{g}}}\right)^{2}+\frac{\cos\varphi}{\sin\varphi}\right]y. \tag{2.40}$$

The theoretical model assumes that the liquid jet does not deform, but in reality, it will deform under aerodynamic forces, which results in a reduction of the effective momentum of the liquid jet. Consequently, the liquid jet deformation factor $\gamma$, which is obtained through the experimental results, is introduced to modify the spray angle theoretical model. In this way, the slope of the liquid jet $\theta$ is obtained as

$$\theta=\gamma\left\{90^{\circ}-\arctan\left[\frac{C_{\mathrm{d}}}{4C_{\mathrm{TMR}}\sin^{2}\varphi}\left(1-\frac{v_{\mathrm{l}}\cos\varphi}{v_{\mathrm{g}}}\right)^{2}+\frac{\cos\varphi}{\sin\varphi}\right]\right\}. \tag{2.41}$$

It is noteworthy that when there is no central propellant deflection and $\varphi=90^{\circ}$, Eq. (2.41) could be simplified as $\theta=\gamma\left[90^{\circ}-\arctan(C_{\mathrm{d}}/4C_{\mathrm{TMR}})\right]$, which formally matches the experimental fitting model $\theta=C_1\arctan(C_2C_{\mathrm{TMR}})$ of liquid-liquid pintle injector in Ref. [104], where $C_1$ and $C_2$ are the fitting parameters.

Similar with the automated scoliosis assessment method in Ref. [105, 106, 107]. A well-trained spray angle estimator is trained to output the angle values from the predictive images in $E_{\mathrm{SA}}$. The architecture of this down-sampling CNN is like $D$, except added one fully-connected layer in the end to output the estimated spray angle $\theta'$. By this way, the loss term from $E_{\mathrm{SA}}$ is calculated as

$$\mathcal{L}_{E_{\mathrm{SA}}}=\left|\theta'-\theta\right|. \tag{2.42}$$

## 2.7 Summary

In this chapter, aiming at the two rocket engine's technical challenges and two key fundamental scientific problems, the governing equations involved are first introduced. In addition, the data set acquisition approaches for the two technical challenges are described. Then, the universal fundamentals and theories of CNN were presented. Finally, for the four different tasks, the corresponding methods were introduced respectively, including a) the basic data-driven method for film cooling field generation, b) the physics-driven method using PDEs as loss function, c) the combined data-driven and physics-driven method with a limited number reference targets, d) the GAN with physical evaluators for spray prediction.

# Chapter 3

# Summaries of publications

In this chapter, the relevant publications of this thesis are briefly summarized.

## 3.1 Supervised learning mixing characteristics of film cooling in a rocket combustor using convolutional neural networks

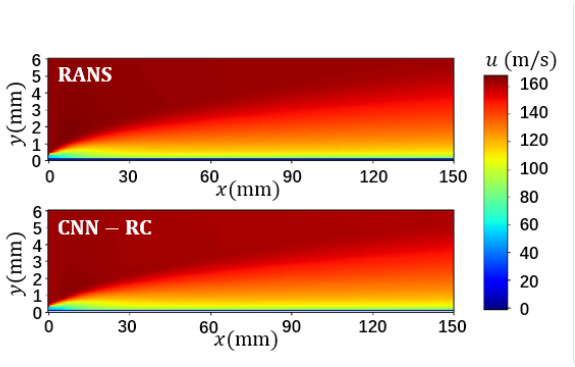
**H. Ma**, Y. X. Zhang, O. J. Haidn, N. Thuerey and X. Y. Hu

### 3.1.1 Summary of the publication

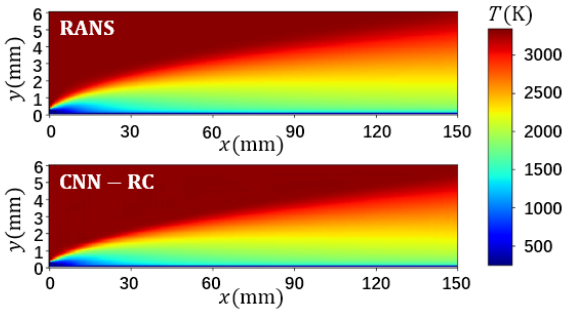
For liquid rocket engines, there is a need to protect the solid surfaces exposed in a high-temperature environment [2]. In common thermal protection methods, film cooling is a technique that can be used in both combustion chambers and nozzles by introducing a portion of fuel along the wall. For film cooling, early studies are mainly carried out with experimental techniques, such as the empirical effectiveness modeling [5], simple discrete layer theory founding [6] and effectiveness data correlations [7]. With the development of the CFD methods and high-performance computing resources, numerical investigations about film cooling were conducted. The results of these practices show that CFD simulation is able to acquire the accurate characteristics of the film cooling flow field. However, the traditional discrete methods used in the numerical simulation are expensive. One complete process to simulate film cooling in rocket engines, considering details, usually costs much time, especially during the meshing and iterative solving phases [108].

The machine learning approach has been applied previously to physical problems such as complex fluid flows. This work presents a method of using convolutional neural networks to directly predict the mixing characteristics between coolant film and combusted gas in a rocket combustion chamber. Based on a reference experiment, numerical solutions are obtained from the Reynolds-Averaged Navier–Stokes simulation campaign and then interpolated into the rectangular grids. A U-net architecture is modified to encode and decode features of the mixing flow field. The influence of training data size and learning time with both normal and re-convolutional loss functions is illustrated. It is noteworthy that when the training set size is small, the re-convolutional method performs much more accurately. To conclude, the re-convolutional methods can decrease iteration steps when aiming for a certain error limitation, and can further get a less time cost when conducting a training task. Also, it can be said that the re-convolutional method is a good choice compared with the normal CNN method to markedly improve the accuracy when it is difficult to obtain a large training data set. By conducting numerical experiments on test cases, the

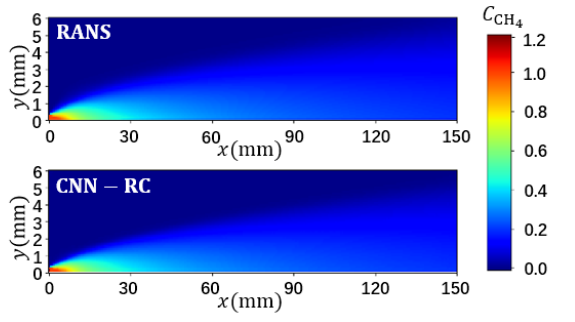
modified architecture and related learning settings are demonstrated with global errors that are less than 0.55%. The results show that the U-net architecture CNN has a very good capability to predict the film cooling flow field accurately.




(A) u




(B) Temperature




(C) Concentratration of $CH_4$

FIGURE 3.1: Comparisons between RANS results and learning results of the baseline case, taking x-component of velocity, temperature, and concentration of $CH_4$ as examples. The predictions made by trained neural networks are almost the same as RANS results. Though the prediction in the vicinity of the film applicator has a relatively bigger difference, the U-net CNN is able to primely predict the overall quantity distribution.

Figure 3.1 shows one of typical learning results.

### 3.1.2 Individual contributions of the candidate

This article [34] was published in the international peer-reviewed journal *Acta Astronautica*. My contribution to this work was the development of the method and

the corresponding computer code for its implementation. I have conducted simulations and numerical experiments, analyzed the results, and written the manuscript for publication.

## 3.2 A Combined Data-driven and Physics-driven Method for Steady Heat Conduction Prediction using Deep Convolutional Neural Networks

**H. Ma**, X. Y. Hu, Y. X. Zhang, N. Thuerey and O. J. Haidn

### 3.2.1 Summary of the publication

With abundant training methods and high-performance computing resources, machine learning has been applied to many scientific research fields, including predicting physics field for reducing or avoiding the large computational cost of the traditional numerical (finite volume/element/difference) methods, such as computational fluid dynamics [108]. In machine learning approaches, data-driven methods were widely used to train the model with a large amount of labeled training data. However, since the training data sets are still generated from traditional numerical solutions [42, 109, 110], it does not truly solve the issue of expansive numerical computation. In addition, in some difficult cases, though a large number of training samples are used, the data-driven method may still not be able to obtain sufficiently accurate solutions. In fact, the unknown physical laws in data-driven methods can be explicitly employed in the learning process [24, 25]. By introducing PDEs into the loss function, a PINN is able to predict the solution that satisfies physics law [26]. The inferred solution is trained to obey the corresponding PDE and boundary conditions. However, the training cost of the physics-driven method is typically more expensive than that of the data-driven method.

In order to remedy the above-mentioned shortcomings, we propose an idea that combines the data- and physics-driven perspectives. Choosing the steady solution of heat conduction as an example, we first consider the original data-driven and physics-driven methods based on a deep CNN respectively and compare their learning progresses for a single case training. It shows that the convergence of the error towards a ground truth solution and the residual of the heat conduction equation exhibit remarkable differences. Based on this observation, we propose a weighted loss function combining the effects of reference target and Laplace equation for training the CNN to predict temperature fields. With this, reference data and physical law are able to simultaneously drive the learning. Then, several numerical experiments are conducted and analyzed to study the improvements achieved by the combined method. For the data-driven based method, the introduction of the physical equation not only is able to speed up the convergence but also produces more physically consistent solutions. From the zoomed-views, the temperature contours obtained from the combined method are much smoother, which suggests the solution is more consistent with physics law. For the physics-driven based method, it is observed that the combined method is able to speed up the convergence up to 49.0% by using a coarse reference. Compared with the original physics-driven method, it is observed that the combined method with all the reference targets, except for zero profiles, is able to obtain a more accurate result by eliminating the large error spot quickly. These results suggest that the requirement for appropriate reference targets is not very restrictive in practical applications.

Figure 3.2 shows one of typical learning results.

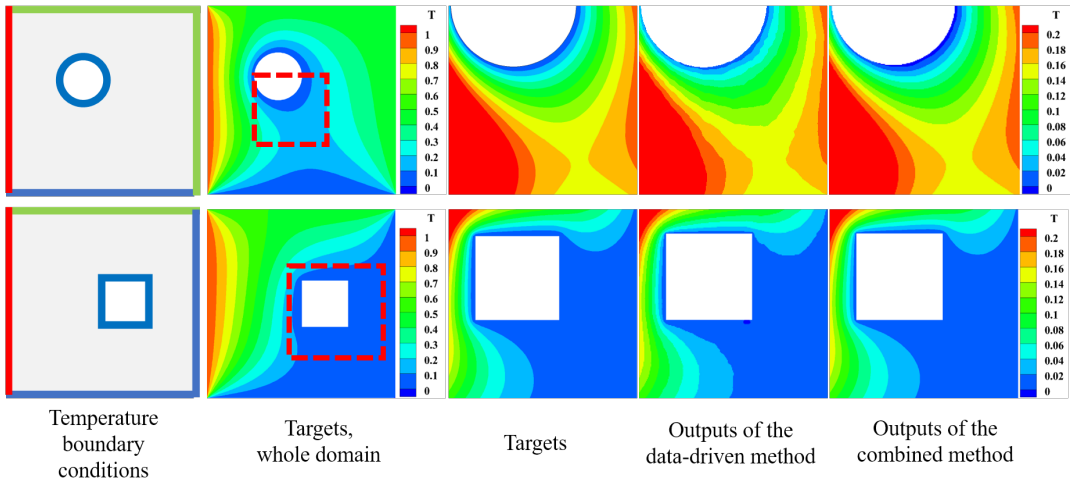



FIGURE 3.2: Enhancements for data-driven method in multiple cases training: outputs of CNN. There are two typical cases from test set with different geometries. The colors in first column represent $T_{\text{boundary}} = 0/0.5/1$. The last three columns show part of the output which marked with the red dot line in the second column. In contrast to the original data-driven method, the temperature contours obtained by combined method are smoother.

### 3.2.2 Individual contributions of the candidate

This article [35] is under review in the international peer-reviewed journal *Journal of Computational Physics*. My contribution to this work was the proposal of the method and the corresponding computer code for its implementation. I have performed numerical experiments, analyzed the results, and written the manuscript for publication.

## 3.3 Physics-driven Learning of the Steady Navier-Stokes Equations using Deep Convolutional Neural Networks

**H. Ma**, Y. X. Zhang, N. Thuerey, X. Y. Hu and O. J. Haidn

### 3.3.1 Summary of the publication

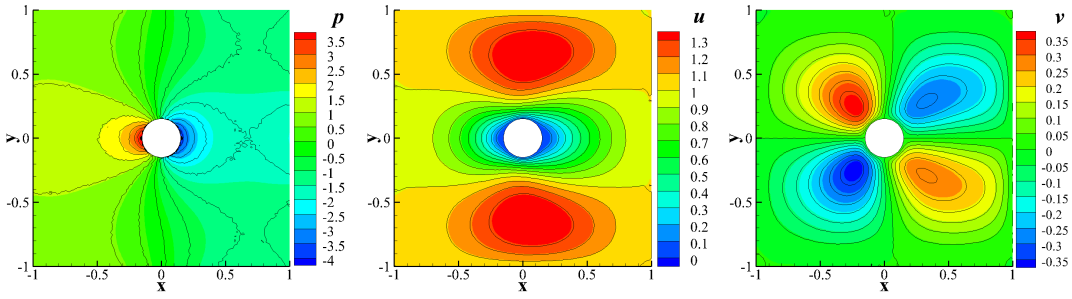
In contrast to classical computational methods, machine learning approaches, and especially the field of deep learning that employs *Neural Networks* (NN), have demonstrated their capabilities to predict flow fields rapidly and accurately [31, 111, 42]. In the multiple machine learning approaches, the physics-driven NN is able to directly obtain the field solution with much less or even no training data. Based on MLP [112], Raissi et al. designed PINN. Due to the constraint of loss function employing partial differential equations, the outputs gradually approach the ones obeying the physics laws [113, 26, 30]. However, due to the full connectivity between the neurons, MLP suffers from extensive memory requirements and statistical inefficiencies [45]. On the contrary, CNN represents a specialized and well-established type of NN to tackle the aforementioned challenges [46].

In this work, we target the physics-driven learning of complex flow fields with high resolutions. We propose the use of convolutional neural networks based on the U-net architecture to efficiently represent and reconstruct the input and output fields, respectively. By introducing Navier-Stokes equations and boundary conditions into loss functions, the physics-driven CNN is designed to predict corresponding steady flow fields directly. In particular, this prevents many of the difficulties associated with approaches employing fully connected neural networks. Several numerical experiments are conducted to investigate the behavior of the CNN approach, and the results indicate that a first-order accuracy has been achieved. Specifically for the case of a flow around a cylinder, different flow regimes can be learned and the adhered “twin-vortices" are predicted correctly. For multiple cases training about flow around a cylinder, the CNN is trained with the variation of Reynolds number while the U-net architecture and inflow conditions are fixed. All the results inside the learning domain are in good agreement with the *Finite Volume Method* (FVM) results. But the values have an “even bias". Compared with the ground truth, all values are closer to the mean value in the whole domain. The numerical results also show that the training for multiple cases is accelerated significantly, especially for the difficult cases at low Reynolds numbers, when limited reference solutions are used as supplementary learning targets.

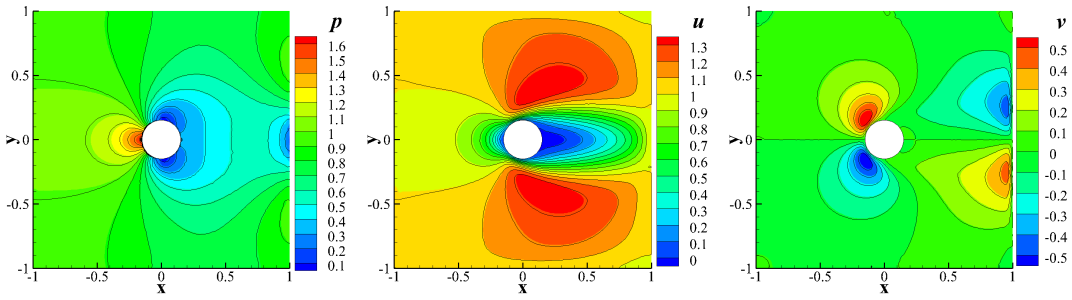
Figure 3.3 shows one of typical learning results.

### 3.3.2 Individual contributions of the candidate

This article [37] is under review in the international peer-reviewed journal *Communications in Computational Physics*. My contribution to this work was the development of the method and the corresponding computer code for its implementation. I have performed numerical experiments, analyzed the results, and written the manuscript for publication.




(A) $Re = 1, p$ (B) $Re = 1, u$ (C) $Re = 1, v$




(D) $Re = 20, p$ (E) $Re = 20, u$ (F) $Re = 20, v$

FIGURE 3.3: Single case when $Re = 1$ and $Re = 20$. Black contour lines are FVM results, colorful contour flooding is PD-CNN results. The learning results agree reference quite well.

## 3.4 Generative Adversarial Networks with Physical Evaluators for Spray Simulation of Pintle Injector

**H. Ma**, B. T. Zhang, C. Zhang and O. J. Haidn

### 3.4.1 Summary of the publication

Due to a wider throttling range and greater combustion stability, pintle injectors are especially suitable for the liquid rocket engines that require deep, fast, and safe throttling [114, 115, 116]. In the practical throttleable engine applications, the pintle is movable to alter the injection area so that the mass flow rate of the injected propellants can be varied continuously according to the economical and safe thrust curve in a given situation [117]. However, using conventional computational fluid dynamics approaches, numerical simulations have to be conducted repeatedly to vary the operating conditions and the computational cost becomes prohibitively expensive [118]. In the previous research, the changes were only considered under discrete condition combinations over a limited number of select operating points [119, 120, 121]. Simulating the complex spray phenomena in the whole range still remains to be a great challenge.

In this work, a novel deep learning approach used to simulate instantaneous spray fields under continuous operating conditions is explored. Based on one specific type of neural networks and the idea of physics constraint, a generative adversarial networks with physical evaluators framework is proposed. The geometry design and mass flux information are embedded as inputs. After the adversarial training between the generator and discriminator, the generated field solutions are fed into the two physics evaluators. In this framework, a mass conversation evaluator is designed to improve the training robustness and convergence. And the spray angle evaluator, which is composed of a down-sampling CNN and theoretical model, guides the networks generating the spray solutions more closely according to the injection conditions. According to the comparison between the simulated and experimental spray morphology under different operating conditions, the generated spray is fairly comparable with the experimental results, meanwhile, the hollow-cone-shaped profiles with a specific spray angle are well reproduced. The typical spray morphology, i.e. liquid column formation, breakup of the column, lateral expansion of the spray, can be clearly noted. The generated spray field shows that the droplets experience a size reduction before approaching a uniform tiny size distribution because of the impingement and collision. The curves of the spray angle versus momentum ratio at different throttling levels also suggest that the GAN-PE results coincide with the experimental results well in a wide momentum ratio range. The work suggests the great potential for prior physics knowledge employment in the simulation of instantaneous flow fields.

Figure 3.4 shows one of typical learning results.

### 3.4.2 Individual contributions of the candidate

This article [38] was published in the international peer-reviewed journal *AIP Advances*. My contribution to this work was the proposal of the method and the corresponding computer code for its implementation. I have performed numerical experiments, analyzed the results, and written the manuscript for publication.

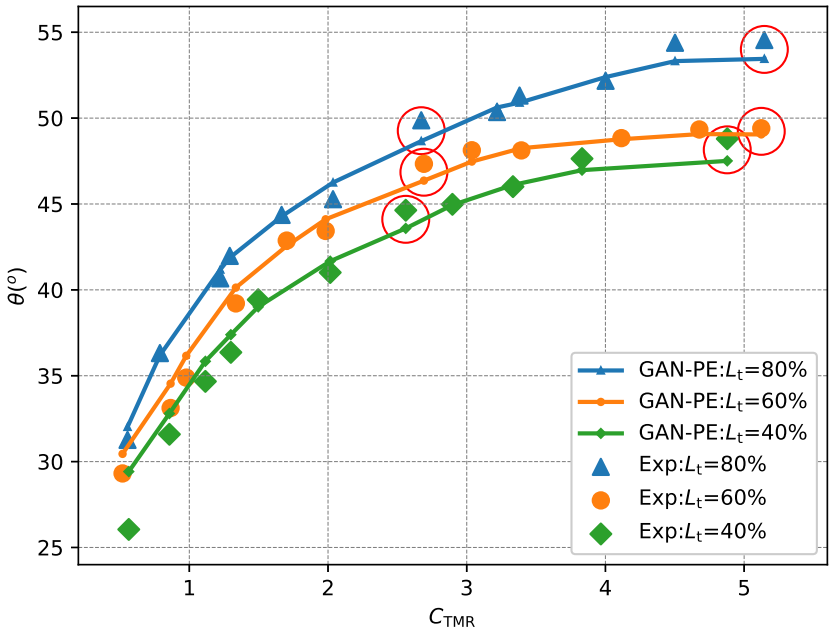



FIGURE 3.4: The comparison of spray angle between GAN-PE and experiments. The red circles show the cases which are out of the learning domain. It can be observed that the GAN-PE results coincide with the experimental results well in a wide momentum ratio range. The simulated solutions also represent the experimental conclusion that the spray angle is mainly determined by the momentum ratio.

# Chapter 4

# Discussions and outlooks

In this thesis, four variants of the deep learning method have been proposed to predict the field solution of film cooling in a combustion chamber, heat conduction of square plates, flow around a cylinder, and injection spray of a pintle injector using convolutional neural networks. Aiming at the practical technical challenges in rocket engines and involving fundamental scientific problems, a data-driven encoder-decoder, a physics-driven method with PDE loss, a combined data-driven and physics-driven method, and a generative adversarial network with physical evaluators are proposed and developed. The objectives have been accomplished successfully and demonstrated in the Paper I - Paper IV. Here, the related literature and the concluding remarks are discussed. And several possibilities of the potential applications and future research are presented.

## 4.1 Discussions

Neural networks have been used to evaluate physics fields for over three decades from the early convective heat transfer evaluation using experimentally based coefficients [122]. Modern neural networks have shown an excellent promise and capability to capture physics characteristics. Based on the generative adversarial networks, Farimani directly predicted the transport phenomena of steady-state heat conduction and incompressible fluid flow [33]. Wu proposed a deep generative Markov state model learning framework for inference of metastable dynamical systems, and the model demonstrated can generate valid distributions for molecular dynamics [123]. Bhatnagar trained an approximation model based on convolutional neural networks which can be used to predict the velocity and pressure field when given the pixelated shape of the object [124]. Utilizing the closure models trained by labeled data to predict the effects of various single and combined physical phenomena has become an attractive research topic. Ling proposed a neural network architecture with a tensor basis embedded by Galilean invariance to model turbulence. The predictions show a significant improvement over the feed-forward MLP. Wang utilized a data-driven machine learning approach for reconstructing discrepancies in RANS modeled Reynolds stresses and excellent predictive performance was observed [125]. Wu presented an extension of Wang which enables the machine learning model to yield improved predictions of Reynolds stresses and mean velocities [126]. The data-driven approaches described above were meant to be an efficient alternative to replace the expensive processes of numerical computation [127, 128]. However, the training data sets for physics field prediction are usually obtained by traditional numerical solutions [36, 42, 109, 110] or even more costly experimental results [122, 129, 130], which seems like an embarrassing loop that does not truly solve the demand.

Based on MLP [112], Raissi et al. designed physics-informed neural networks in 2019 [26]. After that, similar physics-informed/based/constrained methods without training data have been further studied. Sun incorporated the governing PDEs to the loss function and eventually obtained the solution of the flow field without training data. Since the input of the deep NN is only a one-dimensional array, a specific prior ansatz has to be devised to enforce the initial and boundary conditions [131]. Similar work are presented in Ref. [26, 30, 113, 132]. However, due to the full connectivity between the neurons, MLP suffers from extensive memory requirements and statistical inefficiencies [45]. Therefore, it is difficult to handle well the multi-dimensional learning space with high-resolution physics fields containing much more details. In addition, MLP architecture by itself does not take into account the spatial structure of data. The data points in the learning domain irrespective of their distance are treated in the same way [48]. However, the physics laws represented with PDEs are based on the localities of data points, which suggests that the capability of NN to reflect this spatial relationship can be very important, especially for the physics-driven methods which are constrained only by PDEs.

In contrast, CNNs have shown the ability to directly transform the learning domain by employing square or cubic kernels. There is some data-driven learning research using CNN. Sharma applied a weak-supervising paradigm to the physical system governed by non-linear differential equations. The convolutional kernels used to form PDE were beforehand set or trained by a small-size data set [53]. Zhu employed a convolutional encoder-decoder neural network approach for solving PDEs and constructing a surrogate model without labeled data which yielded responses similar to those of data-driven models [54]. Geneva proposed a novel autoregressive dense encoder-decoder CNN to model and solve non-linear dynamical systems, which potentially decreased the computational cost [133].

Based on the physics-constraint idea and CNN fundamentals, Paper II [35] and Paper III [37] proposed a physics-driven deep learning framework using CNN to predict the physical field solutions. By introducing the discretized Laplace/Navier-Stokes equation(s) and boundary conditions as the loss function, the CNN is able to directly predict steady-state heat conduction/laminar flow fields. Compared with the MLP used in previous research, the CNN is able to embed the objective geometry and flow structure into the latent space, and then decode them to reconstruct the corresponding flow fields with physics consistency. The PD-CNN is able to obtain accurate solutions for both single case and multiple cases.

Also, to remedy the shortcomings of the original methods, we proposed a combined data- and physics-driven method based on the PD-CNN to directly predict the field solution of physics problems. The combined method simultaneously utilizes training data and physics law to drive the learning process. For the data-driven based method, besides accelerated convergence, the obtained local structure of the solution achieves a better physics consistency. For the physics-driven based method, the learning process can be accelerated considerably even with not very restrictive choices of reference targets, which is useful for practical application when accurate references are not available. Note that the related CNN architecture and the chosen scientific problems are generic and the combined method is potentially suitable for other physical laws as long as they can be expressed as PDEs.

Besides the fundamental scientific problems, research is also carried out for predicting complex flow fields in practical liquid rocket engines with the existing and proposed methods. In Paper I [34], CNN is used to study the mixing characteristics of the film cooling in a rocket combustor. By modifying the U-net CNN architecture and forming a re-convolutional loss function, the trained model is able to predict

the solution of the flow field in a straightforward way. The work presented in this paper proved the capability of CNN to solve the multiple flows mixing problem by the means of supervised learning with a small sample set. While the work presented is based on the RANS solutions, the modified neural network architecture and the re-convolutional loss function forming method are generic and can be applied to a wide range of partial differential equation problems in a rectangular Cartesian coordinate system.

Similarly with the film cooling, applying the conventional CFD approach to simulate the complex spray phenomena of pintle injectors in a widely throttleable range is a great challenge. Paper IV [38] proposed a novel deep learning framework constrained by physical evaluators to directly predict spray solutions based on generative adversarial networks. The normal discriminator and the mass conservation and spray angle evaluators are used to constrain the CNN to generate the spray solution, including macroscopical morphology and spray angle. The former evaluator is able to improve the training convergence and the latter one helps to obtain more accurate spray angles that are consistent with the operating conditions. It is noteworthy that the related network architecture and spray problem are generic and the proposed framework is potentially suitable for other fluid field simulations which have proper prior physics knowledge.

## 4.2 Outlooks

The present work can be further improved in several directions that are related to both scientific modeling and practical engineering. Some possibilities for future work are:

- In this thesis, the well-trained CNN models, both data- and physics-driven, are actually surrogate models which can be used to easily obtain the key parameters given operating conditions. Therefore, it is of great benefit to employ them for optimization tasks. For example, the thermal topology with heat conduction model and blade shape design with fluid mechanics model.

- The proposed physics-driven CNN was implemented for modeling heat conduction and fluid simulation. It would be interesting that the present method may be extended to simulate the multi-species flow considering chemical reactions, and furthermore employed for solving the practical combustion problems in liquid rocket engines straightforwardly.

- The proposed physics-driven CNN now is only used to simulate the steady-state physics field solutions. The instantaneous spray prediction task also suggests the potential of CNN to consider temporal effects. Future efforts may be done to combine CNN with sequential deep learning methods, such as *Long-Short Term Memory* (LSTM) and transformer, in order to obtain dynamic solutions.

- The present generalized combined data-driven and physics-driven method suggests the potential for practical engineering problems. Furthermore, it is of great benefit to understand whether the present idea that using prior knowledge can be extended to realize other machine learning tasks, such as the *Artificial Intelligence for IT Operations* (AIOps).

# Bibliography


[1] Pedro Simplício, Andrés Marcos, and Samir Bennani. "Reusable launchers: development of a coupled flight mechanics, guidance, and control benchmark". In: *Journal of Spacecraft and Rockets* 57.1 (2020), pp. 74–89.

[2] John P. O'Connor and A. Haji-Sheikh. "Numerical study of film cooling in supersonic flow". In: *AIAA Journal* 30.10 (1992), pp. 2426–2433.

[3] Seong-Ku Kim et al. "Multidisciplinary simulation of a regeneratively cooled thrust chamber of liquid rocket engine: Turbulent combustion and nozzle flow". In: *International Journal of Heat and Mass Transfer* 70 (2014), pp. 1066–1077.

[4] Christoph Ulrich Kirchberger. "Investigation on heat transfer in small hydrocarbon rocket combustion chambers". PhD thesis. Technische Universität München, 2014.

[5] Richard J. Goldstein. "Film cooling". In: *Advances in Heat Transfer*. Vol. 7. Elsevier, 1971, pp. 321–379.

[6] Bryan Richards and J. L. Stollery. "Laminar film cooling experiments in hypersonic flow". In: *Journal of Aircraft* 16.3 (1979), pp. 177–181.

[7] Robert Bass et al. "Supersonic film cooling". In: *2nd International Aerospace Planes Conference*. 1990, p. 5239.

[8] Vinothkumar Sekar et al. "Fast flow field prediction over airfoils using deep learning approach". In: *Physics of Fluids* 31.5 (2019), p. 057103.

[9] Philip Holmes et al. *Turbulence, coherent structures, dynamical systems and symmetry*. Cambridge University Press, 2012.

[10] J. Nathan Kutz et al. *Dynamic mode decomposition: data-driven modeling of complex systems*. Society for Industrial and Applied Mathematics, 2016.

[11] Peter Benner, Serkan Gugercin, and Karen Willcox. "A survey of projection-based model reduction methods for parametric dynamical systems". In: *SIAM Review* 57.4 (2015), pp. 483–531.

[12] Clarence W. Rowley, Tim Colonius, and Richard M. Murray. "Model reduction for compressible flows using POD and Galerkin projection". In: *Physica D: Nonlinear Phenomena* 189.1-2 (2004), pp. 115–129.

[13] Jan S. Hesthaven and Stefano Ubbiali. "Non-intrusive reduced order modeling of nonlinear problems using neural networks". In: *Journal of Computational Physics* 363 (2018), pp. 55–78.

[14] Qian Wang, Jan S. Hesthaven, and Deep Ray. "Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to a combustion problem". In: *Journal of computational physics* 384 (2019), pp. 289–307.

[15] Marti A. Hearst et al. "Support vector machines". In: *IEEE Intelligent Systems and Their Applications* 13.4 (1998), pp. 18–28.

[16] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[17] Giuseppe Carleo et al. "Machine learning and the physical sciences". In: *Reviews of Modern Physics* 91.4 (2019), p. 045002.

[18] Wentao Yan et al. "Data-driven multi-scale multi-physics models to derive process–structure–property relationships for additive manufacturing". In: *Computational Mechanics* 61.5 (2018), pp. 521–541.

[19] Ze Jia Zhang and Karthikeyan Duraisamy. "Machine learning methods for data-driven turbulence modeling". In: *22nd AIAA Computational Fluid Dynamics Conference*. 2015, p. 2460.

[20] Trenton Kirchdoerfer and Michael Ortiz. "Data-driven computational mechanics". In: *Computer Methods in Applied Mechanics and Engineering* 304 (2016), pp. 81–101.

[21] Vasileios Christelis and Aristotelis Mantoglou. "Physics-based and data-driven surrogate models for pumping optimization of coastal aquifers". In: *European Water* 57 (2017), pp. 481–488.

[22] Meekyoung Kim et al. "Data-driven physics for human soft tissue animation". In: *ACM Transactions on Graphics (TOG)* 36.4 (2017), p. 54.

[23] Peter J. Schmid. "Dynamic mode decomposition of numerical and experimental data". In: *Journal of Fluid Mechanics* 656 (2010), pp. 5–28.

[24] Hyuk Lee and In Seok Kang. "Neural algorithm for solving differential equations". In: *Journal of Computational Physics* 91.1 (1990), pp. 110–131.

[25] Isaac E. Lagaris, Aristidis Likas, and Dimitrios I. Fotiadis. "Artificial neural networks for solving ordinary and partial differential equations". In: *IEEE Transactions on Neural Networks* 9.5 (1998), pp. 987–1000.

[26] Maziar Raissi, Paris Perdikaris, and George E. Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics* 378 (2019), pp. 686–707. DOI: 10.1016/j.jcp.2018.10.045.

[27] Alexandre M. Tartakovsky et al. "Learning parameters and constitutive relationships with physics informed deep neural networks". In: *arXiv preprint arXiv:1808.03398* (2018). eprint: 1808.03398v2.

[28] Qizhi He and Jiun-Shyan Chen. "A physics-constrained data-driven approach based on locally convex reconstruction for noisy database". In: *arXiv preprint arXiv:1907.12651* (2019). eprint: 1907.12651v3.

[29] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. "Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations". In: *arXiv preprint arXiv:1711.10561* (2017).

[30] Lu Lu et al. "DeepXDE: A deep learning library for solving differential equations". In: *SIAM Review* 63.1 (2021), pp. 208–228.

[31] Steven L. Brunton, Bernd R. Noack, and Petros Koumoutsakos. "Machine learning for fluid mechanics". In: *Annual Review of Fluid Mechanics* 52 (2019). eprint: 1905.11075v3.

[32] Nils Thuerey et al. "Deep learning methods for Reynolds-averaged Navier-Stokes simulations of airfoil flows". In: *arXiv preprint arXiv:1810.08217* (2018). eprint: 1810.08217v2.

[33] Amir Barati Farimani, Joseph Gomes, and Vijay S. Pande. "Deep learning the physics of transport phenomena". In: *arXiv preprint arXiv:1709.02432* (2017).

[34] Hao Ma et al. "Supervised learning mixing characteristics of film cooling in a rocket combustor using convolutional neural networks". In: *Acta Astronautica* 175 (2020), pp. 11–18. ISSN: 0094-5765.

[35] Hao Ma et al. "A combined data-driven and physics-driven method for steady heat conduction prediction using deep convolutional neural networks". In: *arXiv preprint arXiv:2005.08119* (2020).

[36] Julia Ling, Andrew Kurzawski, and Jeremy Templeton. "Reynolds averaged turbulence modelling using deep neural networks with embedded invariance". In: *Journal of Fluid Mechanics* 807 (2016), pp. 155–166.

[37] Hao Ma et al. "Physics-driven learning of the steady Navier-Stokes equations using deep convolutional neural networks". In: *arXiv preprint arXiv:2106.09301* (2021).

[38] Hao Ma et al. "Generative adversarial networks with physical evaluators for spray simulation of pintle injector". In: *AIP Advances* 11.7 (2021), p. 075007.

[39] Stephen B Pope. *Turbulent flows*. IOP Publishing, 2001.

[40] W. P. Jones and Brian Edward Launder. "The prediction of laminarization with a two-equation model of turbulence". In: *International Journal of Heat and Mass Transfer* 15.2 (1972), pp. 301–314.

[41] Maria Palma Celano et al. "Comparison of single and multi-injector GOC/CH4 combustion chambers". In: *52nd AIAA/SAE/ASEE Joint Propulsion Conference*. 2016, p. 4990.

[42] Nils Thuerey et al. "Deep learning methods for Reynolds-averaged Navier–Stokes simulations of airfoil flows". In: *AIAA Journal* 58.1 (2020), pp. 25–36.

[43] Nikolaos Perakis and Oskar J. Haidn. "Inverse heat transfer method applied to capacitively cooled rocket thrust chambers". In: *International Journal of Heat and Mass Transfer* 131 (2019), pp. 150–166.

[44] Nikolaos Perakis, Maria Palma Celano, and Oskar J. Haidn. "Heat flux and temperature evaluation in a rectangular multi-element GOX/GCH4 combustion chamber using an inverse heat conduction method". In: *7th European Conference for Aerospace Sciences*. 2017.

[45] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, 2016.

[46] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in Neural Information Processing Systems* 25 (2012), pp. 1097–1105.

[47] Matthew Browne, Saeed Shiry Ghidary, and Norbert Michael Mayer. "Convolutional neural networks for image processing with applications in mobile robotics". In: *Speech, Audio, Image and Biomedical Signal Processing using Neural Networks*. Springer, 2008, pp. 327–349.

[48] Hong Hai Le, Ngoc Hoa Nguyen, and Tri-Thanh Nguyen. "Automatic detection of singular points in fingerprint images using convolution neural networks". In: *Asian Conference on Intelligent Information and Database Systems*. Springer. 2017, pp. 207–216.

[49] Aharon Azulay and Yair Weiss. "Why do deep convolutional networks generalize so poorly to small image transformations?" In: *arXiv preprint arXiv:1805.12177* (2018).

[50] Alex Krizhevsky, Geoffrey Hinton, et al. *Learning multiple layers of features from tiny images*. 2009.

[51] Yann LeCun, Yoshua Bengio, et al. "Convolutional networks for images, speech, and time series". In: *The Handbook of Brain Theory and Neural Networks* 3361.10 (1995), p. 1995.

[52] Asifullah Khan et al. "A survey of the recent architectures of deep convolutional neural networks". In: *Artificial Intelligence Review* 53.8 (2020), pp. 5455–5516.

[53] Rishi Sharma et al. "Weakly Supervised Deep Learning of Heat Transport via Physics Informed Loss". In: *arXiv preprint arXiv:1807.11374* (2018). eprint: 1807.11374v2.

[54] Yinhao Zhu et al. "Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data". In: *Journal of Computational Physics* 394 (2019), pp. 56–81. eprint: 1901.06314v1.

[55] Philipp Holl, Vladlen Koltun, and Nils Thuerey. "Learning to control PDEs with differentiable physics". In: *International Conference on Learning Representations (ICLR)*. 2020.

[56] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.

[57] Coenraad Mouton, Johannes C Myburgh, and Marelie H Davel. "Stride and translation invariance in CNNs". In: *Southern African Conference for Artificial Intelligence Research*. Springer. 2021, pp. 267–281.

[58] Min Lin, Qiang Chen, and Shuicheng Yan. "Network in network". In: *arXiv preprint arXiv:1312.4400* (2013).

[59] Waseem Rawat and Zenghui Wang. "Deep convolutional neural networks for image classification: A comprehensive review". In: *Neural Computation* 29.9 (2017), pp. 2352–2449.

[60] Zhe Liu et al. "Owl eyes: Spotting ui display issues via visual understanding". In: *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE. 2020, pp. 398–409.

[61] Marc'Aurelio Ranzato et al. "Unsupervised learning of invariant feature hierarchies with applications to object recognition". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2007, pp. 1–8.

[62] Dominik Scherer, Andreas Müller, and Sven Behnke. "Evaluation of pooling operations in convolutional architectures for object recognition". In: *International Conference on Artificial Neural Networks*. Springer. 2010, pp. 92–101.

[63] Chen-Yu Lee, Patrick W. Gallagher, and Zhuowen Tu. "Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree". In: *Artificial Intelligence and Statistics*. PMLR. 2016, pp. 464–472.

[64] Kaiming He et al. "Spatial pyramid pooling in deep convolutional networks for visual recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.9 (2015), pp. 1904–1916.

[65] Tao Wang et al. "End-to-end text recognition with convolutional neural networks". In: *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. IEEE. 2012, pp. 3304–3308.

[66] Zewen Li et al. "A survey of convolutional neural networks: analysis, applications, and prospects". In: *IEEE Transactions on Neural Networks and Learning Systems* (2021).

[67] Kaiming He et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

[68] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological Review* 65.6 (1958), p. 386.

[69] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[70] Vinod Nair and Geoffrey E. Hinton. "Rectified linear units improve restricted boltzmann machines". In: *International Conference on Machine Learning*. 2010.

[71] Guifang Lin and Wei Shen. "Research on convolutional neural network based on improved Relu piecewise activation function". In: *Procedia computer science* 131 (2018), pp. 977–984.

[72] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International Conference on Machine Learning*. PMLR. 2015, pp. 448–456.

[73] Geoffrey E. Hinton et al. "Improving neural networks by preventing co-adaptation of feature detectors". In: *arXiv preprint arXiv:1207.0580* (2012).

[74] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[75] Ross Girshick. "Fast r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[76] Ning Qian. "On the momentum term in gradient descent learning algorithms". In: *Neural Networks* 12.1 (1999), pp. 145–151.

[77] Diederik P. Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[78] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016).

[79] Shikhar Sharma, Ryan Kiros, and Ruslan Salakhutdinov. "Action recognition using visual attention". In: *arXiv preprint arXiv:1511.04119* (2015).

[80] Tom Sercu et al. "Very deep multilingual convolutional neural networks for LVCSR". In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2016, pp. 4955–4959.

[81] Rikiya Yamashita et al. "Convolutional neural networks: an overview and application in radiology". In: *Insights into imaging* 9.4 (2018), pp. 611–629.

[82] George E. Dahl, Tara N. Sainath, and Geoffrey E. Hinton. "Improving deep neural networks for LVCSR using rectified linear units and dropout". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2013, pp. 8609–8613.

[83] Sepp Hochreiter. "The vanishing gradient problem during learning recurrent neural nets and problem solutions". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6.02 (1998), pp. 107–116.

[84] Hao Li et al. "Visualizing the Loss Landscape of Neural Nets". In: *Advances in Neural Information Processing Systems* 31 (2018).

[85] Gao Huang et al. "Densely connected convolutional networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 4700–4708.

[86] Nikolas Adaloglou. "Intuitive explanation of skip connections in deep learning". In: *https://theaisummer.com/* (2020). URL: `https://theaisummer.com/skip-connections/`.

[87] Hong Hai Hoang and Hoang Hieu Trinh. "Improvement for Convolutional Neural Networks in Image Classification Using Long Skip Connection". In: *Applied Sciences* 11.5 (2021), p. 2092.

[88] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical Image Computing and Computer-assisted Intervention*. Springer. 2015, pp. 234–241.

[89] Olaf Ronneberger et al. "FR-Ro-GE: SUMMARY". In: ().

[90] Adam Paszke et al. "PyTorch: An imperative style, high-performance deep learning library". In: *Advances in Neural Information Processing Systems*. 2019, pp. 8024–8035.

[91] Zichao Long et al. "Pde-net: Learning pdes from data". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 3208–3216.

[92] Zichao Long, Yiping Lu, and Bin Dong. "PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network". In: *Journal of Computational Physics* 399 (2019), p. 108925.

[93] Sankaran Panchapagesan et al. "Multi-task learning and weighted cross-entropy for DNN-based keyword spotting". In: *Interspeech*. Vol. 9. 2016, pp. 760–764.

[94] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 779–788. eprint: `1506.02640v5`.

[95] Huy Phan et al. "DNN and CNN with weighted and multi-task loss functions for audio event detection". In: *arXiv preprint arXiv:1708.03211* (2017). eprint: `1708.03211v2`.

[96] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in Neural Information Processing Systems*. 2014, pp. 2672–2680. eprint: `1406.2661v1`.

[97] Junhyuk Kim and Changhoon Lee. "Deep unsupervised learning of turbulence for inflow generation at various Reynolds numbers". In: *Journal of Computational Physics* 406 (2020), p. 109216.

[98] Jin-Long Wu et al. "Enforcing statistical constraints in generative adversarial networks for modeling chaotic dynamical systems". In: *Journal of Computational Physics* 406 (2020), p. 109209.

[99] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein generative adversarial networks". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 214–223.

[100] Xudong Mao et al. "Least squares generative adversarial networks". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2794–2802.

[101] Tero Karras et al. "Progressive growing of gans for improved quality, stability, and variation". In: *arXiv preprint arXiv:1710.10196* (2017).

[102] Mehdi Mirza and Simon Osindero. "Conditional generative adversarial nets". In: *arXiv preprint arXiv:1411.1784* (2014).

[103] Pei-Kuan Wu et al. "Breakup processes of liquid jets in subsonic crossflows". In: *Journal of Propulsion and power* 13.1 (1997), pp. 64–73.

[104] Johann Freeberg and Jacob Hogge. "Spray cone formation from pintle-type injector systems in liquid rocket engines". In: *AIAA Scitech 2019 Forum*. 2019, p. 0152.

[105] Junlin Yang et al. "Development and validation of deep learning algorithms for scoliosis screening using back images". In: *Communications Biology* 2.1 (2019), pp. 1–8.

[106] Yunliang Cai et al. *Computational methods and clinical applications for spine imaging*. Springer, 2020.

[107] Hongbo Wu et al. "Automated comprehensive adolescent idiopathic scoliosis assessment using MVC-Net". In: *Medical Image Analysis* 48 (2018), pp. 1–11.

[108] John David Anderson and J. Wendt. *Computational fluid dynamics*. Vol. 206. Springer, 1995.

[109] Pedro M. Milani et al. "A machine learning approach for determining the turbulent diffusivity in film cooling flows". In: *Journal of Turbomachinery* 140.2 (2018), p. 021006.

[110] Sangseung Lee and Donghyun You. "Data-driven prediction of unsteady flow fields over a circular cylinder using deep learning". In: *arXiv preprint arXiv:1804.06076* (2018). eprint: 1804.06076v3.

[111] Karthik Duraisamy, Gianluca Iaccarino, and Heng Xiao. "Turbulence modeling in the age of data". In: *Annual Review of Fluid Mechanics* 51 (2019), pp. 357–377. eprint: 1804.00183v3.

[112] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. "Multi-column deep neural networks for image classification". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2012, pp. 3642–3649.

[113] Maziar Raissi and George Em Karniadakis. "Hidden physics models: Machine learning of nonlinear partial differential equations". In: *Journal of Computational Physics* 357 (2018), pp. 125–141.

[114] Gordon Dressler and J. Bauer. "TRW pintle engine heritage and performance characteristics". In: *36th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*. 2000, p. 3871.

[115] S. D. Heister. "Pintle injectors". In: *Handbook of Atomization and Sprays*. Springer, 2011, pp. 647–655.

[116] Kazuki Sakaki et al. "Performance evaluation of rocket engine combustors using ethanol/liquid oxygen pintle injector". In: *52nd AIAA/SAE/ASEE Joint Propulsion Conference*. 2016, p. 5080.

[117] Matthew J. Casiano, James R. Hulka, and Vigor Yang. "Liquid-propellant rocket engine throttling: a comprehensive review". In: *Journal of Propulsion and Power* 26.5 (2010), pp. 897–923.

[118] Jian Yu and Jan S. Hesthaven. "Flowfield reconstruction method using artificial neural network". In: *AIAA Journal* 57.2 (2019), pp. 482–498.

[119] Kanmaniraja Radhakrishnan et al. "Lagrangian approach to axisymmetric spray simulation of pintle injector for liquid rocket engines". In: *Atomization and Sprays* 28.5 (2018), pp. 443–458.

[120] Min Son et al. "Verification on spray simulation of a pintle injector for liquid rocket engine". In: *Journal of Thermal Science* 25.1 (2016), pp. 90–96.

[121] Min Son et al. "Numerical study on the combustion characteristics of a fuel-centered pintle injector for methane rocket engines". In: *Acta Astronautica* 135 (2017), pp. 139–149.

[122] K. Jambunathan et al. "Evaluating convective heat transfer coefficients using neural networks". In: *International Journal of Heat and Mass Transfer* 39.11 (1996), pp. 2329–2332.

[123] Hao Wu et al. "Deep generative markov state models". In: *Advances in Neural Information Processing Systems*. 2018, pp. 3975–3984. eprint: 1805.07601v2.

[124] Saakaar Bhatnagar et al. "Prediction of aerodynamic flow fields using convolutional neural networks". In: *Computational Mechanics* (2019), pp. 1–21. eprint: 1905.13166v1.

[125] Jian-Xun Wang, Jin-Long Wu, and Heng Xiao. "Physics-informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on DNS data". In: *Physical Review Fluids* 2.3 (2017), p. 034603.

[126] Jin-Long Wu, Heng Xiao, and Eric Paterson. "Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework". In: *Physical Review Fluids* 3.7 (2018), p. 074602. eprint: 1801.02762v4.

[127] SoHyeon Jeong et al. "Data-driven fluid simulations using regression forests". In: *ACM Transactions on Graphics (TOG)* 34.6 (2015), p. 199.

[128] Jonathan Tompson et al. "Accelerating eulerian fluid simulation with convolutional networks". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 3424–3433. eprint: 1607.03597v6.

[129] Zhe Bai et al. "Data-driven methods in fluid dynamics: Sparse classification from experimental data". In: *Whither Turbulence and Big Data in the 21st Century?* Springer, 2017, pp. 323–342.

[130] Anand Pratap Singh, Shivaji Medida, and Karthik Duraisamy. "Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils". In: *AIAA Journal* (2017), pp. 2215–2227. eprint: 1608.03990v3.

[131] Luning Sun et al. "Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data". In: *arXiv preprint arXiv:1906.02382* (2019). eprint: 1906.02382v2.

[132] Luning Sun et al. "Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data". In: *Computer Methods in Applied Mechanics and Engineering* 361 (2020), p. 112732.

[133] Nicholas Geneva and Nicholas Zabaras. "Modeling the dynamics of PDE systems with physics-constrained deep auto-regressive networks". In: *arXiv preprint arXiv:1906.05747* (2019). eprint: 1906.05747v3.

# Appendix A

# Original journal papers

Here, the peer-reviewed journal articles of the present work are attached.

## A.1 Paper I

H. Ma, Y. X. Zhang, O. J. Haidn, N. Thuerey and X. Y. Hu

### Supervised learning mixing characteristics of film cooling in a rocket combustor using convolutional neural networks

In *Acta Astronautica*, Volume 175, 2020, pp. 11-18, DOI: `https://doi.org/10.1016/j.actaastro.2020.05.021`.



*Contribution*: My contribution to this work was the development of the method and the corresponding computer code for its implementation. I have conducted simulations and the numerical experiments, analyzed the results, and wrote the manuscript for publication.

Acta Astronautica 175 (2020) 11–18

Contents lists available at ScienceDirect

Acta Astronautica

journal homepage: www.elsevier.com/locate/actaastro

# Supervised learning mixing characteristics of film cooling in a rocket combustor using convolutional neural networks

Hao Ma[a,*], Yu-xuan Zhang[b], Oskar J. Haidn[a], Nils Thuerey[c], Xiang-yu Hu[d]

[a] Department of Aerospace and Geodesy, Technical University of Munich, 85748, Garching b., München, Germany
[b] Beijing Aerospace Propulsion Institute, 100076, Beijing, China
[c] Technical University of Munich, 85748, Garching b., München, Germany
[d] Department of Mechanical Engineering, Technical University of Munich, 85748, Garching b., München, Germany

ARTICLE INFO

*Keywords:*
Deep learning
Convolutional neural network
Flow-field prediction
Film cooling
Combustion chamber

ABSTRACT

Machine learning approach has been applied previously to physical problem such as complex fluid flows. This paper presents a method of using convolutional neural networks to directly predict the mixing characteristics between coolant film and combusted gas in a rocket combustion chamber. Based on a reference experiment, numerical solutions are obtained from Reynolds-Averaged Navier–Stokes simulation campaign and then interpolated into the rectangular target grids. A U-net architecture is modified to encode and decode features of the mixing flow field. The influence of training data size and learning time with both normal and re-convolutional loss function is illustrated. By conducting numerical experiments about test cases, the modified architecture and related learning settings are demonstrated with global errors less than 0.55%.

## 1. Introduction

At the booster and upper stage in reusable launcher systems, hydrocarbon propellants are commendable alternatives because of their lower prices and operational costs [1]. Among the hydrocarbon propellant combinations, methane/oxygen has an overall good performance [2]. Higher impulse, lower probability in carbon deposition, simple extractability from natural gas: all of these make methane an exceptional choice for future reusable launcher engines such as BE-4 and Raptor [3]. However, unlike the other propellant combinations, methane/oxygen still needs to be studied [4,5].

For liquid rocket engines, there is a need to protect the solid surfaces exposed in a high-temperature environment [6]. In common thermal protection methods, film cooling is a technique which can be used in both combustion chambers and nozzles by introducing a portion of fuel along the wall. This introduction in the combustion chamber not only protects the wall from high thermal loads, but also from chemical impact. For film cooling, early studies are mainly carried out with experimental techniques, such as the empirical effectiveness modeling [7], simple discrete layer theory founding [8] and effectiveness data correlations [9]. With the development of the *Computational Fluid Dynamics* (CFD) methods and high-performance computing resources, numerical investigations about film cooling were conducted in the applicator design [10], blowing ratio [11,12], incident angle [13], etc. The results of these practices show that CFD simulation is able to acquire the accurate characteristics of film cooling flow field. However, the traditional discrete methods used in numerical simulation are expensive. One complete process to simulate film cooling in rocket engines, considering details, usually costs much time, especially during the meshing and iterative solving phases [14].

In this decade, machine learning has developed rapidly. From the initial machine vision field where so much research has been conducted, machine learning, especially the deep learning approach, has been applied to many scientific research fields, among which fluid mechanics problem is a worthy choice [15–20]. Raissi et al. utilize Gaussian processes and deep fully connected layers to learn a class of physics phenomena including fluid motion [21–23]. Ling et al. propose a tensor basis neural network embedded by Galilean invariance to model turbulence and the predictions show a significant improvement compared with the commonly used CFD methods [24]. Based on the work of Ling, Milani utilizes a random forest method to predict the turbulent diffusivity in the film cooling flows of gas turbine blades [25,26]. The machine learning work presented above is to train the turbulence model firstly, and then still use the traditional discrete methods to do simulation work. In fact, there is a more straight-forward way. For example, Farimani et al. directly generate solutions for steady state heat conduction by forming a conditional generative adversarial network [27]. Also, Thuerey et al. use *Convolutional Neural Networks*

* Corresponding author.
*E-mail address:* hao.ma@tum.de (H. Ma).

https://doi.org/10.1016/j.actaastro.2020.05.021
Received 8 December 2019; Received in revised form 5 April 2020; Accepted 8 May 2020
Available online 20 May 2020

**Nomenclature**

| | |
|---|---|
| $j_m$ | mass flux of mainstream, kg/(m²·s) |
| $j_c$ | mass flux of coolant, kg/(m²·s) |
| $T_m$ | temperature of mainstream, K |
| $T_c$ | temperature of coolant, K |
| $u$ | $x$ component of velocity, m/s |
| $v$ | $y$ component of velocity, m/s |
| $p$ | pressure, Pa |
| $h$ | height, $m$ |
| $R_{BR}$ | the proportion of the film in the boundary row |

*Subscripts*

| | |
|---|---|
| f | film injecting slot |
| cc | combustion chamber |
| BR | boundary row |

(CNN) to infer the solution of airfoil flows when given certain boundary conditions [28]. However, although machine learning practice discussed above has shown the capability to capture the characteristics of the basic fluid mechanics, the application of the machine learning methods in the practical fluid dynamical problems is rare, especially for the liquid rocket engines with a methane/oxygen combination.

In this paper, a deep learning approach using modified CNN and loss function forming method is applied to study the mixing characteristics of film cooling in rocket engines. Given the inlet flow conditions, the flow field in the subscale gaseous methane/oxygen combustion chamber can be directly predicted. The paper is organized as follows. After the introduction, the process of data set generation from RANS simulation is described in Section 2. The machine learning methods, especially the U-net CNN architecture, are introduced in Section 3. The comparison between the different loss functions and the results of the numerical experiments are presented and analyzed in Section 4. Finally, conclusion is drawn in Section 5.

## 2. Data set from simulation

### 2.1. Reference experiment

In the context of the national research program SFB 40 “Fundamental Technologies for the Development of Future Space-Transport-System Components under High Thermal and Mechanical Loads”, research activities, including experiments and simulations, are conducted in order to get a broader data base for methane operating propulsion systems and a deeper understanding of the key phenomena of methane/oxygen combustion and heat transfer. For this purpose, a subscale multi-element gaseous methane () and gaseous oxygen (GOX) combustion chamber using methane as coolant is designed and tested. The reference experiment, which has a pressure level of 2 MPa, aims to expand the knowledge of film cooling systems in methane/oxygen engines for upper stage/orbital applications. More details of the reference experimental facility can be found in Ref. [29].

The multi-injector chamber with film applicator is depicted in Fig. 1, and the main dimensions of the chamber are shown in Table 1.

### 2.2. Simulation setup

Based on the previous research shown in Ref. [30], a two-dimensional planar model was taken, the chamber being symmetric with respect to the vertical middle plane, and half the chamber was modelled in the simulation.

The coaxial injection of the experiment is not modelled in order to reduce the computational costs of the simulation procedure. Assuming that the mixing and reaction process of fuel and oxidizer only happened in the first segment of the chamber, a homogenous combusted gas mixture is fed into the entrance of the second segment instead. Meantime the coolant film is also assumed to be injected from the beginning of the second segment. It means the computational domain only contains the second segment of the chamber.

The components and heat transfer properties of the combusted gas mixture are obtained from NASA's “Chemical Equilibrium with Applications” program (short CEA) which is a common tool utilized in one-dimensional combustion gas representation along the combustion chamber axis. Based on the minimization of free energy and mass-balance constraints, CEA is able to determine the equilibrium compositions for thermodynamic states specified by an assigned temperature and pressure [31]. The 11 species of the combusted gas mixture are $O_2$, O, H, $HO_2$, OH, $H_2$, $H_2O_2$, $H_2O$, $CO_2$, CO, HCO, while coolant gas used as film is $CH_4$. Also, the chemical reaction is not considered in the RANS simulation.

For steady state simulations, a common approach is to use the temperature filed at one certain moment as the boundary condition. This paper utilizes the in-house tool “*RoqFITT*” to calculate the continuous temperature and heat flux field instead of the separated temperature data measured by thermocouples arranged in the chamber wall [32,33].

As for boundary conditions, both of the inlet mainstream and film are constrained with mass flux, the chamber is defined as no slip walls, the outlet pressure is 1.024 bar which accords with the experimental environment. Table 2 lists more details about the settings about the simulation. By the investigation of heat flux and pressure profiles from simulation and experiments in the previous research [30], the numerical modeling described in this section is properly proven. Eventually the simplification reaches a balance between the computational costs

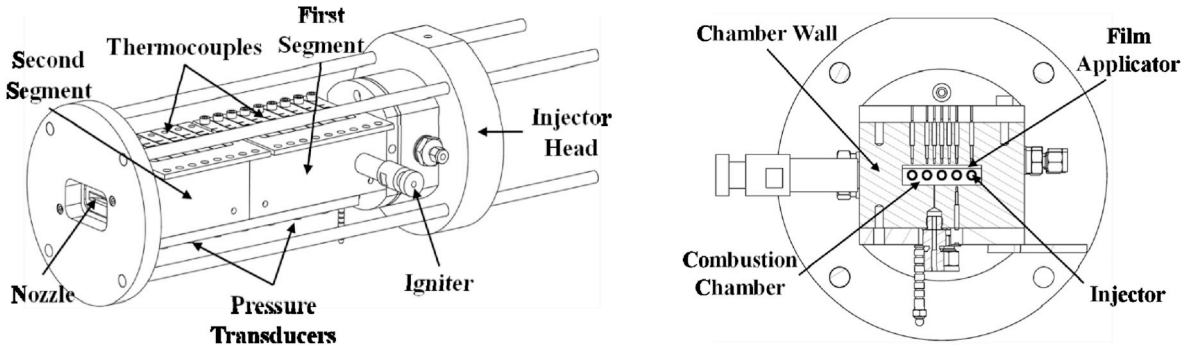



**Fig. 1.** Combustion chamber of the reference experiment.

**Table 1**
Geometric characteristics of the reference experimental hardware.

| Geometric characteristics | Value | Unit |
|---|---|---|
| Chamber length | 277 | [mm] |
| Chamber width | 48 | [mm] |
| Chamber height | 12 | [mm] |
| Throat width | 4.8 | [mm] |
| Throat height | 48 | [mm] |
| Contraction ratio | 2.5 | [] |
| Film applicator height | 0.25 | [mm] |
| Film applicator width | 47 | [mm] |

**Table 2**
Simulation settings.

| Simulation domain | 2-D, half height |
|---|---|
| Mixture ratio (O/F) | 3.0 |
| Inlet mainstream | Mass flux (combusted gas mixture, temperature fixed) |
| Inlet film | Mass flux (methane) |
| Wall temperature | Temperature profile from "*RoqFITT*" |
| Turbulence model | standard k– ε |
| Boundary model | Enhanced wall treatment |
| Chemical mechanism | No reaction |

and accurate mixing characterization.

### 2.3. Simulation campaign

The blowing ratio, which describes the impulse ratio between the mainstream and coolant film, is a crucial parameter when designing the film cooling system in rocket engines [34]. It influences the performance of film cooling significantly [11,12]. When the geometry of the film applicator remains fixed, usually in one actual rocket engine, the blowing ratio varies by injecting coolants with different mass flow rates. In this paper, the chamber and the film applicator keep a unique design, which means the areas to where the hot gas mixture and coolant are injected are fixed, and the initial thickness of the film remains constant as well. So, the variety of mainstream mass flux and coolant mass flux can be used to convey the change of the blowing ratio. It is worthwhile to state that in one actual rocket engine with one certain nozzle design, the variety of the mass flow, especially the mainstream one, usually influences the pressure level of the chamber, eventually changing the performance of the whole engine. In this simulation campaign, aiming to establish a broad data base of the small upper stage engines whose chamber pressure is 2 MPa, the design of nozzle could be regarded as varying to keep the pressure level constant.

The mass fluxes of mainstream and coolant film are generated as random values which are beneficial for the following training work, shown in Table 3, where $j$ denotes mass flux and the subscripts m, denote mainstream, coolant respectively. And the temperatures of the coolant are evenly distributed as six values in the range between 200 K and 300 K; the temperature of the mainstream is 3326.54 K, calculated by CEA. So, the amount of simulation cases is 1176.

### 2.4. Data set acquisition

As the performance of the film cooling far downstream is not as remarkable as that in the vicinity of the applicator, the forepart of simulation domain was chosen as the learning domain for machine learning work; the length of the learning region is 150 mm, while the width is 6 mm, which is the half of the chamber height. The deep learning research in machine vision field usually uses the square-resolution images to adapt to the convolutional neural network architectures. In order to indicate the characteristics of mixing in both dimensions, the learning domain is resampled onto a regular 64 × 256 grid to obtain a data set including inputs and targets.

The three input channels, including mass fluxes of main flow and coolant and coolant temperature, are rearranged as three initial fields, meanwhile, the simulation results of each case are interpolated into a set of Cartesian grids with a same size of 64 × 256.

The value in every row of the input initial fields describes the mass fluxes and temperature. In order to precisely describe the geometry characteristic, we assuming there is a boundary row in the input grid whose values indicate the interface between the mainstream and coolant film. Using $T_c = 300$ K as an example, the value of coolant temperature in the boundary row is calculated by

$$\begin{aligned} T_{c,\,BR} &= T_c \cdot R_{BR} \\ &= T_c \cdot \left( N \cdot \frac{h_f}{h_{cc}} - \mathrm{int}\left( N \cdot \frac{h_f}{h_{cc}} \right) \right) \\ &= 300 \cdot \left( 64 \cdot \frac{2.5\,e-4}{6\,e-3} - \mathrm{int}\left( 64 \cdot \frac{2.5\,e-4}{6\,e-3} \right) \right) \\ &= 200\ K \end{aligned} \tag{1}$$

where $R_{BR}$ denotes the proportion of the film in boundary row. $N$ denotes the number of the total rows and $N = 64$ in this paper. $h_f$ and $h_{cc}$ denote the height of the film injecting slot and the half height of the combustion chamber respectively. By this calculation, the values in the mainstream-film boundary are accurately obtained and the geometry characteristic of the film injecting slot is precisely defined in the grid. As the temperature initial field illustrated in Fig. 2 shows, the bottom three rows present the height of film part of inlet flow in the temperature initial field. Using this method, flow conditions of mainstream and coolant are encoded in a 64 × 256 × 3 grid of values; the first two channels contain mass flux of mainstream and coolant film while the last of the channels contain the temperature of film.

The targets, extracted from the Reynolds-averaged Navier–Stokes simulation (RANS) solution calculated by ANSYS Fluent, include 16 items: both x and y components of velocity, temperature, pressure, and concentrations of the 12 gaseous species from both hot gas mixture and coolant film. So, the data sets for supervised training have the same size with inputs but different channel numbers. The first four channels contain the flow field information including x and y velocity components, pressure, and temperature sequentially. The next twelve channels contain the concentration of every species. From the simulation domain with more than 20 k cells, each target is interpolated into one 64 × 256 matrix, eventually obtaining a 64 × 256 × 16 targets data grid.

## 3. Deep learning method

### 3.1. Pre-processing

Nondimensionalization is a normal data processing method in numerical calculation of fluid mechanics by which the features with different properties can be compared. The involved quantities are usually normalized with respect to the magnitude of the flow, i.e., make them dimensionless. For convenience, we use the corresponding characteristic quantity of inlet flow to represent this magnitude. Thus, the maximum film velocity is chosen to be the characteristic quantity $|v_i|$. So, the dimensionless velocities can be calculated by $\tilde{u}_0 = u_0/|v_i|$ and $\tilde{v}_0 = v_0/|v_i|$. According to the energy equation, the nondimensionalization of pressure could be $\tilde{p}_0 = p_0/|v_i|^2$ in order to remove the quadratic scaling of the values from the target data. The input coolant

**Table 3**
Simulation campaign.

| Variables | Unit | Range | Amount | Values |
|---|---|---|---|---|
| $j_m$ | kg/(m²·s) | 140–400 | 196 | random |
| $j_c$ | kg/(m²·s) | 550–1200 | | |
| $T_c$ | K | 200–300 | 6 | 200,220,240,260,280,300 |

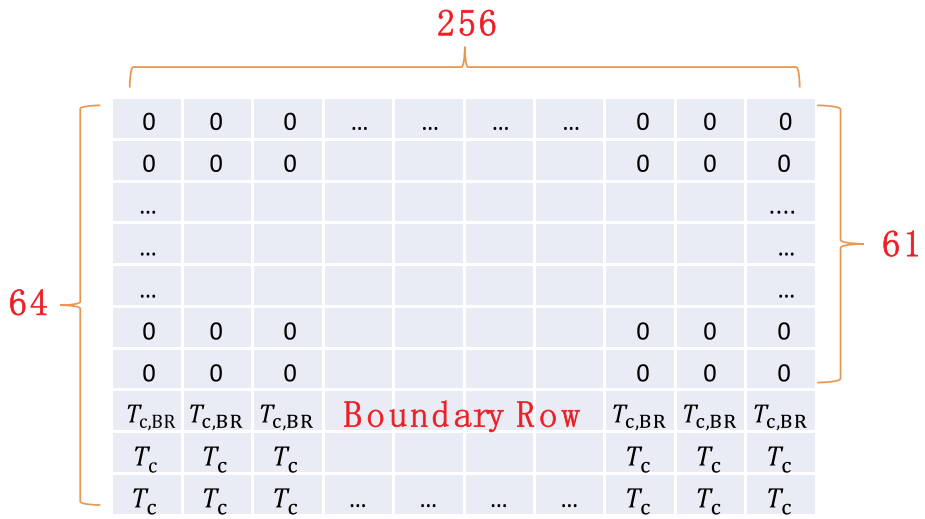



**Fig. 2.** Initial field, coolant temperature as example.

temperature is dimensionless by $\tilde{T}_c = T_c/T_m$, where $T_m = 3326.54$ K denotes the temperature of the mainstream.

In addition, directly using the pressure as targets is an improper choice. It is the pressure gradient rather than the pressure which is involved in the RANS calculation. If we use the pressure directly as one of the targets, the CNN will map the relationship between inputs and pressure which is less correlated. Thus, mean pressure is subtracted from pressure solution and defined as $\hat{p}_0 = \tilde{p}_0 - p_{mean}$, where $p_{mean}$ denotes the mean pressure of all individual pressure samples. With this removal of mean pressure, the learning performance increase by a factor of ca. 4 according to our previous research [28].

Lastly, the values of the quantities in each channel are normalized to the [-1,1] range in order to minimize errors in the training phase. The maximum absolute values of each quantities are found throughout the entire training set, and then the quantities are divided by these maximum values respectively. Before the quotients are transferred into CNN, both inputs and targets are processed in this way. This pre-processing method can flatten the data space and simplify the training task of the deep neural network, eventually accelerate the convergence.

### 3.2. Neural network architecture

The CNN in this paper has a U-net architecture, which is widely used in the field of machine vision. The algorithms are based on the PyTorch platform [35]. By means of convolutional calculation, the features are extracted from the original data set.

At the beginning of the CNN architecture illustrated in Fig. 3, mainstream mass flux, coolant film mass flux, and coolant temperature from the data set are introduced into the architecture as three rectangular matrices whose size is 64 × 256. After two convolutional calculation layers, the original rectangular matrices are transferred into square matrices whose size is 64 × 64 with 32 channels. Then the square filters are utilized until the matrices with only one pixel are obtained. In this encoding process, the matrices of mass fluxes and temperature are progressively down-sampled by convolutional calculations. With the amount of feature channels increasing, abstract and large-scale information is extracted by the neural networks. Then the decoding part, which can be regarded as an inverse convolutional process, mirrors the behavior of the encoding part. The solutions are reconstructed in the up-sampling layers along with the increase of spatial resolution. Eventually rectangular matrices with 16 channels can be obtained, which express the flow field information and concentration information of the gaseous species. It is noteworthy that there are concatenation operations between two different channels as the orange arrows illustrate in Fig. 3, which is actually a “skip connection” process. Concatenating the feature channels from the encoding branch to the

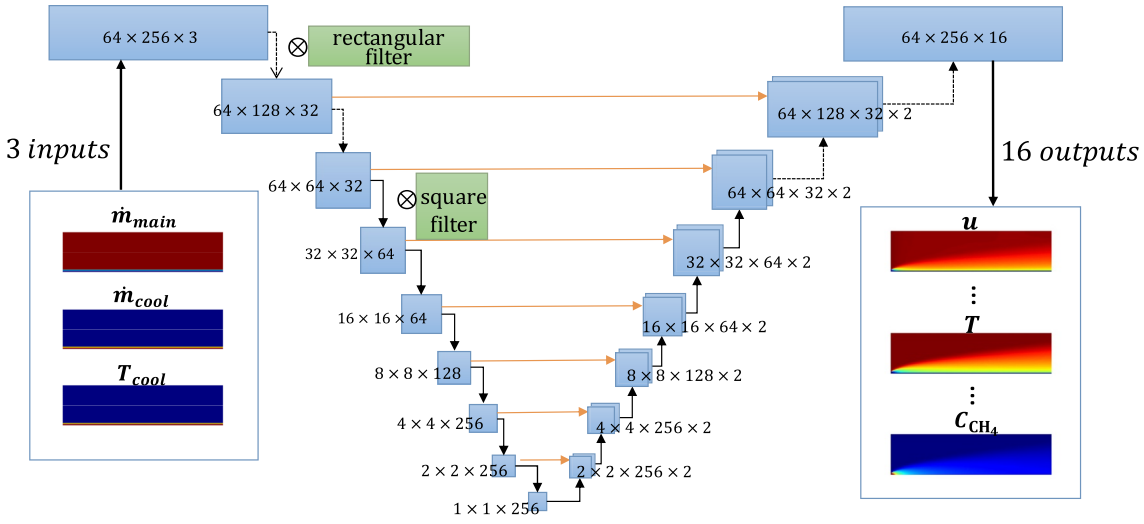



**Fig. 3.** Schematic of U-net architecture.

corresponding decoding branch, the skip connections effectively double the amount of feature channels in every decoding block and enable the neural networks to consider the information from the encoding layers.

Including the inputs layer and outputs layer, the U-net architecture consists of 17 layers and corresponding convolutional blocks. Each convolutional block has a similar structure: batch normalization, active function, convolutional calculation, and dropout. Convolutional blocks (represented by $C$) are usually parametrized by channel factor $c$, convolutional kernel size $k$, and stride $s$. $cX$ shortly denotes $c = X$, the channel number is the product of $c$ and a basic multiplier 32. $kXY$ shortly denotes $k = (X, Y)$ in two dimensions. In addition, $r$ in the block shown below indicates activated by ReLU and $l$ indicates activated by a leaky ReLU [36]. Batch normalization is indicated by $b$.

So, the convolutional blocks in the down-sampling process can be summarized as

$$
\begin{aligned}
&l_0 \rightarrow C(k41,\ s21) \rightarrow l_1\\
&l_1 \rightarrow C(c1,\ k41,\ s21) \rightarrow l_2\\
&l_2 \rightarrow C(c1,\ k44,\ s22,\ l,\ b) \rightarrow l_3\\
&l_3 \rightarrow C(c2,\ k44,\ s22,\ l,\ b) \rightarrow l_4\\
&l_4 \rightarrow C(c2,\ k44,\ s22,\ l,\ b) \rightarrow l_5\\
&l_5 \rightarrow C(c4,\ k22,\ s11,\ l,\ b) \rightarrow l_6\\
&l_6 \rightarrow C(c8,\ k22,\ s11,\ l,\ b) \rightarrow l_7\\
&l_7 \rightarrow C(c8,\ k22,\ s11,\ l,\ b) \rightarrow l_8.
\end{aligned}
$$

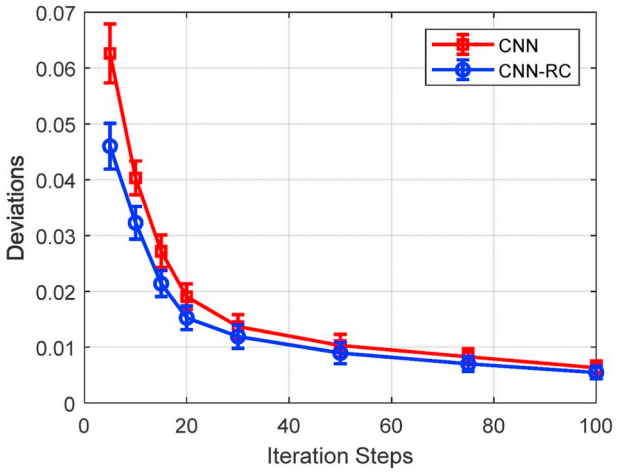
The convolutional blocks in the up-sampling process can be summarized as

$$
\begin{aligned}
&l_8 \rightarrow \mathrm{up}[C(c8, k22, s11, r, b)] \rightarrow l_9\\
&\mathrm{conc}(l_9, l_7) \rightarrow \mathrm{up}[C(c16, k22, s11, r, b)] \rightarrow l_{10}\\
&\mathrm{conc}(l_{10}, l_6) \rightarrow \mathrm{up}[C(c16, k22, s11, r, b)] \rightarrow l_{11}\\
&\mathrm{conc}(l_{11}, l_5) \rightarrow \mathrm{up}[C(c8, k44, s22, r, b)] \rightarrow l_{12}\\
&\mathrm{conc}(l_{12}, l_4) \rightarrow \mathrm{up}[C(c4, k44, s22, r, b)] \rightarrow l_{13}\\
&\mathrm{conc}(l_{13}, l_3) \rightarrow \mathrm{up}[C(c4, k44, s22, r, b)] \rightarrow l_{14}\\
&\mathrm{conc}(l_{14}, l_3) \rightarrow \mathrm{up}[C(c2, k41, s21, r, b)] \rightarrow l_{15}\\
&\mathrm{conc}(l_{15}, l_2) \rightarrow \mathrm{up}[C(c2, k41, s21, r, )] \rightarrow l_{16}.
\end{aligned}
$$

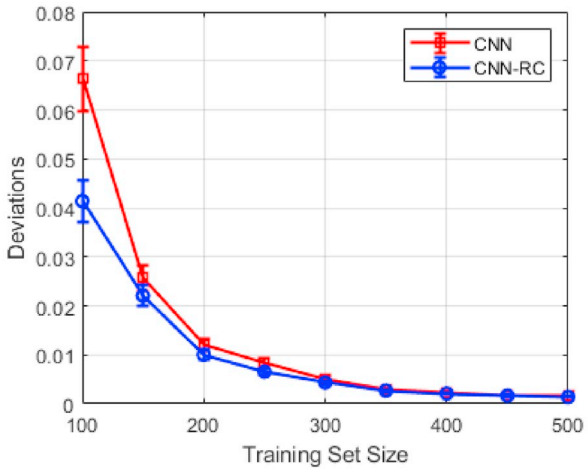
In order to improve accuracy, up-sampling (up []) is used to substitute converse convolutional calculation which is widely used in the research of super-resolution. And "conc ()" denotes concatenation operation. More details of the U-net architecture and CNN can be found in Refs. [28].

### 3.3. Loss function

In the training process, the weights of every convolutional blocks are updated with Adam optimizer [37] by the means of back-propagation, which need one loss function to calculate the difference between results and ground truth. For supervised training, a basic loss function can be defined as




(a) Deviations versus iteration steps (training set size = 200)




(b) Deviations versus training set size (iteration steps = 50)

**Fig. 5.** Comparison of the loss functions.

$$L = \sum_{i=1}^{I} \sum_{n=1}^{N} |X_{\mathrm{out}} - X_{\mathrm{tar}}| \tag{2}$$

where $X$ denotes the quantities which exist in both outputs and targets. $I$ = *targets amount*; meanwhile $N$ = *batch size* denotes the number of cases input into the CNN architecture in one batch operation.

Besides that, we modified the basic loss function by adding a re-convolutional term, so the new loss function is defined as

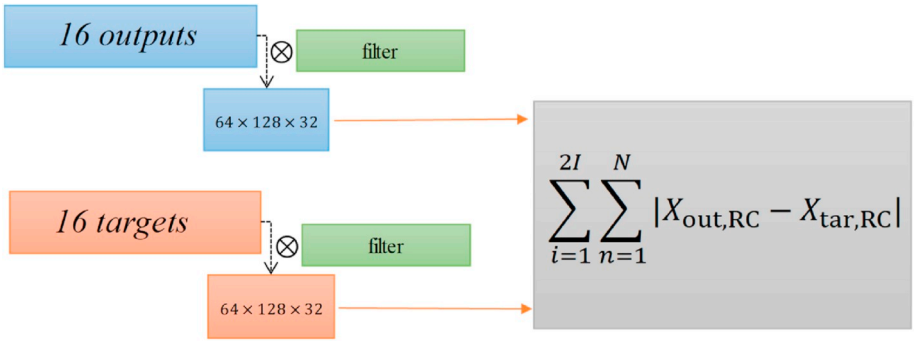



**Fig. 4.** Loss function in re-convolutional approach.

$$L = L_1 + L_2 = \sum_{i=1}^{I}\sum_{n=1}^{N}|X_{\text{out}} - X_{\text{tar}}| + \sum_{i=1}^{2I}\sum_{n=1}^{N}|X_{\text{out,RC}} - X_{\text{tar,RC}}| \quad (3)$$

where $X_{\text{out,RC}} = X_{\text{out}} \otimes filter$, $X_{\text{tar,RC}} = X_{\text{tar}} \otimes filter$, meaning do a convolutional calculation for both outputs and targets with a filter. In this way, the loss is able to directly penalizing gradient differences between the output of CNN and the targets. Different from the second-order central-difference version of the gradient difference loss function proposed in Ref. [38], the re-convolutional method is more accord with the architecture of CNN and easily to be conducted since the grid actually represent the computational domain of film cooling problems. The influence of utilizing such a re-convolutional method to form the loss function on the learning performance will be discussed below. This approach is illustrated in Fig. 4.

## 4. Learning results

The data set is split into the training set and the test set randomly by a ratio of 4/1. We use the samples in training set to train CNN model and then use the boundary conditions from test set to validate the trained model. The cases whose results are analyzed below are all from the test data set if there is no specific declaration. The baseline case discussed in Section 4.2 is one typical case in the test set.

In order to estimate the performance of the learning methods, it is necessary to define a quantitative comparison between learning results and target data. A sum of the errors in every cell for all the output matrices, containing both flow field and concentration information, is a good alternative. The global deviation metric is defined as

$$Deviation = \frac{\sum |x_{\text{out}} - x_{\text{tar}}|}{\sum |x_{\text{tar}}|}, \quad (4)$$

where $x$ denotes the value in every pixel. It is noteworthy that the different random training data may yield to the nondeterminacy of solution. Besides, due to the quite non-linearity of the neural networks method, the different runs with same training samples may also obtain slightly different trained models. Instead of a single run, multiple runs are conducted to check the overall performance. The deviations shown in the next section are the mean of five runs with identical architecture settings apart from the random training data set in order to avoid the effect of non-deterministic calculations.

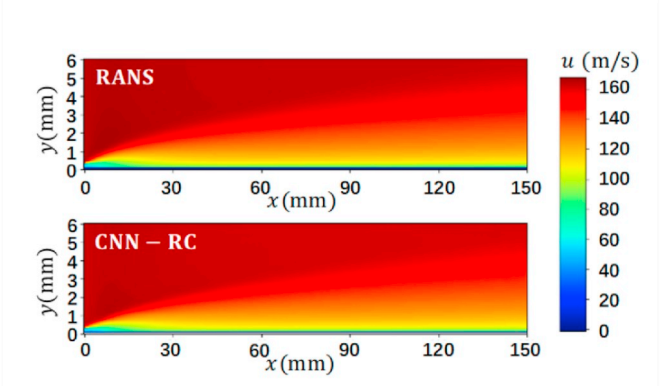
### 4.1. Comparison of the loss functions

In order to compare the training performances between two different loss function forming methods, normal CNN method and re-convolutional CNN method, the training time or training set size varies in the training phase. Firstly, fixing the training set size, the accuracy is tested versus the iteration steps. Fig. 5 (a) shows the comparison of learning results between the two methods. With an increase of iteration steps, both deviations of CNN and re-convolutional CNN methods decrease rapidly until they trend towards stability. Improvement can be achieved by utilizing the re-convolutional methods when the iteration steps are few, while the difference between two methods becomes smaller when the iteration steps increase. The decreasing trend of deviations with training set size increasing is similar (shown in Fig. 5 (b)). With the increase of training set size, the deviations eventually tend to be uniform. It is noteworthy that when the training set size is small, the re-convolutional method performs much more accurately. To conclude, the re-convolutional methods can decrease iteration steps when aiming for a certain error limitation, and further can get a less time cost when conducting a training task. Also, it can be said that the re-convolutional method is a good choice compared with the normal CNN method to markedly improve the accuracy when it is difficult to obtain a large training data set.

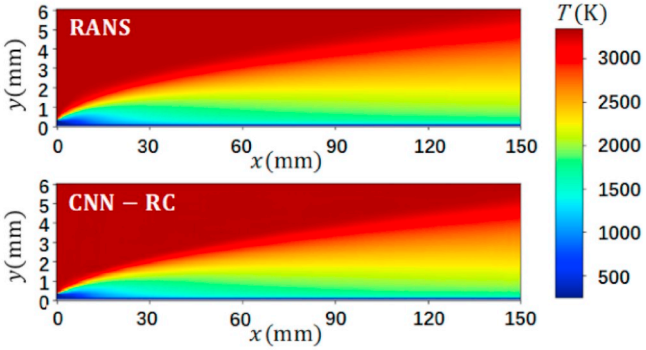
Although the existing machine learning methods which have been applied in practical engineering problems are still relying on the large amount of training data, there are more advanced algorithms attempting to obtain authentic results by a moderate amount of labeled training data [39,40]. This advantageous attempt of re-convolutional CNN methods can help to improve the learning performance with limited training samples in future work.

## 5. Results and analysis

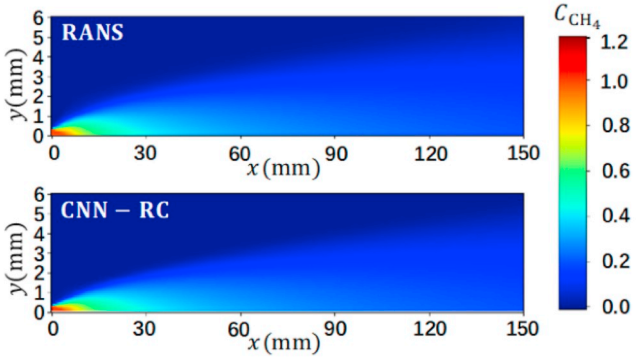
The baseline case ($j_m$ = 240 kg/(m²·s), $j_c$ = 800 kg/(m²·s), $T_c$ = 260 K) describes one typical film cooling design for small upper stage engines. This case is a typical one in the test set and appropriate to estimate the predictive performance of the trained neural networks model. The learning results are calculated from the fixed trained model which is obtained by the validated CNN-RC method. Taking x-component of velocity, temperature




(a) u

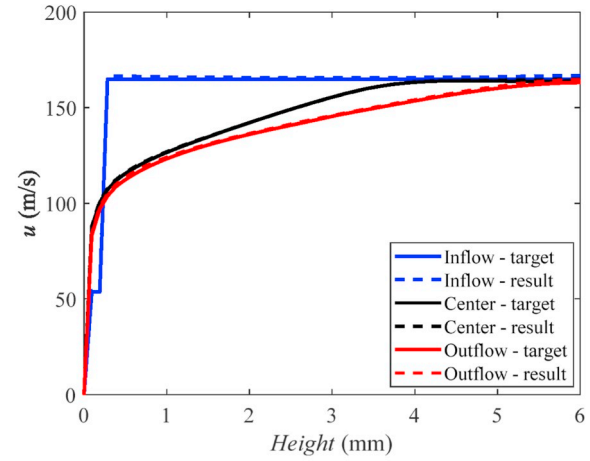



(b) Temperature

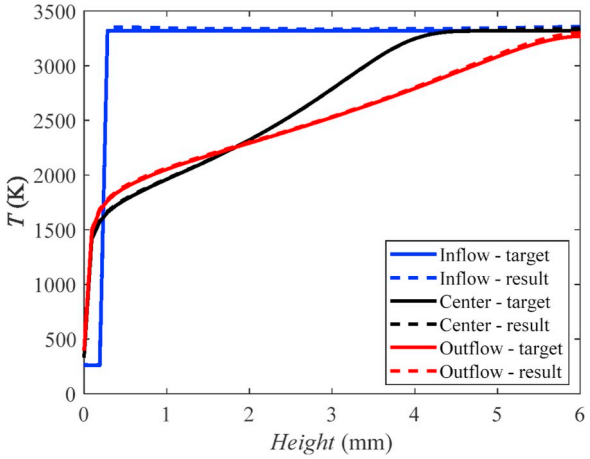



(c) Concentration of $CH_4$

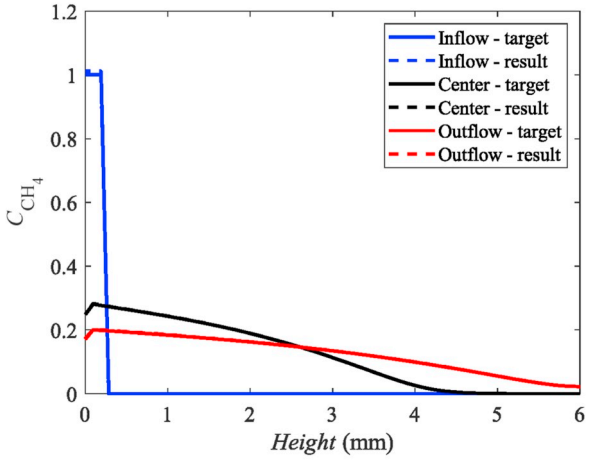
**Fig. 6.** Comparison between simulation results and learning results of domain.




(a) u




(b) Temperature




(c) Concentration of $CH_4$

**Fig. 7.** Comparison between simulation results and learning results of detection lines.

and concentration of $CH_4$ as examples, the comparisons between RANS results and learning results of the baseline case are shown in Fig. 6. The predictions made by trained neural networks are almost the same with RANS results. Though the prediction in the vicinity of film applicator has a relatively bigger difference, the overall trend about the concentration of each species can be primely predicted.

We also picked up three key detection lines in the flow field of baseline case to compare the targets from RANS solution and the learning results from our deep learning algorithm. The inflow, center and outflow shown in Fig. 7 denote the first, the 128th and the last column of the learning domain respectively. The learning results fit targets quite well especially in the central area. The results show that the U-net architecture CNN has a very good capability to predict the film cooling flow field accurately.

In addition, in order to describe the degree that outputs satisfy the real solution, we defined the global error which is the absolute mean value of errors in all elements between the CNN outputs and target. The deviations between prediction and simulation results of the test cases are shown in Fig. 8. All the deviations are under $5.5 \times 10^{-3}$. The data points with different colors show that the coolant temperature has little influence on the learning accuracy. However, along with the increase of blowing ratio $F$ (defined as $F = {\rho_c u_c}/{\rho_m u_m} = {j_c}/{j_m}$) [34], the deviations firstly decrease then increase. By means of statistics, the reason for this trend might be the distribution of the training data. The most amount of training data is in the vicinity of 2 and 3, where the deviations reaches minimum values. Along with the decrease of training data amount, the deviations get bigger.

The potential factors causing this may deduced by Fig. 9. The insufficiency of the training data results in the higher deviation. Despite the enhancement from the re-convolutional loss function, the CNN are difficult to acquire new information from the limited training data set. In addition, the regions with high gradient presents a greater deviation compared with the smooth regions. So, in the future research of reducing prediction error, these singular regions or cases should be considered emphatically when generate and utilize a larger training data set.

## 6. Conclusions

In this paper, we used CNN to study the mixing characteristics of the film cooling in a rocket combustor. By modifying the U-net CNN architecture and forming a re-convolutional loss function, the trained model is able to predict the solution of the flow field in a straightforward way.

Facing the small upper stage engines, a subscale film cooled $CH_4/O_2$ combustion chamber is chosen as a reference experiment. The simplified two-dimensional simulation campaign is conducted to generate the training data set. With the varieties of inlet flow conditions, the simulation results are obtained with standard k− εRANS method and then interpolated into the rectangular matrices which are suitable for neural networks.

After the pre-processing work, the training data set is introduced into the modified CNN which has a 17 layers' U-Net architecture. One new CNN-RC method to form the loss function is put forward to update the model. The influence of training data size and integrative steps on the accuracy of solutions, with both normal and new methods, is illustrated, and it shows the new loss function has a better performance when the data size is small. The trained CNN model is implied into the baseline and test film cooling cases to predict the solution of film cooling flow field and the results can reach the accuracy with global errors less than $5.5 \times 10^{-3}$.

The work presented in this paper proved the capability of CNN to solve the multiple flows mixing problem by the means of supervised learning with small sample set. While the work presented is based on the RANS solutions, the modified neural network architecture and the re-convolutional loss function forming method are generic, and can be applied to a wide range of partial differential equation problems in rectangular cartesian coordinate system.

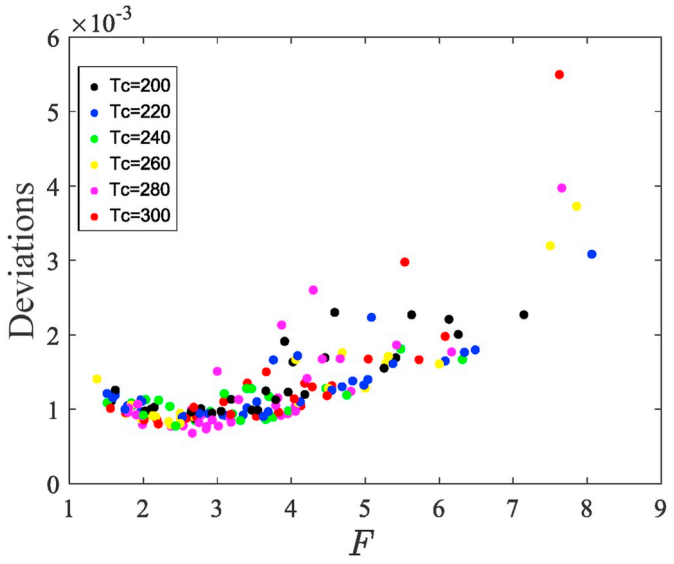



**Fig. 8.** Deviations of test cases with blowing ratio.

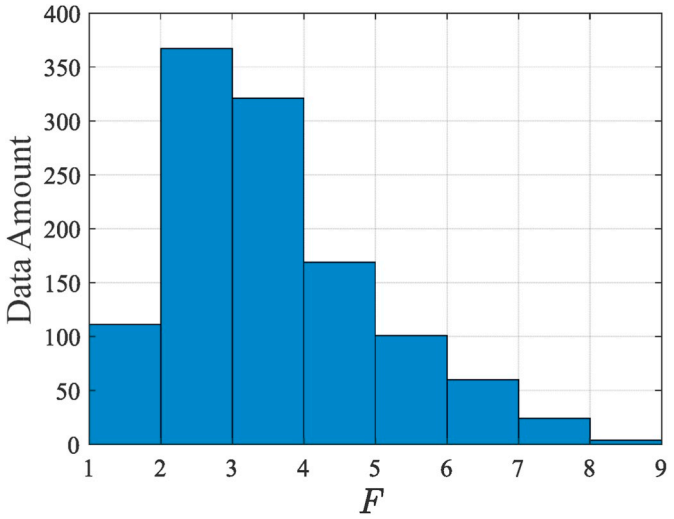



**Fig. 9.** Training data distribution.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgements

Hao Ma is supported by China Scholarship Council, China (No. 201703170250). The authors wish to thank Andrej Sternin for his help in the simulation model simplification stage of this research.

## References

[1] C. Obrien, R. Ewen, Advanced Oxygen-Hydrocarbon Rocket Engine Study, NASA, 1981.
[2] H. Burkhardt, M. Sippel, A. Herbertz, J. Klevanski, Comparative study of kerosene and methane propellant engines for reusable liquid booster stages, 4th International Conference on Launcher Technology" Space Launcher Liquid Propulsion", Belgium, 2002.
[3] M.D. Klem, LOX/Methane In-Space Propulsion Systems Technology Status and Gaps, (2017).
[4] J.S. Kim, H. Jung, J.H. Kim, State of the art in the development of methane/oxygen liquid-bipropellant rocket engine, Journal of the Korean Society of Propulsion Engineers 17 (2013) 120–130.
[5] T. Neill, D. Judd, E. Veith, D. Rousar, Practical uses of liquid methane in rocket engine applications, Acta Astronaut. 65 (2009) 696–705.
[6] J.P. O'Connor, A. Haji-Sheikh, Numerical study of film cooling in supersonic flow, AIAA J. 30 (1992) 2426–2433.
[7] R.J. Goldstein, Film cooling, Advances in Heat Transfer, Elsevier, 1971, pp. 321–379.
[8] B. Richards, J. Stollery, Laminar film cooling experiments in hypersonic flow, J. Aircraft 16 (1979) 177–181.
[9] R. Bass, L. Hardin, R. Rodgers, R. Ernst, Supersonic film cooling, 2nd International Aerospace Planes Conference, 1990, p. 5239.
[10] C. Wang, J. Zhang, J. Zhou, Optimization of a fan-shaped hole to improve film cooling performance by RBF neural network and genetic algorithm, Aero. Sci. Technol. 58 (2016) 18–25.
[11] M. Celano, S. Silvestri, C. Kirchberger, G. Schlieben, D. Suslov, O. Haidn, Gaseous film cooling investigation and model assessment in a sub-scale single element GOX/GCH4 combustion chamber, Proceedings of the 30th International Symposium on Space Technology and Science, 2015.
[12] C.U. Kirchberger, Investigation on Heat Transfer in Small Hydrocarbon Rocket Combustion Chambers, Technische Universität München, 2014.
[13] V. Venkatesh, J. Sriraam, K. Subhash, R.K. Velamati, A. Srikrishnan, B. Ramakrishnananda, S. Batchu, Studies on effusion cooling: impact of geometric parameters on cooling effectiveness and coolant consumption, Aero. Sci. Technol. 77 (2018) 58–66.
[14] J.D. Anderson, J. Wendt, Computational Fluid Dynamics, Springer, 1995.
[15] S.L. Brunton, B.R. Noack, P. Koumoutsakos, Machine learning for fluid mechanics, Annu. Rev. Fluid Mech. 52 (2019) 2020.
[16] A.P. Singh, S. Medida, K. Duraisamy, Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils, AIAA J. (2017) 2215–2227.
[17] Z. Bai, S.L. Brunton, B.W. Brunton, J.N. Kutz, E. Kaiser, A. Spohn, B.R. Noack, Data-driven methods in fluid dynamics: sparse classification from experimental data, Whither Turbulence and Big Data in the 21st Century? Springer, 2017, pp. 323–342.
[18] V. Sekar, Q. Jiang, C. Shu, B.C. Khoo, Fast flow field prediction over airfoils using deep learning approach, Phys. Fluids (2019) 31.
[19] J.N. Kutz, Deep learning in fluid dynamics, J. Fluid Mech. 814 (2017) 1–4.
[20] Z.J. Zhang, K. Duraisamy, Machine learning methods for data-driven turbulence modeling, 22nd AIAA Computational Fluid Dynamics Conference, 2015, p. 2460.
[21] M. Raissi, G.E. Karniadakis, Hidden physics models: machine learning of nonlinear partial differential equations, J. Comput. Phys. 357 (2018) 125–141.
[22] A. Yazdani, M. Raissi, G. Karniadakis, Hidden fluid mechanics: Navier-Stokes informed deep learning from the passive scalar transport, Bull. Am. Phys. Soc. 63 (2018).
[23] M. Raissi, Z. Wang, M.S. Triantafyllou, G.E. Karniadakis, Deep learning of vortex-induced vibrations, J. Fluid Mech. 861 (2019) 119–137.
[24] J. Ling, A. Kurzawski, J. Templeton, Reynolds averaged turbulence modelling using deep neural networks with embedded invariance, J. Fluid Mech. 807 (2016) 155–166.
[25] P.M. Milani, J. Ling, G. Saez-Mischlich, J. Bodart, J.K. Eaton, A machine learning approach for determining the turbulent diffusivity in film cooling flows, J. Turbomach. 140 (2018) 021006.
[26] P.M. Milani, J. Ling, J.K. Eaton, Physical interpretation of machine learning models applied to film cooling flows, J. Turbomach. 141 (2019) 011004.
[27] A.B. Farimani, J. Gomes, V.S. Pande, Deep Learning the Physics of Transport Phenomena, (2017) arXiv preprint arXiv:1709.02432.
[28] N. Thuerey, K. Weissenow, H. Mehrotra, N. Mainali, L. Prantl, X. Hu, Deep Learning Methods for Reynolds-Averaged Navier-Stokes Simulations of Airfoil Flows, (2018) arXiv preprint arXiv:1810.08217.
[29] M.P. Celano, S. Silvestri, C. Bauer, N. Perakis, G. Schlieben, O.J. Haidn, Comparison of single and multi-injector GOC/CH4 combustion chambers, 52nd AIAA/SAE/ASEE Joint Propulsion Conference, 2016, p. 4990.
[30] A. Sternin, N. Perakis, M.P. Celano, M. Tajmar, O.J. Haidn, CFD-analysis of the effect of a cooling film on flow and heat transfer characteristics in a GCH4/GOX rocket combustion chamber, Space Propulsion Conference 2018, 2018.
[31] S. Gordon, B.J. McBride, Computer Program for Calculation of Complex Chemical Equilibrium Compositions and Applications. Part 1: Analysis, (1994).
[32] N. Perakis, M.P. Celano, O.J. Haidn, Heat flux and temperature evaluation in a rectangular multi-element GOX/GCH4 combustion chamber using an inverse heat conduction method, 7th European Conference for Aerospace Sciences, 2017.
[33] N. Perakis, O.J. Haidn, Inverse heat transfer method applied to capacitively cooled rocket thrust chambers, Int. J. Heat Mass Tran. 131 (2019) 150–166.
[34] H. Ziebland, R.C. Parkinson, Heat transfer in rocket engines, Advisory Group for Aerospace Research and Development Paris (France), 1971.
[35] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, PyTorch: an imperative style, high-performance deep learning library, Advances in Neural Information Processing Systems, 2019, pp. 8024–8035.
[36] B. Xu, N. Wang, T. Chen, M. Li, Empirical Evaluation of Rectified Activations in Convolutional Network, (2015) arXiv preprint arXiv:1505.00853.
[37] D.P. Kingma, J. Ba, Adam, A Method for Stochastic Optimization, (2014) arXiv preprint arXiv:1412.6980.
[38] M. Mathieu, C. Couprie, Y. LeCun, Deep Multi-Scale Video Prediction beyond Mean Square Error, (2015) arXiv preprint arXiv:1511.05440.
[39] N.B. Erichson, L. Mathelin, Z. Yao, S.L. Brunton, M.W. Mahoney, J.N. Kutz, Shallow Learning for Fluid Flow Reconstruction with Limited Sensors and Limited Data, (2019) arXiv preprint arXiv:1902.07358.
[40] G.-J. Both, S. Choudhury, P. Sens, R. Kusters, DeepMoD, Deep Learning for Model Discovery in Noisy Data, (2019) arXiv preprint arXiv:1904.09406.

## A.2 Paper II

H. Ma, X. Y. Hu, Y. X. Zhang, N. Thuerey and O. J. Haidn

### A combined data-driven and physics-driven method for steady heat conduction prediction using deep convolutional neural networks

*Contribution*: My contribution to this work was the proposal of the method and the corresponding computer code for its implementation. I have performed numerical experiments, analyzed the results, and wrote the manuscript for publication.

# A Combined Data-driven and Physics-driven Method for Steady Heat Conduction Prediction using Deep Convolutional Neural Networks

Hao Ma[a], Xiangyu Hu[b,*], Yuxuan Zhang[c], Nils Thuerey[d], Oskar J. Haidn[a]

[a]*Department of Aerospace and Geodesy, Technical University of Munich, 85748 Garching, Germany*
[b]*Department of Mechanical Engineering, Technical University of Munich, 85748 Garching, Germany*
[c]*Beijing Aerospace Propulsion Institute, 100076, Beijing, China*
[d]*Technical University of Munich, 85748, Garching, Germany*

**Abstract**

With several advantages and as an alternative to predict physics field, machine learning methods can be classified into two distinct types: data-driven relying on training data and physics-driven using physics law. Choosing heat conduction problem as an example, we compared the data- and physics-driven learning process with deep Convolutional Neural Networks (CNN). It shows that the convergences of the error to ground truth solution and the residual of heat conduction equation exhibit remarkable differences. Based on this observation, we propose a combined-driven method for learning acceleration and more accurate solutions. With a weighted loss function, reference data and physical equation are able to simultaneously drive the learning. Several numerical experiments are conducted to investigate the effectiveness of the combined method. For the data-driven based method, the introduction of physical equation not only is able to speed up the convergence, but also produces physically more consistent solutions. For the physics-driven based method, it is observed that the combined method is able to speed up the convergence up to 49.0% by using a not very

*Corresponding author. Tel.: +49 89 289 16152.
*Email addresses:* hao.ma@tum.de (Hao Ma), xiangyu.hu@tum.de (Xiangyu Hu ), uruz7@live.com (Yuxuan Zhang), nils.thuerey@tum.de (Nils Thuerey), oskar.haidn@tum.de (Oskar J. Haidn)

restrictive coarse reference.
*Keywords:* Machine learning, Physics field prediction, Data driven, Physics driven, Convolutional neural networks, Convergence speed, Combined method, Weighted loss function

---

## 1. Introduction

With abundant training methods and high-performance computing resources, machine learning has been applied for many scientific research fields, including computational physics for modeling[1, 2], optimization[3, 4], control[5] and other critical tasks[6, 7]. A specific application is to predict physics field for reducing or avoiding the large computational cost of the traditional numerical (finite volume/element/difference) methods, such as *Computational Fluid Dynamics* (CFD) which solve *Partial Differential Equations*(PDEs)[8].

In machine learning approaches, data-driven methods were widely used to train the model with a large amount of labeled training data. For physics field prediction, the data-driven methods can be roughly identified as indirect to train closure models[9, 10] and direct to obtain the solutions[11, 12]. While the indirect method has achieved great successes in recent[13, 14], the direct data-driven method has also exhibited the capability to capture physics characteristics and to provide accurate estimates without resorting to expensive numerical computations[15, 16, 17, 18]. However, since the training data sets are still generated from traditional numerical solutions[19, 20, 21], it leads to an embarrassing loop that it does not truly solve the issue of expansive numerical computation. In addition, in some difficult cases, though a large number of training samples are used, the data-driven method may still not be able to obtain sufficiently accurate solutions. Choosing the work in Ref.[19] as an example, independent of the number of training samples, considerable errors always manifest themselves in the inferred flow field just behind the airfoil. Similar phenomenon also happens in Ref.[12]. It requires another novel approach to eliminate this shortcoming other than simply utilizing an even larger training data set.

In fact, the physics law which is unknown in data-driven methods could be explicitly employed in the learning process[22, 23]. Raissi et al. introduced this idea into machine learning algorithms and named it as *Physics Informed Neural Networks* (PINN)[24]. By introducing PDEs into the loss function, PINN is

able to predict the solution that satisfies physics law. The inferred solution is trained to obey the corresponding PDE and boundary conditions. The effectiveness of this physics-driven method has been demonstrated through a collection of physics problems[25, 26, 27, 29]. Compared with data-driven methods, it extricate machine learning from the dependence of training data, remarkably decrease the cost of data set generation[28, 30, 31]. However, as shown in a single case training later, the cost of physics-driven method is typically more expensive than that of the data-driven method.

In order to remedy the above mentioned shortcomings, we propose an idea that combining data- and physics-drivens together. To our knowledge, this is the first attempt that simultaneously utilizes training data and physics law to drive machine learning for physics field prediction. Choosing the steady solution of heat conduction as an example, we first consider the original data-driven and physics-driven methods based on a deep CNN respectively and compare their learning progresses for a single case training. Then, based on the comparisons, we propose a weighted loss function combining the effects of reference target and physics law (given as Laplace equation) for training the CNN to predict temperature fields. After this, several numerical experiments are conducted and analyzed to study the improvements achieved by the combined method.

## 2. Preliminaries

We choose the steady solution of heat conduction whose physics law can be expressed with a second-order PDE, i.e. Laplace equation as an example. Focus on this problem, we first describe the CNN architecture used and the original data- and physics-driven methods, especially their distinct loss functions.

### *2.1. U-net Architecture for CNN*

The CNN used here is based on a U-net architecture, which is firstly proposed for machine vision[32]. Including the input and output layer, the U-net architecture consists of 17 layers and corresponding convolutional blocks. Each

convolutional block has a similar structure: batch normalization, active function, convolutional calculation, and dropout[33].

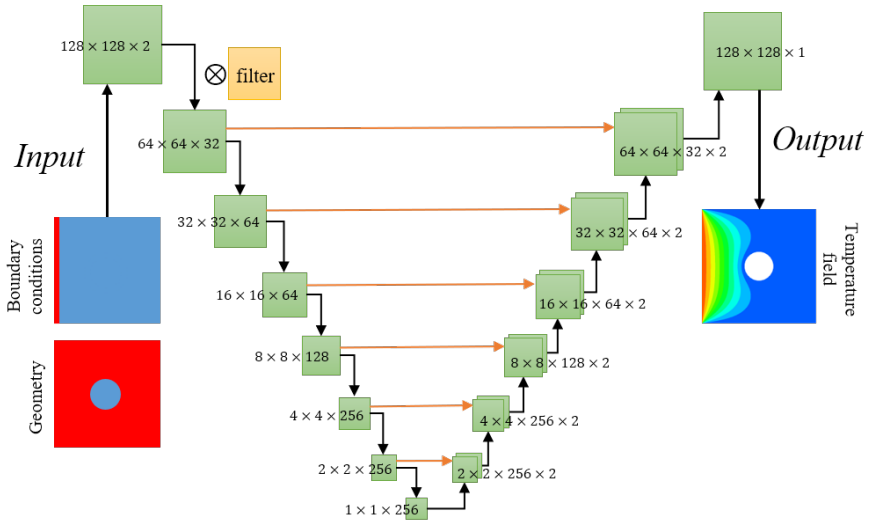



Figure 1: Schematic of U-net architecture. In the input layer, the color red and blue represent value 1 and 0 respectively. Each green box corresponds to a multi-channel feature matrix. Black corner arrows denote the down-sampling or up-sampling operation using convolutional calculation. Orange arrows denote the concatenation of the feature channels between encoding and decoding.

As shown in Figure 1, geometry and boundary conditions are input into the architecture as square matrices with a size of $128 \times 128$. Then the corresponding square filters are utilized to conduct the convolutional calculation layer by layer until the matrices with only one data point are obtained. In this encoding process, the values of input matrices are progressively down-sampled by convolutional calculations. With the amount of feature channels increasing, large-scale information is extracted. Then the decoding process which can be regarded as a series of inverse convolutional operations mirrors the behavior of encoding. The solutions are reconstructed in the up-sampling layers along with the increase of spatial resolution and the decrease of feature channel amounts. Eventually the output only has one channel giving the temperature field. It is noteworthy that there are concatenation operations between corresponding en-

coding and decoding blocks as shown in Figure 1. These connections effectively double the amount of feature channels in every decoding block and enable the neural networks to consider the information from the encoding layers.

The CNNs are trained using stochastic gradient descent optimization, which requires a loss function to calculate the model error. Except for different loss functions, the U-nets and other training settings are kept same in the following description on the original data- and physics-driven methods. By means of backpropagation, the weights and other parameters of the entire networks are adjusted by Adam optimizer[34] and eventually the loss is minimized and the CNN are able to reconstruct the solution of heat conduction problems. More details of the U-net architecture and CNN can be found in Ref. [19].

### *2.2. Original Data- and Physics-driven Methods*

In the data-driven method, the loss function compares the difference between training target and output result as

$$\mathcal{L}_{\text{data}} = \left| T_{\text{out}} - T_{\text{tar}} \right|. \tag{1}$$

The subscript "data" here denotes data-driven. $T_{\text{out}}$ and $T_{\text{tar}}$ are output and target temperature distributions respectively.

Based on Fouriers law, when thermal conductivity is considered as constant and there is no inner heat source, the physics law of heat conduction can be described as two-dimensional Laplace equation

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0, \tag{2}$$

which is a typical PDE whose solution is important in many branches of physics.

In the physics-driven method, this physics law is used to drive the learning process by the loss function

$$\mathcal{L}_{\text{phy}} = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = e_1. \tag{3}$$

The boundaries are constrained with Dirichlet boundary conditions by which the temperatures of the outer and inner boundaries are kept as a constant.

The boundary conditions are implemented differently for the outer and inner boundaries. While the temperatures at the outer boundaries are assigned as constants, their values at the inner boundary as well as the inside void region are constrained by a loss function as

$$\mathcal{L}_{\mathrm{BC,in}} = T - T_{\mathrm{BC,in}} = e_2. \tag{4}$$

A schematic of the physics-driven method is shown in Figure 2. Note that, for

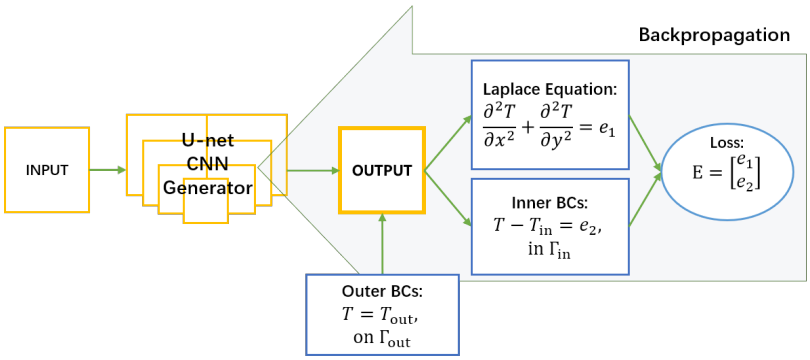



Figure 2: Schematic of the physics-driven method. U-net CNN generates the solution. The backpropagation computes the gradient of the loss function and update the weights of the multilayer CNN to satisfy the Laplace equation and boundary conditions.

the physics-driven method, the second inputting channel of the U-net CNN not only describes the geometry but also functions as a mask, by which the Laplace equation is not effective in the void region.

## 3. Observations on Learning Processes and Combined Method

In the following, we describe the observations on the learning processes of the original data- and physics-driven methods, which motivates the combined method, for a single case training.

### *3.1. Single Case Training*

In general, machine learning methods are able to train multiple cases simultaneously. Here, the single case which is defined by a specific geometry and

boundary condition combination is considered and the CNN are trained to output the corresponding field solution. By this way, 4 training tasks as shown in Figure 3 are carried out. The learning algorithms are based on PyTorch framework[33] and the trainings are conducted on a single NVIDIA GeForce RTX 2080 Ti Card. The training samples for data-driven method are obtained by *Finite Volume Method* (FVM)[35] and every numerical solution is interpolated into a $128 \times 128$ grid to suit the learning domain with a same resolution. Taking Task 1 as an example, it takes data-driven learning about 3k iterative steps (110 s) and physics-driven learning about 23k iterative steps (750 s) , respectively, to reach the errors less than 0.2%.

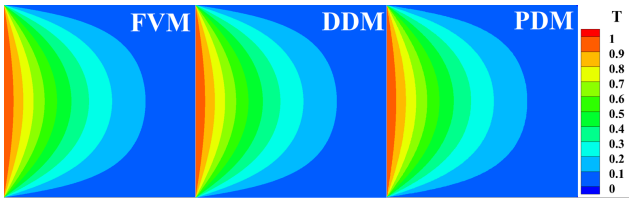
### *3.2. Comparison of Learning Processes*

As shown in Figure 4, in the data-driven learning process of Task 1, after a few iterative steps, a brief form of the global temperature profile (global structure) starts to appear, all local values approach those of target solution. The temperature contours are rough at first and smoothed successively with the global structure itself unchanged. However, in the physics-driven learning process, the contours become smooth after a few iterative steps. Then a large error spot (denoted as a valley) appears, which also happens in the other training tasks. The global structure is very different from that of the true solution. After the gradual disappearance of the valley, the still existing residuals near the boundaries make the subsequent learning process similar to a typical numerical solution process of an unsteady heat conduction problem with a Dirichlet boundary condition, as shown in the last two sub-figures of Figure 4b.

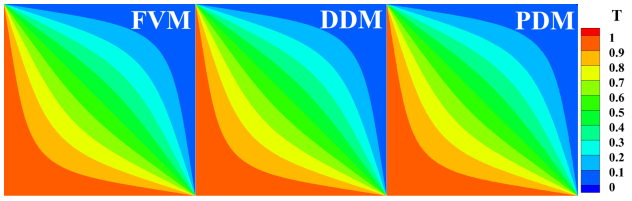
In order to study the convergence process, we defined the overall error $E$ and the Laplace residual $LR$. The former is defined as

$$E = \frac{1}{n \times n} \sum_{i=1}^{n \times n} \left| T_{\text{out}} - T_{\text{tar}} \right|, \tag{5}$$

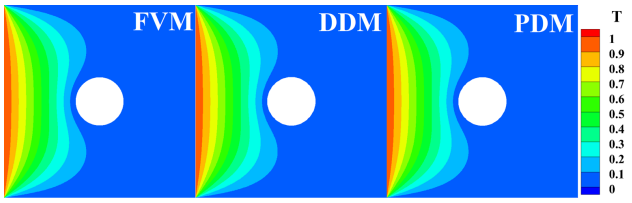
where $n \times n$ is the resolution of learning domain. It is an estimation the degree that the outputs satisfy the solution. While the Laplace residual $LR$ is defined



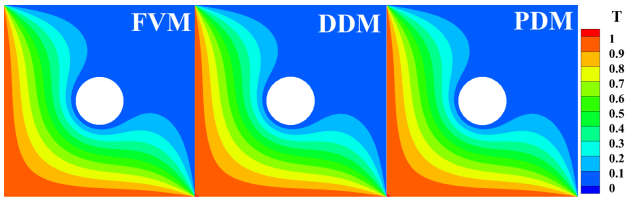

(a) Task 1




(b) Task 2

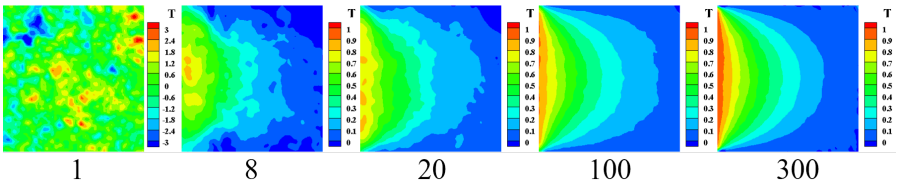



(c) Task 3

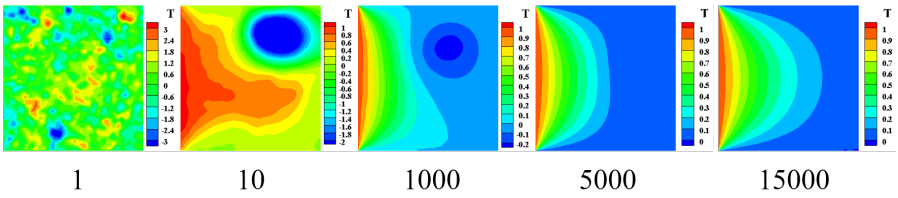



(d) Task 4

Figure 3: Single case training with different geometries and boundary conditions. The geometry of first two tasks is a simple square plate. Task 1: the temperature of left boundary is 1, the other three are 0. Task 2: the temperatures of left and bottom boundaries are 1, the other two are 0. The geometry of last two tasks is a square plate with a central hole. The boundary conditions of Tasks 3 and 4 are the same with Tasks 1 and 2 respectively, except that the temperature of inner hole is 0. Left to right: the results obtained by finite volume (FVM), data-driven (DDM) and physics-driven (PDM) methods respectively. The learned results are almost identical with numerical simulation references.




(a) Data-driven learning




(b) Physics-driven learning

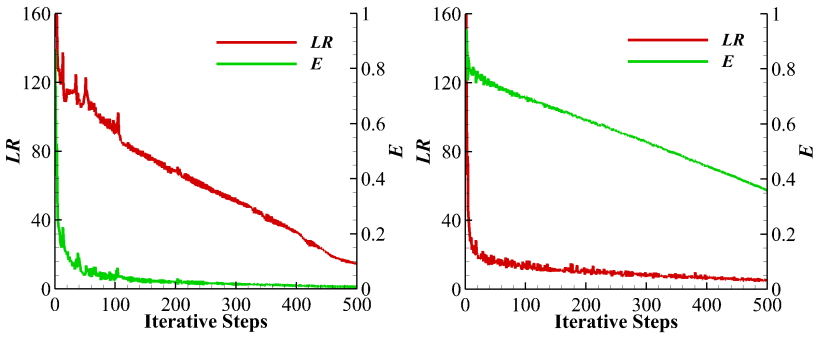
Figure 4: Comparison of learning process of Task 1. The numbers below the contours are iterative steps. For data-driven learning, contours transform from rough to smooth while the global structure keeps unchanged. For physics-driven learning, the contours become smooth at the early training stage and then remedy "valley" gradually.

as

$$LR = \frac{1}{n \times n} \sum_{i=1}^{n \times n} \left| \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right|. \tag{6}$$

It represents the degree that the outputs satisfy Laplace equation, i.e., the local structure of solution. For data-driven learning, the loss function only considers the error between output and target rather than the residual of Laplace equation, so we call $E$ loss term or explicit error and $LR$ non-loss term or implicit error. Similarly in physics-driven learning, $LR$ is explicit error and $E$ implicit error.

As shown in Figure 5, for both data-driven and physics-driven learning, $E$ and $LR$ drop dramatically in the beginning. However, after approximate 10 iterative steps, the convergence behaviors of the two terms, as explicit or implicit error, exhibits significant differences. When the explicit errors have gradually approached an adequately small value, the implicit errors are still large. Finally, after much large number of iteration steps, both errors decrease to sufficiently small values, i.e. both the solution and its local structure are obtained.




(a) Data-driven method (b) Physics-driven method

Figure 5: Training history of $LR$ and $E$ (steps $= 0 \sim 500$). The convergence behaviors of two learning methods have significant differences.

For data-driven learning, as shown in Figure 4a the temperature contour keeps rough for a long period. And there are even small isolated islands in the neighboring temperature levels. The reason why this phenomenon happens is that the error of each data point approach to zero locally and separately. There is no corresponding explicit relation between these adjacent data points to restrict the local structures, which is given as Laplace equation in this case. For physics-driven learning, the relation of adjacent data points or local structure is constrained by Laplace equation explicitly and the values varies smoothly. However, due to the lack of explicit restriction to target solution or the global structure, the implicit error $E$ decreases slowly and the convergence of the solution is much slower than that of the data-driven learning.

### *3.3. Combined Method*

Based on the above observations, we propose to improve physics consistency and increase the convergence speed by combining both $E$ and $LR$ being into the loss terms.

It is observed that the scale of $LR$ is significantly larger than $E$ as shown in Figure 5. Utilizing a simple summation of these two errors as the total loss

leads a skewed optimization[37] with a dominance of the Laplace residual. In order to remedy this issue, we employ a weighted loss function which has been widely used in object detection[38] and audio detection[39]. The weighted loss function considering both target data and Laplace equation is written as

$$\mathcal{L} = \mathcal{L}_{\text{data}} + \mathrm{R} * \mathcal{L}_{\text{phy}}, \tag{7}$$

where R is a constant hyperparameter which is tuned to adapt the scales. With this weighted loss function, the different loss terms can be easily scaled to an equivalent magnitude. The combined method actually has two types: data-driven based and physics-driven based. For the data-driven based method, the loss function is Equation (7) and employed during the whole learning process. For the physics-driven based method, the loss function is modified as

$$\mathcal{L} = \begin{cases} \mathcal{L}_{\text{data,ref}} + \mathrm{R} * \mathcal{L}_{\text{phy}} & \mathcal{L} \geq \mathcal{L}_{\text{thr}} \\ \mathcal{L}_{\text{phy}} & \mathcal{L} < \mathcal{L}_{\text{thr}} \end{cases}, \tag{8}$$

where $\mathcal{L}_{\text{data,ref}}$ is the error term with some reference targets depending on different practical considerations. $\mathcal{L}_{\text{thr}}$ is the threshold value of the loss indicating that once the loss is less than the threshold, the loss function will only consist of the Laplace term.

## 4. Numerical Experiments

### *4.1. Data-driven Based Training*

The data-driven based training uses two distinct data sets: one has only a single sample and the another consists of multiple samples.

*Single Case Training*

As shown in Figure 6, unlike the original method, $LR$ and $E$ of combined method exhibit a similar convergence behavior. After the dramatic drop in the beginning period, they change to the relatively slow decrease and then the steady decline together. $LR$ has obtained a considerable acceleration of convergence. To check the overall performance, instead of a single run, 5 independent runs

are conducted with the original and combined methods. It is observed that the averaged iterative steps when $E$ reaches the criteria of convergence (0.005) are 684 and 267 respectively. This result gives that the combined method remarkably accelerates the learning with a rate of 60.9%. In addition, the combined

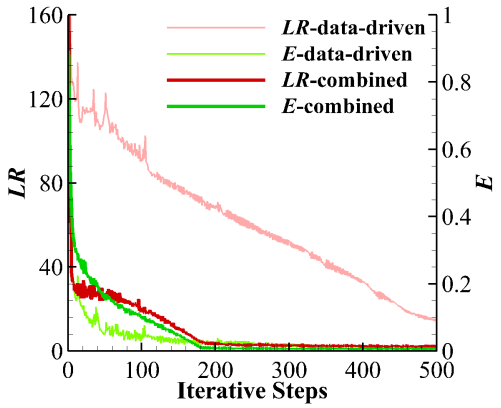



Figure 6: Enhancements for data-driven method in single case training: $LR$ and $E$. The convergence speed of $LR$ has a notable improvement.

method also leads to a notable improvement on obtaining the physics-consistent solution, even though the overall errors have a same level (Figure 7). This is due to that the Laplace residuals are much smaller, i.e. the local structure of solution has a better physics consistency.

*Multiple Cases Training*

For multiple cases training, the data set is obtained with the variation of boundary temperature ($T_{boundary}$ = 0/0.5/1), hole shape (square/round) and position (9 different positions). There are 4,374 samples split randomly into training/test sets with an 80/20 ratio.

As shown in Figure 8, the decrease trend of $E$ are almost same for the original and combined methods. When the training reaches 1000 epochs, both of them are sufficiently small. However, $LR$ of them exhibit different convergence behaviors. The $LR$ of combined method, as loss term, decreases much faster. For the temperature profile, the enhancement obtained by combined method is

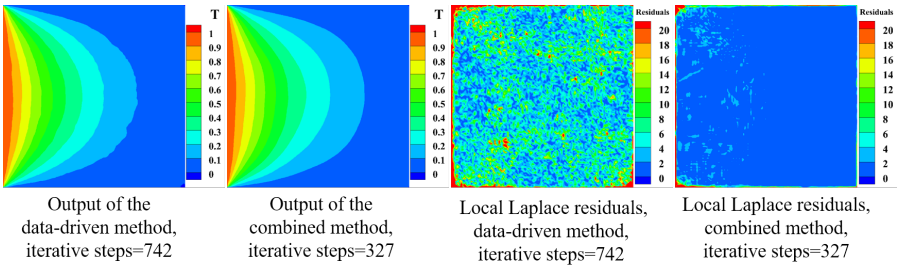
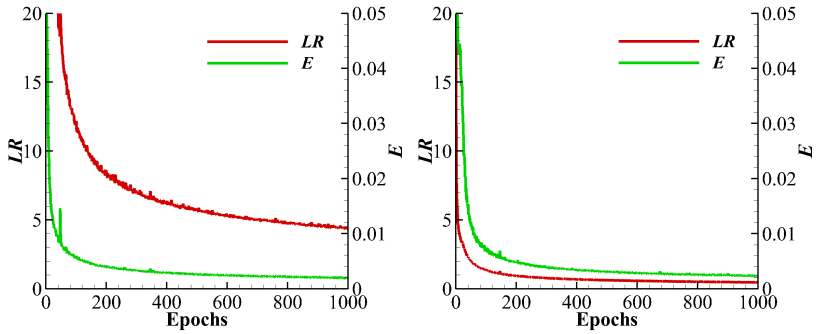



Figure 7: Enhancements for data-driven method in single case training: CNN outputs and local Laplace residuals. $E$ of the two methods are both 0.05. Compared with the data-driven method, the contour of combined method is smoother and the local Laplace residuals in data points are much smaller.




(a) Data-driven method

(b) Combined method

Figure 8: Enhancements for data-driven method in multiple cases training): $LR$ and $E$. The histories of $E$ of the two methods are almost same, while the $LR$ of combined method decreases much faster.

similar to that in the single case training. From the zoomed-views (as shown in Figure 9), the temperature contours obtained from combined method are much smoother, which suggests the solution is more consistent with physics law.

### *4.2. Physics-driven Based Training*

In this section, single case is considered for physics-driven based training. According to Equation (8), a reference target is required beforehand. Here, as shown in Figure 10, 5 reference targets are chosen. Among these targets,

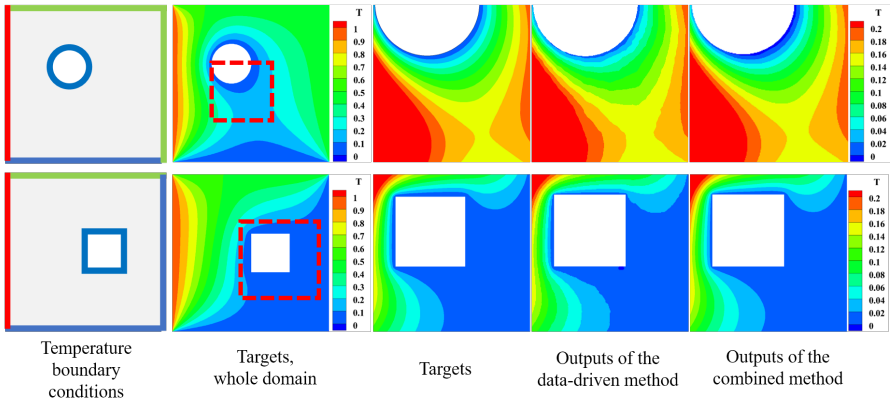
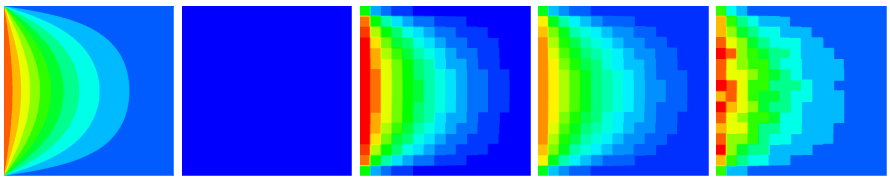



Figure 9: Enhancements for data-driven method in multiple cases training: outputs of CNN. There are two typical cases from test set with different geometries. The colors in first column represent $T_{boundary} = 0/0.5/1$. The last three columns show part of the output which marked with the red dot line in the second column. In contrast to the original data-driven method, the temperature contours obtained by combined method are smoother.

the true and zero profiles represent the true or false limits, while the 3 coarse temperature profiles mimic the practical application where the accurate solution is not available.

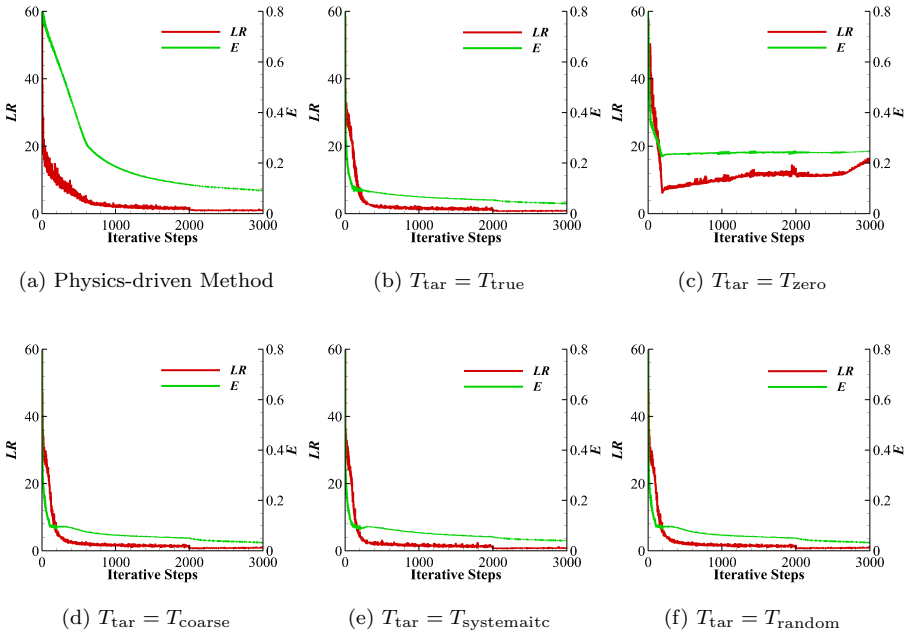


(a) $T_{\text{true}}$ (b) $T_{\text{zero}}$ (c) $T_{\text{coarse}}$ (d) $T_{\text{systematic}}$ (e) $T_{\text{random}}$

Figure 10: Reference targets. $T_{\text{true}}$ and $T_{\text{zero}}$ are true and zero temperature profile respectively. $T_{\text{coarse}}$ is down-sampled from the true temperature profile, which could be a result of numerical simulation with a coarse mesh. $T_{\text{systematic}}$ is the coarse profile with a different boundary temperature and represents the experimental results with systematic errors. It may also estimate whether a reference target can be applied for training other cases. $T_{\text{random}}$ is the coarse temperature profile with random errors which represents an experimental result with measurement uncertainties.

Due to the introduction of $E$ as one loss-term, the convergence speeds with

the true and coarse targets are improved a lot. As shown in Figure 11, compared with the steady decline in the physics-driven learning, $E$ of combined method drops more dramatically in the beginning. We set the threshold $\mathcal{L}_{\text{thr}}$ in Equation 8 as 0.1. After $\mathcal{L}_{\text{thr}}$ reached, only $LR$ is the loss term. So $E$ changes to steady descent immediately while $LR$ still goes on rapid decline. For $T_{\text{tar}} = T_{\text{zero}}$, $LR$ and $E$ have the similar trend at the very beginning. However, the overall error $E$ never reaches the threshold as the optimizer cannot find a way to reduce the gradient because of the incorrect reference target.

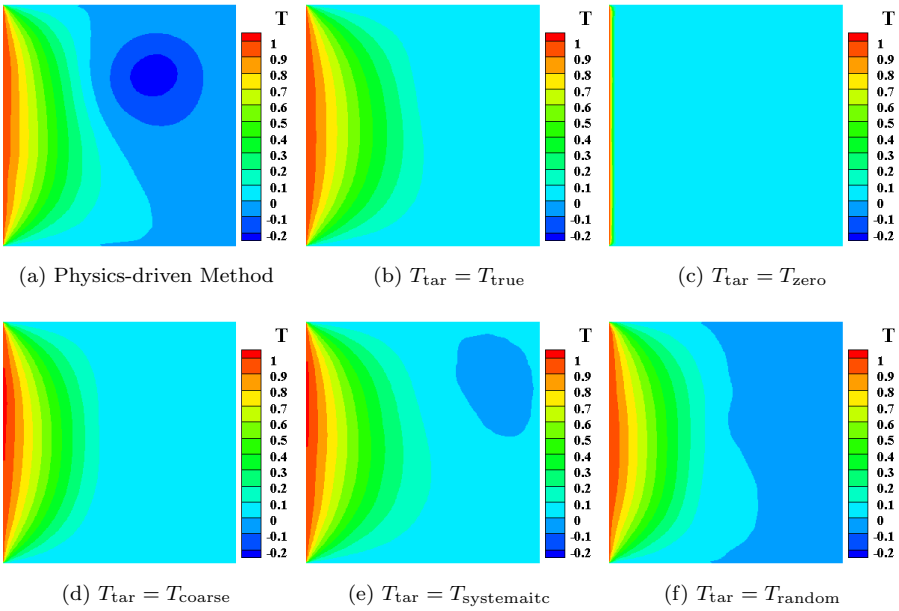



(a) Physics-driven Method (b) $T_{\text{tar}} = T_{\text{true}}$ (c) $T_{\text{tar}} = T_{\text{zero}}$

(d) $T_{\text{tar}} = T_{\text{coarse}}$ (e) $T_{\text{tar}} = T_{\text{systemaitc}}$ (f) $T_{\text{tar}} = T_{\text{random}}$

Figure 11: Enhancements for physics-driven method: $LR$ and $E$ (threshold = 0.1). Except for $T_{\text{zero}}$, all the reference targets are able to accelerate the learning notably.

Compared with the original physics-driven method, it is observed that the combined method with all the reference targets, except for zero profiles, are able to obtain a more accurate result by eliminating the large error spot quickly (as shown in Figure 12). These results suggest that the requirement for appropriate reference targets are not very restrictive in practical applications.




(a) Physics-driven Method (b) $T_{\text{tar}} = T_{\text{true}}$ (c) $T_{\text{tar}} = T_{\text{zero}}$

(d) $T_{\text{tar}} = T_{\text{coarse}}$ (e) $T_{\text{tar}} = T_{\text{systemaitc}}$ (f) $T_{\text{tar}} = T_{\text{random}}$

Figure 12: Enhancements for physics-driven method: outputs of CNN (threshold = 0.1, iterative step = 1000). $T_{\text{zero}}$ is not able to obtain the right solution. The other four reference targets are able to remedy the "valley" faster.

To study the influence of the threshold $\mathcal{L}_{\text{thr}}$, the mean costs of 5 independent trainings with different reference targets are summarized in Table 1. It is observed that the combined method improves physics-driven learning considerably. Compared with the true reference target, the other three coarse targets have a only slightly slower convergence speed. With decreasing $\mathcal{L}_{\text{thr}}$ =0.15, 0.1 and 0.05, the acceleration obtained by using the true coarse target over the original physics-driven method are 22.7%, 34.4% and 49.0% respectively. Similar behaviors have been obtained for the other two coarse references. It is concluded that the smaller thresholds result in the bigger improvements. While in a real application, a too small threshold may result in the non-convergence as suggested by the zero profile reference. So one may choose a moderate threshold for safety.

Table 1: Costs of different methods. The cost is represented as the number of iteration steps when $E$ reaches $\mathcal{L}_{\text{thr}}$ and the convergence absolute criteria $\mathcal{C}$ (0.01).

| Methods | $\mathcal{L}_{\text{thr}} = 0.15$ | | $\mathcal{L}_{\text{thr}} = 0.1$ | | $\mathcal{L}_{\text{thr}} = 0.05$ | |
|---|---|---|---|---|---|---|
| | $\mathcal{L}_{\text{thr}}$ | $\mathcal{C}$ | $\mathcal{L}_{\text{thr}}$ | $\mathcal{C}$ | $\mathcal{L}_{\text{thr}}$ | $\mathcal{C}$ |
| Physics-driven | 1322 | 13494 | 2542 | 13494 | 6334 | 13494 |
| $T_{\text{true}}$ | 51 | 9875 | 112 | 8904 | 138 | 6758 |
| $T_{\text{coarse}}$ | 57 | 10428 | 109 | 8853 | 145 | 6876 |
| $T_{\text{systematic}}$ | 48 | 10091 | 116 | 9025 | 140 | 6894 |
| $T_{\text{random}}$ | 53 | 9917 | 105 | 8963 | 139 | 7054 |

## 5. Conclusion

In this paper, we proposed a combined data-driven and physics-driven method to directly predict field solution of physics problems using deep CNN. The combined method simultaneously utilizes training data and physics law to drive the learning process. For the data-driven based method, besides accelerated convergence, the obtained local structure of solution achieves a better physics consistency. For the physics-driven based method, learning process can be accelerated considerably even with not very restrictive choices of reference targets, which is useful for practical application when a accurate reference is not available. It is noteworthy that the related CNN architecture and heat conduction problem are generic and the combined method suits for other physics laws which can be expressed as PDEs. Further research will be carried out for predicting complex flow field with the present method.data

## 6. Acknowledgement

Hao Ma (No. 201703170250) and Yuxuan Zhang (No. 201804980021) are supported by China Scholarship Council when they conduct the work this paper represents. The authors thank the colleagues for the beneficial discussion, especially Chi Zhang and Nikolaos Perakis.

## References


[1] W. Yan, S. Lin, O. L. Kafka, Y. Lian, C. Yu, Z. Liu, J. Yan, S. Wolff, H. Wu, E. Ndip-Agbor, et al., Data-driven multi-scale multi-physics models to derive process–structure–property relationships for additive manufacturing, Computational Mechanics 61 (5) (2018) 521–541.

[2] Z. J. Zhang, K. Duraisamy, Machine learning methods for data-driven turbulence modeling, in: 22nd AIAA Computational Fluid Dynamics Conference, 2015, p. 2460.

[3] T. Kirchdoerfer, M. Ortiz, Data-driven computational mechanics, Computer Methods in Applied Mechanics and Engineering 304 (2016) 81–101.

[4] V. Christelis, A. Mantoglou, Physics-based and data-driven surrogate models for pumping optimization of coastal aquifers, European Water 57 (2017) 481–488.

[5] M. Kim, G. Pons-Moll, S. Pujades, S. Bang, J. Kim, M. J. Black, S.-H. Lee, Data-driven physics for human soft tissue animation, ACM Transactions on Graphics (TOG) 36 (4) (2017) 54.

[6] P. J. Schmid, Dynamic mode decomposition of numerical and experimental data, Journal of fluid mechanics 656 (2010) 5–28.

[7] S. L. Brunton, B. R. Noack, P. Koumoutsakos, Machine learning for fluid mechanics, Annual Review of Fluid Mechanics 52 (2019).

[8] J. D. Anderson, J. Wendt, Computational fluid dynamics, Vol. 206, Springer, 1995.

[9] J.-X. Wang, J.-L. Wu, H. Xiao, Physics-informed machine learning approach for reconstructing reynolds stress modeling discrepancies based on dns data, Physical Review Fluids 2 (3) (2017) 034603.

[10] J.-L. Wu, H. Xiao, E. Paterson, Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework, Physical Review Fluids 3 (7) (2018) 074602.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in neural information processing systems, 2014, pp. 2672–2680.

[12] A. B. Farimani, J. Gomes, V. S. Pande, Deep learning the physics of transport phenomena, arXiv preprint arXiv:1709.02432 (2017).

[13] K. Duraisamy, G. Iaccarino, H. Xiao, Turbulence modeling in the age of data, Annual Review of Fluid Mechanics 51 (2019) 357–377.

[14] J. Ling, A. Kurzawski, J. Templeton, Reynolds averaged turbulence modelling using deep neural networks with embedded invariance, Journal of Fluid Mechanics 807 (2016) 155–166.

[15] H. Wu, A. Mardt, L. Pasquali, F. Noe, Deep generative markov state models, in: Advances in Neural Information Processing Systems, 2018, pp. 3975–3984.

[16] S. Bhatnagar, Y. Afshar, S. Pan, K. Duraisamy, S. Kaushik, Prediction of aerodynamic flow fields using convolutional neural networks, Computational Mechanics (2019) 1–21.

[17] S. Jeong, B. Solenthaler, M. Pollefeys, M. Gross, et al., Data-driven fluid simulations using regression forests, ACM Transactions on Graphics (TOG) 34 (6) (2015) 199.

[18] J. Tompson, K. Schlachter, P. Sprechmann, K. Perlin, Accelerating eulerian fluid simulation with convolutional networks, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org, 2017, pp. 3424–3433.

[19] N. Thuerey, K. Weissenow, H. Mehrotra, N. Mainali, L. Prantl, X. Hu, Deep learning methods for reynolds-averaged navier-stokes simulations of airfoil flows, arXiv preprint arXiv:1810.08217 (2018). arXiv:1810.08217v2.

[20] P. M. Milani, J. Ling, G. Saez-Mischlich, J. Bodart, J. K. Eaton, A machine learning approach for determining the turbulent diffusivity in film cooling flows, Journal of Turbomachinery 140 (2) (2018) 021006.

[21] S. Lee, D. You, Data-driven prediction of unsteady flow fields over a circular cylinder using deep learning, arXiv preprint arXiv:1804.06076 (2018).

[22] H. Lee, I. S. Kang, Neural algorithm for solving differential equations, Journal of Computational Physics 91 (1) (1990) 110–131.

[23] I. E. Lagaris, A. Likas, D. I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE transactions on neural networks 9 (5) (1998) 987–1000.

[24] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations, arXiv preprint arXiv:1711.10566 (2017).

[25] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations, arXiv preprint arXiv:1711.10561 (2017).

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

[27] L. Lu, X. Meng, Z. Mao, G. E. Karniadakis, Deepxde: A deep learning library for solving differential equations, arXiv preprint arXiv:1907.04502 (2019).

[28] L. Sun, H. Gao, S. Pan, J.-X. Wang, Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data, arXiv preprint arXiv:1906.02382 (2019).

[29] R. Sharma, A. B. Farimani, J. Gomes, P. Eastman, V. Pande, Weakly-supervised deep learning of heat transport via physics informed loss, arXiv preprint arXiv:1807.11374 (2018).

[30] Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, P. Perdikaris, Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data, Journal of Computational Physics 394 (2019) 56–81.

[31] N. Geneva, N. Zabaras, Modeling the dynamics of pde systems with physics-constrained deep auto-regressive networks, arXiv preprint arXiv:1906.05747 (2019).

[32] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: International Conference on Medical image computing and computer-assisted intervention, Springer, 2015, pp. 234–241.

[33] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., Pytorch: An imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems, 2019, pp. 8024–8035.

[34] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[35] 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.

[36] S. Ruder, An overview of gradient descent optimization algorithms, arXiv preprint arXiv:1609.04747 (2016).

[37] S. Panchapagesan, M. Sun, A. Khare, S. Matsoukas, A. Mandal, B. Hoffmeister, S. Vitaladevuni, Multi-task learning and weighted cross-entropy for dnn-based keyword spotting., in: Interspeech, Vol. 9, 2016, pp. 760–764.

[38] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 779–788.

[39] H. Phan, M. Krawczyk-Becker, T. Gerkmann, A. Mertins, Dnn and cnn with weighted and multi-task loss functions for audio event detection, arXiv preprint arXiv:1708.03211 (2017).

## A.3 Paper III

H. Ma, Y. X. Zhang, N. Thuerey, X. Y. Hu and O. J. Haidn

### Physics-driven learning of the steady Navier-Stokes equations using deep convolutional neural networks

*Contribution*: My contribution to this work was the development of the method and the corresponding computer code for its implementation. I have performed numerical experiments, analyzed the results, and wrote the manuscript for publication.

# Physics-driven Learning of the Steady Navier-Stokes Equations using Deep Convolutional Neural Networks

Hao Ma[a], Yuxuan Zhang[b], Nils Thuerey[c], Xiangyu Hu[d,*], Oskar J. Haidn[a]

[a] *Department of Aerospace and Geodesy, Technical University of Munich, 85748 Garching, Germany*
[b] *Beijing Aerospace Propulsion Institute, 100076, Beijing, China*
[c] *Technical University of Munich, 85748, Garching, Germany*
[d] *Department of Mechanical Engineering, Technical University of Munich, 85748 Garching, Germany*

**Abstract**

Recently, physics-driven deep learning methods have shown particular promise for the prediction of physical fields, especially to reduce the dependency on large amounts of pre-computed training data. In this work, we target the physics-driven learning of complex flow fields with high resolutions. We propose the use of *Convolutional neural networks* (CNN) based U-net architectures to efficiently represent and reconstruct the input and output fields, respectively. By introducing Navier-Stokes equations and boundary conditions into loss functions, the physics-driven CNN is designed to predict corresponding steady flow fields directly. In particular, this prevents many of the difficulties associated with approaches employing fully connected neural networks. Several numerical experiments are conducted to investigate the behavior of the CNN approach, and the results indicate that a first-order accuracy has been achieved. Specifically for the case of a flow around a cylinder, different flow regimes can be learned and the adhered "twin-vortices" are predicted correctly. The numerical results also show that the training for multiple cases is accelerated significantly, especially for the difficult cases at low Reynolds numbers, and when limited

*Corresponding author. Tel.: +49 89 289 16152.
*Email addresses:* hao.ma@tum.de (Hao Ma), uruz7@live.com (Yuxuan Zhang), nils.thuerey@tum.de (Nils Thuerey), xiangyu.hu@tum.de (Xiangyu Hu ), oskar.haidn@tum.de (Oskar J. Haidn)

reference solutions are used as supplementary learning targets.

*Keywords:* Deep learning, Physics-driven method, Convolutional neural networks, Navier–Stokes equations

---

## 1. Introduction

In some practical fluid mechanics problems such as real-time or frequent query analysis, a large number of solutions for different initial/boundary condition combinations are to be considered [1, 2, 3]. For the traditional discrete analysis, numerical simulations have to be conducted repeatedly and the computational cost quickly becomes overly expensive [4]. In contrast to classical computational methods, machine learning approaches, and especially the field of deep learning that employs *Neural Networks* (NN), have demonstrated their capabilities to predict flow fields rapidly and accurately [5, 6, 7].

The previous research on flow field prediction using NN is mainly focused on data-driven methods. Besides the indirect way using closure model [8, 9], the field solution can also be directly obtained from the network trained with a large number of samples [10, 11]. However, for complex flows in practical engineering problems, the training samples very often require extraction, preprocessing and may be hard to obtain [12]. Some data-driven learning work utilizes *Computational Fluid Dynamics* (CFD) approach to generate the data sets [13, 14, 15], and it does not really solve the demand of avoiding the big computational cost of discrete methods.

In order to remedy the above-mentioned shortcomings, physics-driven methods are a relatively new development. By providing physics information, NN are able to directly obtain the field solution with much less or even no training data. Based on *Multi-layer Perceptron* (MLP) [16], Raissi et al. designed a *Physics Informed Neural Networks*(PINN). Due to the constraint of loss function employing *Partial Differential Equations* (PDEs), the outputs gradually approach the ones obeying the physics laws [17, 18, 19]. Also, Sun et al. used MLP to predict fluid flows, in which a specific prior ansatz was devised to force the network satisfying the geometric boundary of a flow [20]. However, due to the full connectivity between the neurons, MLP suffer from extensive memory requirements and statistical inefficiencies [21]. Therefore, it is difficult to handle well the multi-dimensional learning space with high-resolution physics fields

containing much more details. Taking the highest resolution solution in Ref. [22] as an example, the MLP with one single temperature channel has over 1 million weights. Considering more complex fluid dynamics problems requiring multiple feature channels, the weight count would increase even further. One avenue for alleviating this problem is to employ the reduced-order modeling to compress and reconstruct the flow fields apart from training the network [10, 23]. However, this operation not only is complicated but also may introduce additional errors from the projection onto reduced space [24]. In addition, MLP architecture by itself does not take into account the spatial structure of data. The data points in the learning domain irrespective of their distance are treated in a same way [25]. However, the physics laws represented with PDEs are based on the localities of data points, which suggests that the capability of NN to reflect this spatial relationship can be very important, especially for the physics-driven methods which are constrained only by PDEs.

On the other hand, *Convolutional Neural Networks*(CNN) represent a specialized and well-established type of NN to tackle the aforementioned challenges [26]. In previous research using data-driven methods, CNNs have presented a good performance to predict high-fidelity physics solutions. E.g., without an extra reduced-order modeling step, the CNN can directly compress and reconstruct high-fidelity flow fields with a series of convolutional calculations [7]. In physics-driven methods, CNNs succeeded in solving simple physics problems which obey a single PDE, such as Laplace equation [22] and Darcy's law [27], achieving high computational efficiency in capturing multi-scale features of the physics fields. Meanwhile, CNNs also show the capacity to learn spatial connections between the adjacent data points [28], or the long-term control of fluids with physical losses [29].

In this paper, based on the idea of physics constraints and a specific CNN architecture, we propose a *Physics-driven Convolutional Neural Networks* (PD-CNN) method. With this method, *Navier-Stokes* (N-S) equations and boundary conditions are introduced as a loss function that is discretized on the computational mesh in a controlled manner. The geometry of the object and other

flow conditions are additionally embedded in the input layer. To our knowledge, this is the first attempt that using complex, discrete PDE formulations to drive CNNs for predicting physics fields.

## 2. Methodology

In this section, the CNN architecture used to compress input and to reconstruct the output are described. Then the physics-driven learning framework for N-S equations is introduced, while an accelerating approach employing reference targets is presented at last.

### *2.1. U-net architecture of CNN*

The U-net is a widely-used architecture of CNN which is first designed for biomedical image segmentation [30] and has previously been used for flow field reconstruction with data-driven learning [7]. In this paper, we modify it to suit physics-driven learning approaches.

As Figure 1 shows, including the input and output layers, the U-net architecture consists of 17 layers and corresponding convolutional blocks. The input layer consists of four channels. The first two, $u_0$ and $v_0$, are inflow velocities in both $x$ and $y$ directions, which are uniform non-dimensional values in the whole learning domain. The geometry channel $G$ describes the shape of the object in the flow fields. When there is an object in the flow, all values inside it be marked as 1 and the other as 0. In this way, the geometry is embedded into the network and it is also used for evaluating physics loss as shown in later discussion. To study the capability of the proposed method in characterizing the different patterns of flow, the Reynolds number is introduced as the fourth channel with the definition

$$Re = \frac{\rho v L}{\mu} \tag{1}$$

where $\rho$ is density, $v$ inflow velocity, $L$ characteristic length, and $\mu$ the dynamic viscosity. Since the $\rho$, $v$, $L$ are all unit values in this paper, $Re$ is only depends on $\mu$. The output layer consists of three channels $u$, $v$, and $p$, which are velocities

in both x and y directions and pressure respectively. These outputs are also non-dimensional values.

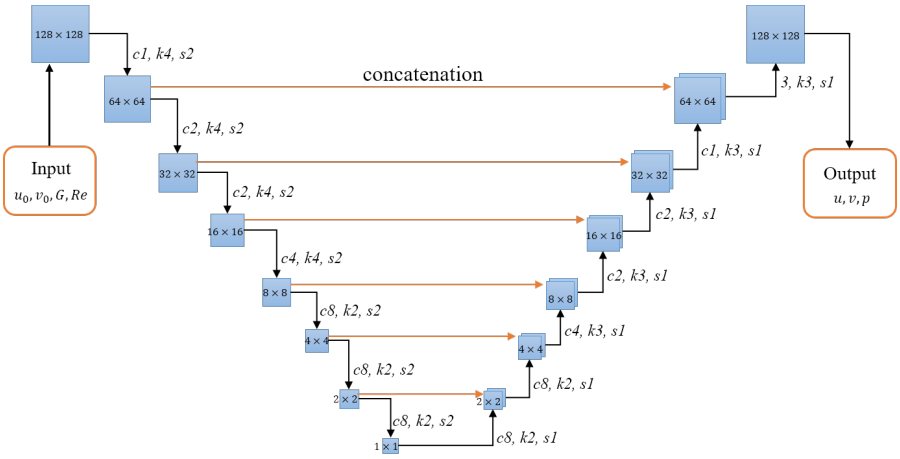



Figure 1: Schematic of U-net architecture. Each blue box corresponds to a multi-channel feature map generated by a convolutional layer. The resulting size are denoted in the box. Black corner arrows denote the down-sampling or up-sampling operation through convolutional layers. $cX, kX, sX$ shortly denotes channel factor $c = X$, convolutional kernel size $k = (X, X)$, stride $s = X$ respectively. And the channel number is the product of $c$ and a basic multiplier 64. The activation functions *Rectified Linear Unit* (ReLU) is used to introduce the non-linearity. Orange arrows denote "skip-connections" via concatenation.

From inputs towards outputs, the network consists of two processes: encoding and decoding. In the encoding process, the input fields are progressively down-sampled by convolutional calculations with corresponding kernels. In this way, the matrices with a size of $128 \times 128$ are reduced to a 512 component vector. The decoding part works in an opposite manner, using an inverse convolutional process mirroring the behavior of the encoding part. Along with the increase of spatial resolution, the flow fields are reconstructed by up-sampling operations and convolutions.

The core target of this work is to generate a flow field constrained by given physics laws. For steady problems, the PDEs, i.e. the mathematical expressions of underlying physics laws, represent the spatial relationships of adjacent positions. Similarly, the convolutional kernels extract the spacial feature of the

receptive field consisting of a group of adjacent pixels. In the U-net architecture we used for our CNN, the encoding part is responsible for recognizing the geometry and inflow conditions of the flow field, in order to extract the necessary features representing the physics of the inputs using convolution operation layer by layer. These features are the basis for the subsequent decoding part. Here, the layers of the decoding process at different depths store the physical feature maps and the spacial relationship are recovered by the inverse convolutional calculation. Eventually, the decoding part is able to reconstruct the proper flow field under the constrain of PDEs. In addition, there are the concatenations of the feature channels between encoding and decoding as the orange arrows denoted in Figure 1. Duplicating the feature channels from the encoding blocks to the corresponding decoding ones, the "skip connections" effectively double the number of feature channels in each decoding layer and enable the network to consider the information from the encoding layers, which extracted from the geometry and inflow conditions.

The architecture used in this paper is symmetrical, which means the encoding and decoding processes have the same depth, meanwhile, the amounts and dimensions of corresponding blocks are the same. However, the depths of the two processes are both adjustable. A coarse input compressed by fewer encoding layers is also able to generate a high-resolution solution reconstructed by more decoding layers. More details of the U-net architecture and convolutional block, including activation function, pooling, and dropout, can be found in Ref. [7] and [31].

### *2.2. Physics-driven learning*

The PDEs controlling the behavior of Newtonian fluid are Navier-Stokes (N-S) Equations. In our study, the steady and incompressible form of N-S Equations is chosen as follows:

$$\nabla \cdot \boldsymbol{U} = 0, \tag{2}$$

$$\boldsymbol{U} \cdot \nabla \boldsymbol{U} + \nabla P - \mu \nabla^2 \boldsymbol{U} = 0, \tag{3}$$

where $\boldsymbol{U} \equiv \boldsymbol{U}(u, v)$, $P$, $\mu$ are velocity, pressure and viscosity respectively. Equation (2) is the continuity equation, which imposes the incompressibilities of the fluid. Equation (3) is the momentum conservation equation, in which the first term represents the momentum convection, $\nabla P$ the pressure gradient and $\mu \nabla \boldsymbol{U}$ the viscous dissipation. As shown in Figure 2, once the preliminary flow field

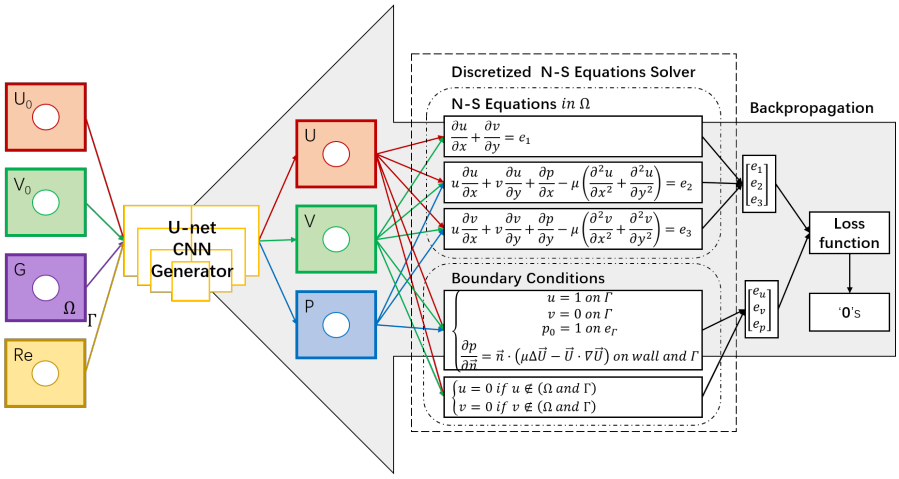



Figure 2: Physics-driven learning for solving the N-S equations. The U-net CNN generates the solution. The backpropagation computes the gradient of the loss function and updates the weights of the CNN to satisfy the discretized N-S equations and boundary conditions.

is obtained from the U-net CNN generator, we apply the physical constraint to this field. The learning domain is separated as inner domain and boundaries, which are represented by $\Omega$ and $\Gamma$ respectively. In the inner domain $\Omega$, the left hand sides of N-S Equations are employed as loss function and 3 residuals can be obtained as

$$\begin{cases} \dfrac{\partial u}{\partial x} + \dfrac{\partial v}{\partial y} = e_1 \\ u\dfrac{\partial u}{\partial x} + v\dfrac{\partial u}{\partial y} + \dfrac{\partial p}{\partial x} - \mu\left(\dfrac{\partial^2 u}{\partial x^2} + \dfrac{\partial^2 u}{\partial y^2}\right) = e_2 \\ u\dfrac{\partial v}{\partial x} + v\dfrac{\partial v}{\partial y} + \dfrac{\partial p}{\partial y} - \mu\left(\dfrac{\partial^2 v}{\partial x^2} + \dfrac{\partial^2 v}{\partial y^2}\right) = e_3 \end{cases} \tag{4}$$

So the residuals of the N-S equations in $\Omega$ can be represented as $E_\Omega = [e_1, e_2, e_3]^{\mathrm{T}}$.

In order to obtain differentiable formulation of the physics in the loss, we construct suitable convolutional filters to compute the N-S equations via finite differences. Similar approaches have been proposed previously for simple PDEs [22], and the partial differential operators for two different dimensions are constructed separately [32, 33]. The construction via convolutions has the advantage that the backpropagation of a deep learning framework can be used, and the finite difference kernels yield well controlled accuracies for the derivative calculations. Choosing the x direction as an example, the weights of the filters are represented as

$$\mathrm{W}_{\frac{\partial}{\partial \mathrm{x}}} = \begin{bmatrix} 0 & -0.5 & 0 \\ 0 & 0 & 0 \\ 0 & 0.5 & 0 \end{bmatrix}, \mathrm{W}_{\frac{\partial^2}{\partial \mathrm{x}^2}} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \tag{5}$$

After the rotating and moving operation through the matrix obtained from the last layer, the first- or second-order partial derivatives of local quantities are calculated. Choosing $u$ as an example, this procedure can be written as:

$$g_{i,j} = \sum_{m=0}^{2} \sum_{n=0}^{2} u_{i+m-1,j+n-1} \cdot f_{m,n}, \tag{6}$$

where $f_{m,n}$ is the convolutional filter and the $g_{i,j}$ is the central difference of $u$ in each data point.

On the boundary $\Gamma$, including inflow and outflow side and walls, the Dirichlet and Neumann boundary conditions are considered as shown in 2. The residuals of $u$, $v$ and $p$ are represented as $E_\Gamma = [e_u, e_v, e_p]^{\mathrm{T}}$. Combining both as $E = [E_\Gamma, E_\Omega]^{\mathrm{T}}$, the whole residual of the physics-driven method is obtained.

To reduce the residuals, the CNN is trained in an iterative manner using a stochastic gradient descent variant (we employ Adam [34]). After the CNN generator, the preliminary flow fields are introduced in the loss function, and then the residuals $E$ are obtained. Once the backpropagation is applied, the weights and bias of CNN are adapted to minimize these physics residuals. Eventually, the high-resolution flow fields which obey N-S Equations and corresponding

boundary conditions can be obtained.

### *2.3. Acceleration with reference targets*

In the process of the physics-driven learning, the weights of CNN are adapted only to minimize the residuals of PDEs, the solution itself is not constrained, which means there is no target being offered for reference. In order to accelerate the convergence and eventually improve the training performance, besides the physical laws, we provide additional reference targets for constraining the network.

Similar to data-driven methods, there is a reference loss term comparing the difference between output and target, which is defined as

$$\mathcal{L}_{\mathrm{ref}} = \sum_{i=1}^{\mathcal{I}} \sum_{n=1}^{\mathcal{N}} \left| \mathcal{X}_{\mathrm{out}} - \mathcal{X}_{\mathrm{tar}} \right| . \tag{7}$$

The subscript "ref" here denotes reference targets. Generally, $\mathcal{X}_{\mathrm{out}}$ and $\mathcal{X}_{\mathrm{tar}}$ are output quantities and corresponding targets, respectively, $\mathcal{I} = \mathit{targets\ amount}$, meanwhile $\mathcal{N} = \mathit{batch\ size}$ denotes the amount of training data in one batch operation. The total loss considering both reference targets and physics laws can be represented as

$$\mathcal{L} = \mathcal{L}_{\mathrm{ref}} + \mathrm{R} * \mathcal{L}_{\mathrm{phy}}, \tag{8}$$

where $\mathcal{L}_{\mathrm{phy}}$ is the physics loss term considering the N-S equations and boundary conditions. R is a constant hyperparameter which is tuned to adapt the scales. With this weighted loss function, the different loss terms can be easily scaled to an equivalent magnitude.

In the physics-driven training, a certain amount of randomly picked Reynolds numbers are input as one batch in each iterative step. In contrast, in the accelerating approach with reference targets, we also use some constant Reynolds numbers besides the variable ones. So, the new batch includes two groups as shown in Figure 3. The cases in the random group vary in every iterative step and are trained with physics loss $\mathcal{L}_{\mathrm{phy}}$. While the ones in constant group are fixed in the whole iterative process and are trained with reference loss $\mathcal{L}_{\mathrm{ref}}$.

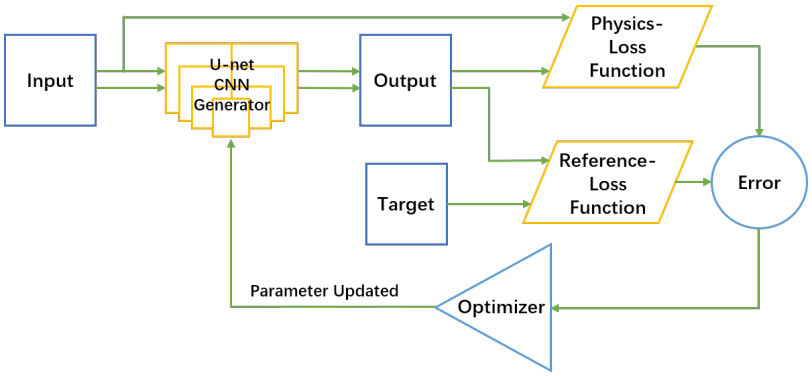



Figure 3: Acceleration with reference targets. One batch consists of the physics and reference groups. The outputs of physics group are introduced to physics loss function, while the outputs of reference group are introduced to the reference loss function.

This accelerating approach using reference targets is merely an enhancement of the original physics-driven method, which means the PD-CNN without references is sufficient to predict the final solution of the flow field. In order to clearly represent this property, section 3 only presents the physics-driven-alone results while the enhancement of this reference acceleration will be discussed in section 4.

## 3. Results

### *3.1. Overview*

In this section, several numerical experiments are conducted to estimate the capability of the proposed PD-CNN framework to predict steady laminar flow fields. The experiments follow two distinct patterns, single case and multiple cases.

Single case means the PD-CNN only predicts the solution of one specific flow field. Given one specific combination of boundary conditions and fluid properties, the network is trained and then fixed to generate the unique corresponding solution. This pattern is similar to a traditional CFD simulation or a

normal PINN training, and it is used to estimate the capacity of PD-CNN as an alternative to solve a specific problem.

Multiple cases mean the PD-CNN is used to obtain the solutions of multiple cases with a unique network. The input inflow conditions are varied in a moderate range in the training stage. After that, given any of parameter combinations inside the span, the fixed network is able to generate the corresponding flow field. We use this pattern to estimate the capability of the PD-CNN learning different physics and whether these physics can coexist within one unique network. Compared with CFD simulation, this capability allows the trained network to directly generate the solutions under different conditions with tiny computational cost, which makes the real-time or many-query analysis feasible.

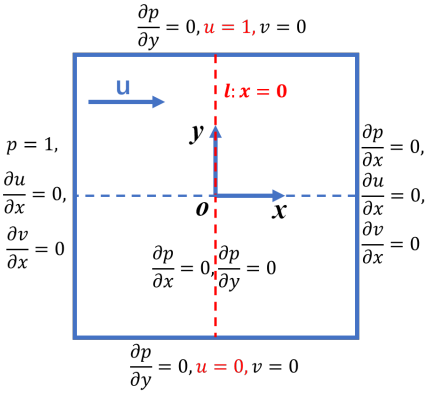
In the study of single case, we choose several different 2D cases, such as Couette flow, Poiseuille flow, and flow around a cylinder. While for multiple cases, there are only cases of flow around cylinder. All of them are steady-state flows with low Reynolds numbers. For the Couette and Poiseuille flow, analytical solutions are used as references. For flow around cylinder cases, the numerical results calculated by the mature *Finite Volume Method* (FVM) are used as references [35]. The mesh for simulations is generated by Gmesh [36] and an unstructured grid. Then the numerical solutions are interpolated into a Cartesian grid to compare with the PD-CNN results.

### *3.2. Single case*

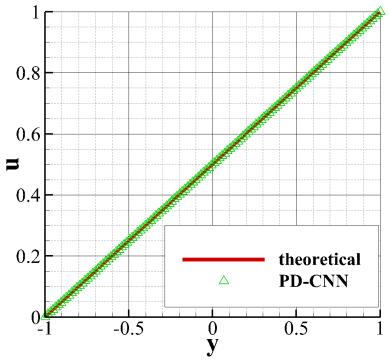
#### *3.2.1. Couette flow*

Couette flow is the flow of a viscous fluid in the space between two surfaces moving relatively, which is frequently used in engineering courses to illustrate shear-driven fluid motion. In our case, the top surface is moving along the positive direction of x-axis with a constant speed, and the bottom surface keeps still as shown in Figure 4a. The results on the detecting line $l : x = 0$ and the entire domain are shown in 4b and 4c respectively.

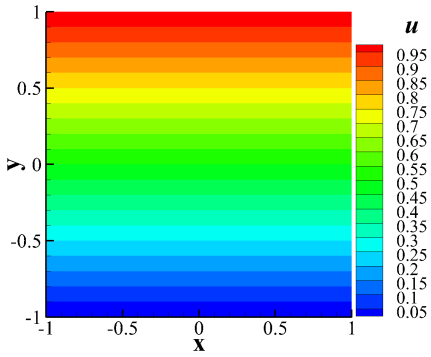
The Couette flow is a tangential velocity-driven flow. The interference from $p$ and $v$ are relatively low. The only term that actually affects the flow is $\partial^2 u/\partial y^2$,




(a) Schematic




(b) learning results on $l$




(c) learning results in the whole domain

Figure 4: Couette Flow.The learning domain is $x \in [-1, 1]$ and $y \in [-1, 1]$. The speed of top surface is $u = 1$. And the viscosity coefficient $\mu = 1$. For detecting line $l : x = 0$, the angle between velocity profile line and horizontal axis is almost exactly 45°, which is the same as theoretical value. For the whole domain, $u$ along x-axis is isotropic which match the zero-gradient condition of $p$. Furthermore, there is no discontinuity on the boundary.

which is the linear term of N-S Equations. Thus, this case tests the ability of PD-CNN to learn linear processes. It also means that a Laplace-like equation can be learned by the PD-CNN.

*3.2.2. Poiseuille flow*

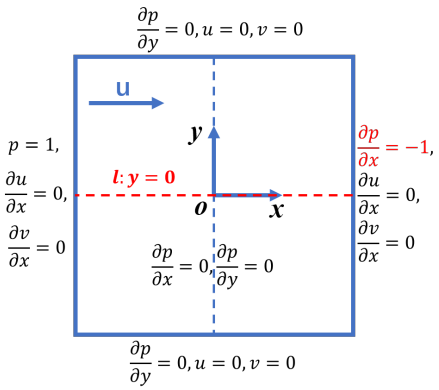
A pressure-driven incompressible flow between two surfaces is calculated in this section. The boundary conditions are shown in Figure 5a. In the entire field, a gradient of pressure is imposed, which is the driven force of the movement of flow. For Poiseuille flow, the flow fields with different $\mu$ are the same, which is presented very well as shown in Figure 5b. This property indicates the rationality that designing multiple cases by altering $\mu$ in sub-section 3.3.

Although Poiseuille flow is also a linear flow, the flow is induced by the pressure gradient. The properties of all three variables, $u$, $v$, and $p$, are well represented. And the obtained flow field is swirl-free and symmetric. This proves that our treatment of the physics field is isotropic, and the error is effectively controlled.

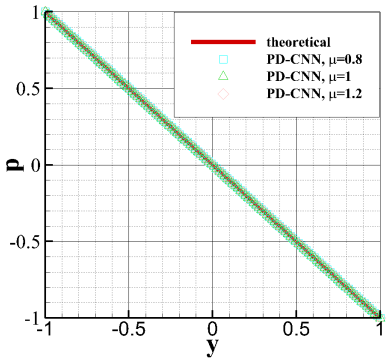
*3.2.3. Flow around cylinder*

In this section, the network is trained to solve a flow field around a cylinder(as shown in Figure 6). Flow around a cylinder is a classic problem in fluid mechanics and its flow field exhibits different physical phenomena under different Reynolds numbers. Here, we consider solving the steady incompressible flows with Reynolds number from 1 to 20. According to Ref. [37], this range covers the two different regimes which are designated as creeping laminar state (L1) and laminar flow with steady separation (L2). In general, when the Reynolds number is smaller than approximately 5, the flow field will be the creeping laminar flow. Once the Reynolds number is larger than approximately 5, the flow field will transfer to a steady flow with a pair of symmetric trapped vortexes behind the cylinder. These two types of flow regimes are very different.

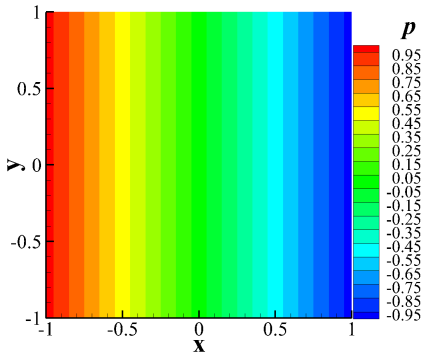
Figure 7 and Figure 8 show the flow fields predicted by PD-CNN with $Re = 1$ and $Re = 20$ respectively. The learning results agree the numerical results obtained by FVM well.




(a) Schematic




(b) learning results on $l$




(c) learning results in the whole domain. $\mu = 1$.

Figure 5: Poiseuille Flow. The learning domain is $x \in [-1, 1]$ and $y \in [-1, 1]$. Both top and bottom wall are stationary walls. For detecting line $l : y = 0$, for all different $\mu$, the angle between velocity profile line and horizontal axis are almost exactly 135°, which is the same as theoretical values. For the whole domain, $p$ along y-axis is isotropic and there is no discontinuity on the boundary.

### *3.3. Multiple cases*

For multiple cases training about flow around a cylinder, the CNN is trained with the variation of Reynolds number while the U-net architecture and inflow conditions are fixed. The Reynolds numbers, which are represented as the reciprocal of dynamic viscosity, are randomly picked between 0.8 and 20.2 in the

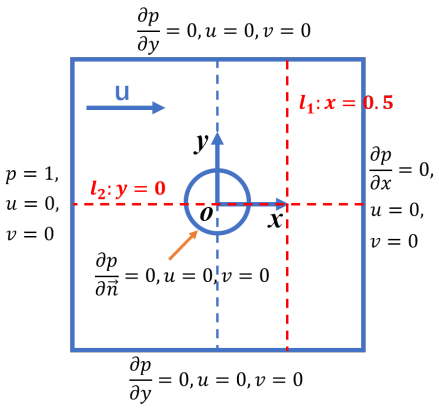
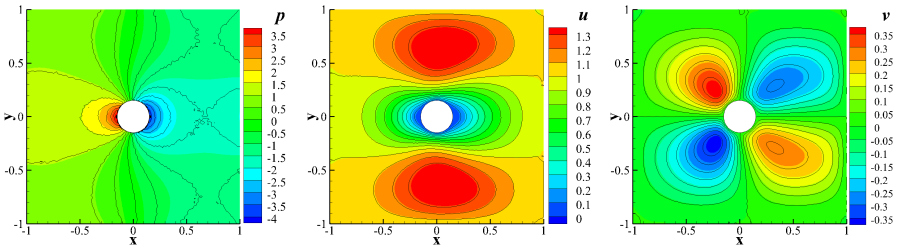



Figure 6: Schematic of flow around cylinder. The computing domain is $x \in [-1, 1]$ and $y \in [-1, 1]$ and diameter of the central cylinder is 0.3. There are two detecting lines $l_1$ and $l_2$ for later discussion.

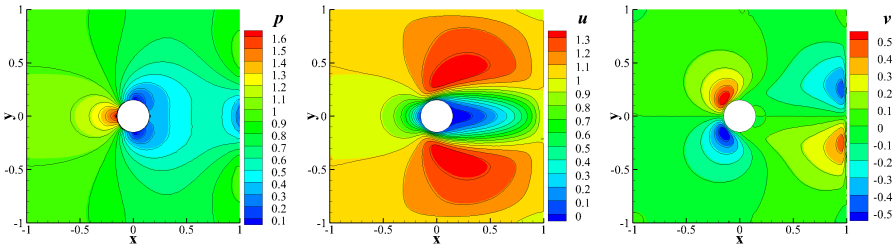



(a) $p$ (b) $u$ (c) $v$

Figure 7: Single case when $Re = 1$. Black contour lines are FVM results, colorful contour flooding is PD-CNN results. For velocities, the learning results agree reference quite well. While for pressure, especially in the region behind the cylinder, the learning results have a slight deviation.

whole training process. It is similar to the generation of a large training set in data-driven methods. The random variation of the Reynolds number provides the PD-CNN inexhaustible training cases and forces the networks to consider the whole span.

Figure 9 shows the capability of PD-CNN to reconstruct the two flow regimes L1 and L2. After the L1 regime, the flow separates on the cylinder surface and the wake behind started to form the contra-rotating vortices. In the whole range

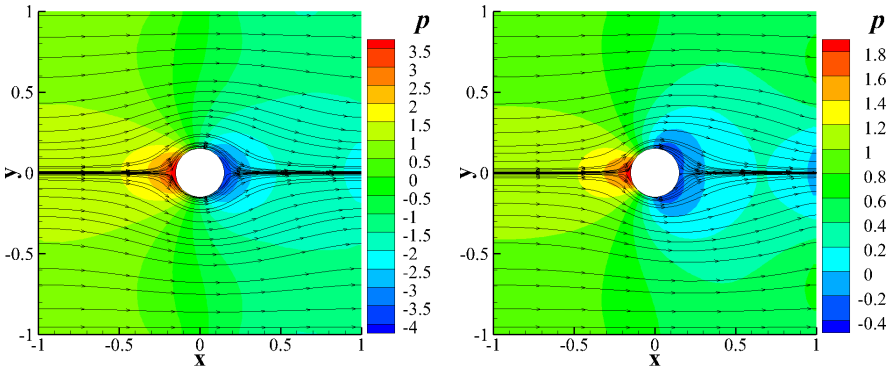



(a) $p$ (b) $u$ (c) $v$

Figure 8: Single case when $Re = 20$. Black contour lines are FVM results, colorful contour flooding is PD-CNN results. For all velocities and pressure, the learning results agree reference quite well.

of L2, the pair of symmetric vortices, adhere stably behind the cylinder. For the relatively small value of the Reynolds number, the closed wake region which contains the "twin-vortices" is presented clearly in the rear of the cylinder. And with the Reynolds number increasing, this pair of vortices grows and becomes more and more elongated in the flow direction. The results talked above show a unique network obtained by PD-CNN is able to learn the different physical properties and predict the distinct flow fields according to input parameters.

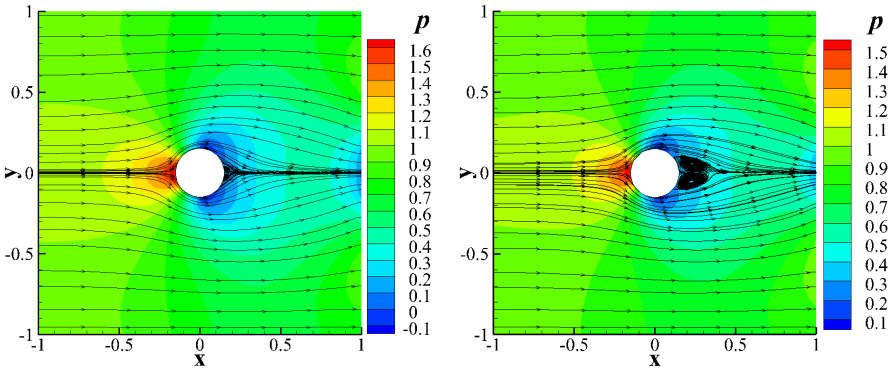
The contra-rotating vortices can be noticed from the stream trace pictures of $Re = 8$ but not $Re = 7$. But it can not be concluded that the critical Reynolds number at which the pair of symmetric contra-rotating vortices begin to form is between 7 and 8. Because in the neighborhood of the critical Reynolds number, the dimension of the contra-rotating vortices is very small and can not be directly observed in the stream trace pictures. By employing the method proposed by Taneda [38], the critical Reynolds number can be estimated. As shown in Figure 10, the sizes of the twin-vortices are measured against the Reynolds numbers. It can be observed that as the Reynolds number increases, the length of twin-vortices grows. Hence, from the linear curve fit the critical Reynolds number can be deduced. So the pair of the contra-rotating vortices begin to form in the rear of the cylinder at $Re = 6.63$, which is close with the FVM result, $Re = 6.1$ [39].




(a) $Re = 1$




(b) $Re = 5$




(c) $Re = 10$




(d) $Re = 20$

Figure 9: Stream Traces with different $Re$. The results discriminate two different flow regimes and represent the evolution of the vortices along with the growing of $Re$. After the creeping flow regime, (a) and (b), a pair of symmetric contra-rotating vortices appears in the rear of the cylinder, (c) and (d).

When $Re < 20.2$, reasonably good agreements are observed between the learning results and the numerical solutions for the size of the closed wake. We also evaluate the extrapolation capabilities of the network with the Reynolds numbers between 20.2 and 25. However, as the comparison shows, the predicted length of the twin-vortices has bigger deviations and the extrapolated flow fields don't represent reliable references. It can be concluded that for the PD-CNN it is hard to accurately recover the neighboring flow fields which are

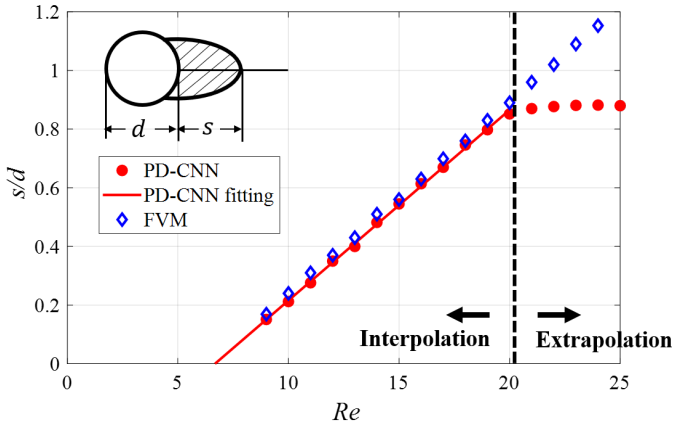
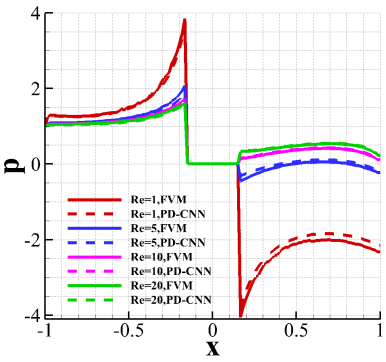


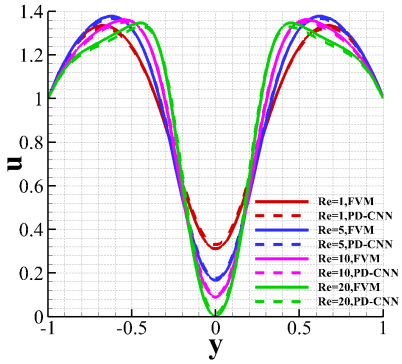

Figure 10: Size of the vortices pair against $Re$. By linear fitting, the critical Reynolds number at which the vortices pair begin to form in the rear of a cylinder is obtained.

completely missing at the learning stage. This phenomenon reflects the fundamentally interpretive characteristic of NN modeling and the model is merely well approximated in the training span [5]. If the objective field is relatively complex and the trainable parameters of NN are extensive, the network deteriorates the extrapolation accuracy badly.

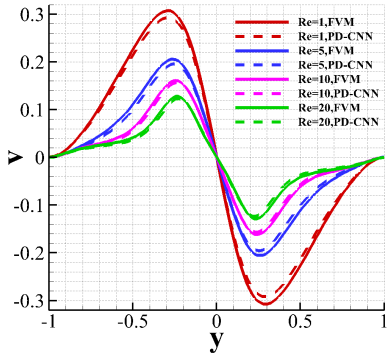
Figure 11 shows the comparison between the learning results and the numerical solutions. All the learning results of these four cases are in good agreement with the FVM results. But the values have an "even-bias". Compared with the ground truth, all values are closer to the mean value in the whole domain. For $p$, the mean value is in the vicinity of 0. For $u$, it is 0.7, and $v$ is 0. That means the generator has a trend to predict all the flow fields in the whole parameter space more evenly. In addition, the predictive flow fields which have lower Reynolds number have bigger deviations. Choosing $p$ as an example, the flow fields with relatively big Reynolds number have better accuracy, while the case of $Re = 1$ has a bigger deviation. In the generation of the training cases, the Reynolds numbers are chosen randomly between 0.8 and 20.2, which means the distribution of training cases is even. However, for the case with $Re = 1$, the flow field is more distinctive and requires more training effort. So, when designing the




(a) $p$




(b) $u$




(c) $v$

Figure 11: Comparison between PD-CNN and FVM. The pressure is about $l_1 : x = 0.5$ and the velocities are about $l_2 : y = 0$. The results of lower $Re$ have relatively larger deviations.

training space, more training cases should be distributed near these exceptional parameter combinations.

## 4. Discussion

### *4.1. Accuracy*

The drag and lift coefficient of a cylinder can be expressed as follows [40]

$$C_{\mathrm{d}} = \frac{2F_{\mathrm{D}}}{\rho U^2 A},\ C_{\mathrm{l}} = \frac{2F_{\mathrm{L}}}{\rho U^2 A}, \tag{9}$$

where $A$ represents the frontal area. $F_{\mathrm{D}}$ and $F_{\mathrm{L}}$ denote the drag and lift force respectively. When the mean physical residual $E$ of 1k epochs is lower than 0.4%, the training stops and the coefficients are calculated. These two coefficients obtained by PD-CNN are shown in Table 2. For $C_{\mathrm{d}}$, PD-CNN with the resolution $384 \times 384$ predicts achieve a relative error less than 2.9%. The value of $C_{\mathrm{l}}$ is very small and the learning results are in the same magnitude compared with FVM references. At the same time, it can be clearly seen that with the resolution increasing, both the drag and lift coefficients gradually approach the numerical solutions, which proves that the field solutions obtained by PD-CNN are convergent.

In addition, we define the order of convergence as

$$Order = |\frac{\log(e_i/e_j)}{\log(h_i/h_j)}|, \tag{10}$$

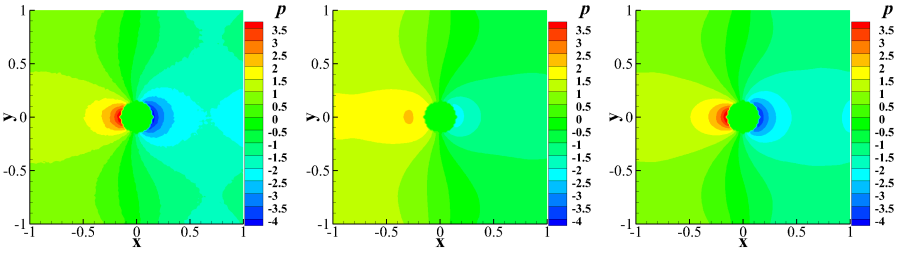
where $e$ is the error of drag or lift coefficient under different resolutions, and $h$ is the resolution size. Based on this definition the PD-CNN can achieve approximate first-order accuracy.

Table 1: Accuracy of PD-CNN with different resolutions

| Coefficient | Resolution | $Re = 1$ | | | $Re = 20$ | | |
|---|---|---|---|---|---|---|---|
| | | Result | Error | Order | Result | Error | Order |
| $C_{\mathrm{d}}$ | FVM | 44.3 | - | - | 5.96 | - | - |
| | $128 \times 128$ | 39.7 | -4.6 | - | 5.52 | -0.44 | - |
| | $256 \times 256$ | 42.3 | -2.0 | 1.21 | 5.71 | -0.25 | 0.82 |
| | $384 \times 384$ | 43.0 | -1.3 | 1.08 | 5.89 | -0.07 | 3.09 |
| $C_{\mathrm{l}}$ | FVM | 0.0028 | - | - | 0.0097 | - | - |
| | $128 \times 128$ | -0.059 | -0.0618 | - | -0.044 | -0.0537 | - |
| | $256 \times 256$ | -0.015 | -0.0178 | 1.79 | -0.013 | -0.0227 | 1.24 |
| | $384 \times 384$ | -0.0059 | -0.0087 | 1.76 | 0.0028 | -0.0069 | 2.94 |

*4.2. Acceleration*

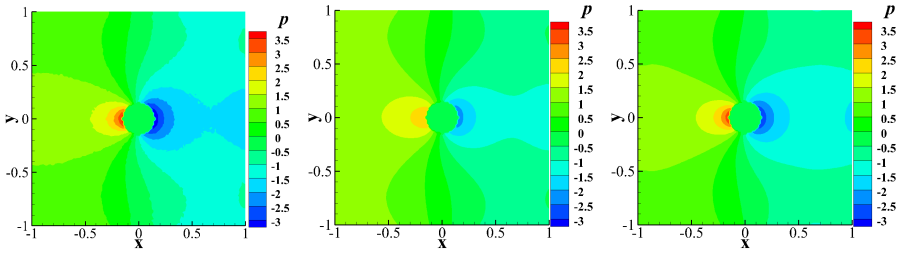
In the numerical experiments of PD-CNN which are accelerated with reference targets, there are 18 cases in each batch. The first half batch consists of 9 random $Re$ varied with epoch number, while the other half batch consists of 9 constant $Re$ fixed in the whole training process. According to Equation (8), a moderate number of constant $Re$ for the reference targets requires being defined beforehand. As discussed in section 3.3, the cases whose Reynolds numbers are near to 1 are much more difficult to train. So, the manually defined 9 constant Reynolds numbers are 1.0, 1.5, 2.0, 3.0, 4.0, 6.0, 8.0, 12.0 and 18.0. In the whole training process, the numerical solutions obtained by FVM of these 9 constant Reynolds numbers are input as nine targets and the CNN is trained to minimize Equation (7).




(a) FVM (b) PD-CNN (c) PD-CNN with references

Figure 12: Comparison of pressure field of different methods when $Re = 1$ (epoch=10k). For this typical case with reference, the training with targets is able to obtain accurate solution faster.

In this way, the reference targets restrain the results generated by the physics-driven method to approximate real solutions faster (as shown in Figure 12 and Figure 13). Since the targets only include a limiting number of cases and are used through the whole training process, this approach reduces the expensive data generation cost of the traditional data-driven methods. In practical engineering applications, the reference targets can be easily picked from the existing data, e.g., experimental and numerical results.




(a) FVM (b) PD-CNN (c) PD-CNN with references

Figure 13: Comparison of pressure field of different methods when $Re = 1.2$ (epoch=10k). For this typical case without reference, the training with targets is also able to obtain accurate solution faster.

## 5. Conclusion

In this paper, we proposed a physics-driven method based on a CNN with a U-net structure. By introducing the discretized Navier-Stokes equations and boundary conditions as the loss function, the CNN is able to directly predict steady-state laminar flow fields. Compared with the MLP used in previous research, the CNN is able to embed the objective geometry and flow structure into the latent space, and then decode them to reconstruct the corresponding flow fields with physics consistency. The PD-CNN is able to obtain accurate solutions for both single case and multiple cases. In the multiple cases prediction of flow around cylinder, PD-CNN is capable of describing the transformation of the "twin-vortices" and obtaining the critical Reynolds number between creeping laminar state and laminar flow with separation. By constraining with a small number of reference targets, the network training was accelerated, especially for the difficult cases. Further research will be carried out for predicting complex flow fields in practical engineering problems with the present method.

## 6. Acknowledgement

Hao Ma (No. 201703170250) and Yuxuan Zhang (No. 201804980021) are supported by China Scholarship Council when they conduct the work this paper

represents.

## References


[1] L. Carr, M. Chandrasekhara, N. Brock, Quantitative study of unsteady compressible flow on an oscillating airfoil, Journal of aircraft 31 (4) (1994) 892–898.

[2] T. Ray, H. Tsai, Swarm algorithm for single-and multiobjective airfoil design optimization, AIAA journal 42 (2) (2004) 366–373.

[3] A. Sharma, M. S. Sarwara, H. Singh, L. Swarup, R. K. Sharma, Cfd and real time analysis of a symmetric airfoil, International Journal of Research in Aeronautical and Mechanical Engineering 2 (7) (2014) 84–93.

[4] J. Yu, J. S. Hesthaven, Flowfield reconstruction method using artificial neural network, Aiaa Journal 57 (2) (2019) 482–498.

[5] S. L. Brunton, B. R. Noack, P. Koumoutsakos, Machine learning for fluid mechanics, Annual Review of Fluid Mechanics 52 (2019). arXiv:1905.11075v3.

[6] K. Duraisamy, G. Iaccarino, H. Xiao, Turbulence modeling in the age of data, Annual Review of Fluid Mechanics 51 (2019) 357–377. arXiv:1804.00183v3.

[7] N. Thuerey, K. Weißenow, L. Prantl, X. Hu, Deep learning methods for reynolds-averaged navier–stokes simulations of airfoil flows, AIAA Journal 58 (1) (2020) 25–36.

[8] J. Ling, A. Kurzawski, J. Templeton, Reynolds averaged turbulence modelling using deep neural networks with embedded invariance, Journal of Fluid Mechanics 807 (2016) 155–166.

[9] E. J. Parish, K. Duraisamy, A paradigm for data-driven predictive modeling using field inversion and machine learning, Journal of Computational Physics 305 (2016) 758–774.

[10] J. S. Hesthaven, S. Ubbiali, Non-intrusive reduced order modeling of nonlinear problems using neural networks, Journal of Computational Physics 363 (2018) 55–78.

[11] M. D. Ribeiro, A. Rehman, S. Ahmed, A. Dengel, Deepcfd: Efficient steady-state laminar flow approximation with deep convolutional neural networks, arXiv preprint arXiv:2004.08826 (2020).

[12] Z. Bai, S. L. Brunton, B. W. Brunton, J. N. Kutz, E. Kaiser, A. Spohn, B. R. Noack, Data-driven methods in fluid dynamics: Sparse classification from experimental data, in: Whither Turbulence and Big Data in the 21st Century?, Springer, 2017, pp. 323–342.

[13] X. Guo, W. Li, F. Iorio, Convolutional neural networks for steady flow approximation, in: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016, pp. 481–490.

[14] S. Bhatnagar, Y. Afshar, S. Pan, K. Duraisamy, S. Kaushik, Prediction of aerodynamic flow fields using convolutional neural networks, Computational Mechanics (2019) 1–21arXiv:1905.13166v1.

[15] H. Ma, Y.-x. Zhang, O. J. Haidn, N. Thuerey, X.-y. Hu, Supervised learning mixing characteristics of film cooling in a rocket combustor using convolutional neural networks, Acta Astronautica (2020).

[16] D. Ciregan, U. Meier, J. Schmidhuber, Multi-column deep neural networks for image classification, in: 2012 IEEE conference on computer vision and pattern recognition, IEEE, 2012, pp. 3642–3649.

[17] M. Raissi, G. E. Karniadakis, Hidden physics models: Machine learning of nonlinear partial differential equations, Journal of Computational Physics 357 (2018) 125–141.

[18] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems

involving nonlinear partial differential equations, Journal of Computational Physics 378 (2019) 686–707. doi:10.1016/j.jcp.2018.10.045.

[19] L. Lu, X. Meng, Z. Mao, G. E. Karniadakis, Deepxde: A deep learning library for solving differential equations, arXiv preprint arXiv:1907.04502 (2019).

[20] L. Sun, H. Gao, S. Pan, J.-X. Wang, Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data, Computer Methods in Applied Mechanics and Engineering 361 (2020) 112732.

[21] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT press, 2016.

[22] R. Sharma, A. B. Farimani, J. Gomes, P. Eastman, V. Pande, Weakly supervised deep learning of heat transport via physics informed loss, arXiv preprint arXiv:1807.11374 (2018). arXiv:1807.11374v2.

[23] Q. Wang, J. S. Hesthaven, D. Ray, Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to a combustion problem, Journal of computational physics 384 (2019) 289–307.

[24] C. W. Rowley, T. Colonius, R. M. Murray, Model reduction for compressible flows using pod and galerkin projection, Physica D: Nonlinear Phenomena 189 (1-2) (2004) 115–129.

[25] H. H. Le, N. H. Nguyen, T.-T. Nguyen, Automatic detection of singular points in fingerprint images using convolution neural networks, in: Asian Conference on Intelligent Information and Database Systems, Springer, 2017, pp. 207–216.

[26] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105.

[27] Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, P. Perdikaris, Physics-constrained deep learning for high-dimensional surrogate modeling and un-

certainty quantification without labeled data, Journal of Computational Physics 394 (2019) 56–81. arXiv:1901.06314v1.

[28] H. Ma, X. Hu, Y. Zhang, N. Thuerey, O. J. Haidn, A combined data-driven and physics-driven method for steady heat conduction prediction using deep convolutional neural networks, arXiv preprint arXiv:2005.08119 (2020).

[29] P. Holl, V. Koltun, N. Thuerey, Learning to control pdes with differentiable physics, in: International Conference on Learning Representations (ICLR), 2020.

[30] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: International Conference on Medical image computing and computer-assisted intervention, Springer, 2015, pp. 234–241.

[31] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., Pytorch: An imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems, 2019, pp. 8024–8035.

[32] Z. Long, Y. Lu, X. Ma, B. Dong, Pde-net: Learning pdes from data, in: International Conference on Machine Learning, PMLR, 2018, pp. 3208–3216.

[33] Z. Long, Y. Lu, B. Dong, Pde-net 2.0: Learning pdes from data with a numeric-symbolic hybrid deep network, Journal of Computational Physics 399 (2019) 108925.

[34] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[35] 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.

[36] C. Geuzaine, J.-F. Remacle, Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities, International journal for numerical methods in engineering 79 (11) (2009) 1309–1331.

[37] M. Zdravkovich, P. Bearman, Flow Around Circular Cylinders—Volume 1: Fundamentals, 1998.

[38] S. Taneda, Experimental investigation of the wakes behind cylinders and plates at low reynolds numbers, Journal of the Physical Society of Japan 11 (3) (1956) 302–307.

[39] B. Rajani, A. Kandasamy, S. Majumdar, Numerical simulation of laminar flow past a circular cylinder, Applied Mathematical Modelling 33 (3) (2009) 1228–1247.

[40] M. Schäfer, S. Turek, F. Durst, E. Krause, R. Rannacher, Benchmark computations of laminar flow around a cylinder, in: Flow simulation with high-performance computers II, Springer, 1996, pp. 547–566.

## A.4 Paper IV

H. Ma, B. T. Zhang, C. Zhang and O. J. Haidn

### Generative adversarial networks with physical evaluators for spray simulation of pintle injector

In *AIP Advances*, Volume 11, 2021, pp. 075007, DOI: `https://doi.org/10.1063/5.0056549`.



*Contribution*: My contribution to this work was the proposal of the method and the corresponding computer code for its implementation. I have performed numerical experiments, analyzed the results, and wrote the manuscript for publication.

# Generative adversarial networks with physical evaluators for spray simulation of pintle injector

Cite as: AIP Advances 11, 075007 (2021); doi: 10.1063/5.0056549
Submitted: 11 May 2021 • Accepted: 10 June 2021 •
Published Online: 2 July 2021



Hao Ma,[1,a] Botao Zhang,[2] Chi Zhang,[3] and Oskar J. Haidn[1]

AFFILIATIONS

[1] Department of Aerospace and Geodesy, Technical University of Munich, 85748 Garching, Germany
[2] Key Laboratory for Liquid Rocket Engine Technology, Xi'an Aerospace Propulsion Institute, 710100 Xi'an, China
[3] Department of Mechanical Engineering, Technical University of Munich, 85748 Garching, Germany

[a] Author to whom correspondence should be addressed: hao.ma@tum.de

## ABSTRACT

Due to the adjustable geometry, pintle injectors are especially suitable for liquid rocket engines, which require a widely throttleable range. However, applying the conventional computational fluid dynamics approaches to simulate the complex spray phenomenon in the whole range still remains a great challenge. In this paper, a novel deep learning approach used to simulate instantaneous spray fields under continuous operating conditions is explored. Based on one specific type of neural network and the idea of physics constraint, a *Generative Adversarial Networks with Physics Evaluators* framework is proposed. The geometry design and mass flux information are embedded as inputs. After the adversarial training between the generator and discriminator, the generated field solutions are fed into two physics evaluators. In this framework, a mass conversation evaluator is designed to improve the training robustness and convergence. A spray angle evaluator, which is composed of a down-sampling *Convolutional Neural Network* and theoretical model, guides the networks to generate the spray solutions more closely according to the injection conditions. The characterization of the simulated spray, including the spray morphology, droplet distribution, and spray angle, is well predicted. This work suggests great potential for prior physics knowledge employment in the simulation of instantaneous flow fields.



## I. INTRODUCTION

Due to their wider throttling range and greater combustion stability, pintle injectors are especially suitable for liquid rocket engines that require deep, fast, and safe throttling,[1–3] such as the descent propulsion system in the Apollo program[4] and the reusable Merlin engine of SpaceX.[5]

In practical throttleable engine applications, the pintle is movable to alter the injection area so that the mass flow rate of the injected propellants can be varied continuously according to the economical and safe thrust curve in a given situation.[6] However, in the previous spray simulations of pintle injectors, the changes were only considered under discrete condition combinations over a limited number of select operating points.[7–9] For the traditional discrete methods they used, simulations have to be conducted repeatedly to vary the operating conditions and the computational cost becomes prohibitively expensive.[10] Innovations for the spray simulation of the pintle injector are needed to address this issue.

Contrarily, the machine learning approach, especially the *Neural Network* (NN), has demonstrated its efficiency to predict the flow fields under different conditions with a single surrogate model.[11,12] Previous research studies on flow field prediction using the NN are mainly focused on the data-driven method. In addition to the indirect way using the closure model,[13,14] the field solution can also be directly obtained from the network model, which is trained with a large number of samples.[15–18] However, some predictive results obtained by data-driven methods may still exhibit considerable errors against physics laws or operating conditions.[19–21] In addition, in some sparse data regimes, some machine learning techniques lack robustness and fail to provide guarantees of convergence.[22] For the purpose of remedying the above-mentioned shortcomings of data-driven methods, the physics-driven/informed methods are proposed recently.[23,24] By providing physics information, NNs are

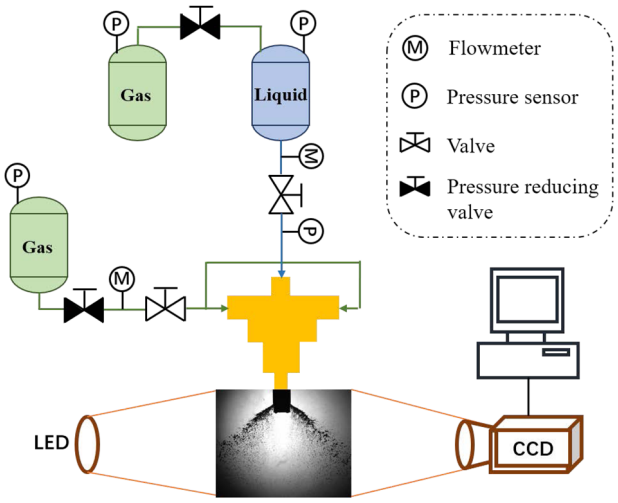



**FIG. 1.** Schematic of the experimental facilities. The test bench is composed of a gas–liquid pintle injector, a propellant feed system, and a control system. The spray visualization system includes a LED lamp and a high-speed camera.

able to directly obtain field solutions that obey physical laws and operating conditions.[25] In these works, *Partial Differential Equations* (PDEs) were employed in the loss function to explicitly constrain the network training.[26,27]

In the state-of-the-art neural network methods, *Generative Adversarial Networks* (GANs) proposed by Goodfellow *et al.*[28] are efficient to generate the instantaneous flow fields.[29,30] Despite the impressive performance for unsupervised learning tasks, the quality of generated solutions by GANs is still limited for some realistic tasks.[31] In addition, as shown in the training results later, the transient nature of the spray injection and liquid sheet break results in the extreme difficulty of usual networks to qualify the place and intensity of dominating characterizations.

In this paper, based on one specific type of GAN and the idea of physics constraint, a novel *Generative Adversarial Networks with Physical Evaluators* (GAN-PE) framework is proposed. By introducing mass conversation and spray angle models as the two evaluators, this framework has a better training convergence and predictive accuracy. The trained model is able to simulate the macroscopic morphology and characterization of the instantaneous flow fields under different conditions. This paper is organized as follows: We first introduce the experimental settings and dataset acquisition. Second, the architecture of GAN-PE and the detailed parts are described. Then, the learning results of numerical experiments are presented for validation. Finally, conclusions are drawn.

## II. DATASET FROM EXPERIMENTS

Our training data are extracted from the spray experimental results of the pintle injectors.

### A. Experimental facilities

The non-reactive cold experiments were conducted at atmospheric pressure. The dry air is used for axial flows and filtered water for radial flows. The schematic of experimental facilities is shown in Fig. 1. A back-lighting photography technique is used for instantaneous spray image visualization. The image acquisition system consists of a *light-emitting diode* (LED) light source, a high-speed camera, and a computer. The exposure time is 10 $\mu$s, and the frame rate is 50k fps.

The detailed gas–liquid pintle injector is shown in Fig. 2. In order to study the influence of the momentum ratio on the spray angle, the experimental device is designed to use the replaceable parts. In the experiment, the height of the radial liquid jet outlet $L_{\text{open}}$ and the thickness of the axial gas sheet $T_{\text{gs}}$ are adjusted by changing the height of the sleeve and the axial gap distance, respectively. When the liquid propellant is injected radially from the two sides of the pintle end through the manifold, the liquid columns are formed. These columns are broken by the axial gas propellant injection from the gap cling to the pintle. Finally, due to impingement and collision, the liquid columns break and form a plane conical spray like a hollow-cone atomizer. This design induces vigorous mixing of the gaseous and liquid propellants, which yields a high combustion efficiency.[32]

### B. Dataset acquisition

The spray experiments are carried out with the throttling level $L_{\text{t}}$ of 40%–80%. $L_{\text{t}}$ is varied by the linear adjustment of the height of the radial liquid jet outlet and the thickness of the axial gas sheet. The radial liquid jet outlet heights at throttling levels of 40%, 60%,

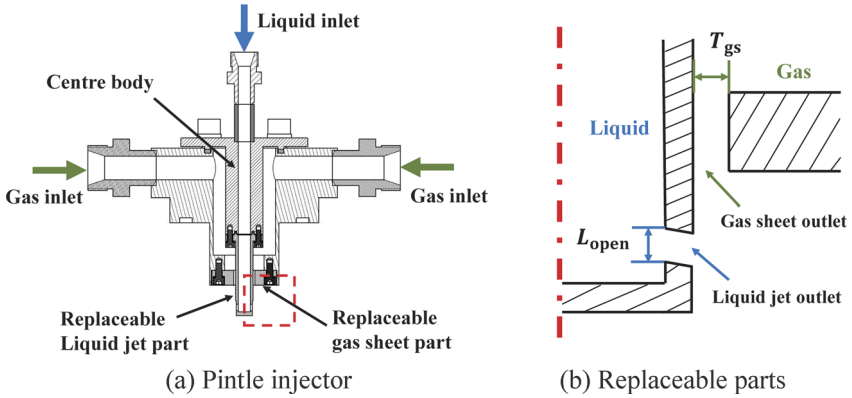



**FIG. 2.** Gas–liquid pintle injector. (a) Pintle injector within manifolds. The experimental injector consists of a gas manifold, a replaceable liquid manifold, a replaceable axial gas sheet adjustment annular, a central cylinder, and a sleeve. (b) Schematic of the replaceable parts in the red square of (a). In order to facilitate the optical observation about the spray angle, two symmetrical radial liquid jet orifices are designed on the replaceable central cylinder. $L_{\text{open}}$ and $T_{\text{gs}}$ are the injector opening distance and gas sheet thickness, respectively.

**TABLE I.** Experimental operating conditions. $m_g$ and $m_l$ are the mass flow rate of gaseous and liquid propellants, respectively. $C_{TMR}$ is the momentum ratio of the two propellants.

| $L_t$ (%) | $L_{open}$ (mm) | $T_{gs}$ (mm) | $m_g$ (g/s) | $m_l$ (g/s) | $C_{TMR}$ |
|---|---|---|---|---|---|
| 80 | 4.0 | 4.0 | 22.17 | 18.55–40.54 | 1.01–4.92 |
| 60 | 3.0 | 3.0 | 15.70 | 14.85–30.46 | 1.22–5.14 |
| 40 | 2.0 | 2.0 | 9.85 | 8.85–20.45 | 0.98–5.21 |

and 80% are 2, 3, and 4, respectively. When $L_t$ is fixed, the height of the radial liquid jet outlet is fixed and equal to the thickness of the axial gas sheet. Table I shows the operating conditions of the experimental campaign and the corresponding key specifications of the pintle injector. The temperature of the liquid and gas is 298.15 K. The mass flow rate $m_l$ is determined by the variation of liquid pressure upstream. Eventually, 35k raw images were captured and 29k of them were used for training and the others for validation.

As shown in Fig. 3, to measure the spray angle, the spray images obtained in the experiment are post-processed to clarify the spray boundary. The average of ten images with the same time interval is used to measure the spray angle manually. Every operation condition has 1k raw images, so the 100th, 200th, . . ., 900th, and 1000th images are averaged. Then, the spray angles of the time-averaged spray images, defined as $\theta = \frac{1}{2}(\theta_1 + \theta_2)$, are manually measured. Since the raw images are all captured in the steady injection stage and no temporal fluctuations, the measured angle value is unique per operating condition. Note that the average images and the corresponding manually measured angles are only used to train the spray angle estimator, i.e., the down-sampling CNN in the spray angle evaluator. The raw images are used in the training of GAN-PE.

The resolutions of the instantaneous spray image are 640 × 480. In order to reduce the training cost, the images are interpolated to the images with a resolution of 128 × 128. While the measured angle values, which represented the nature of the spray phenomenon, are fixed despite the image scaling.

## III. METHODOLOGY

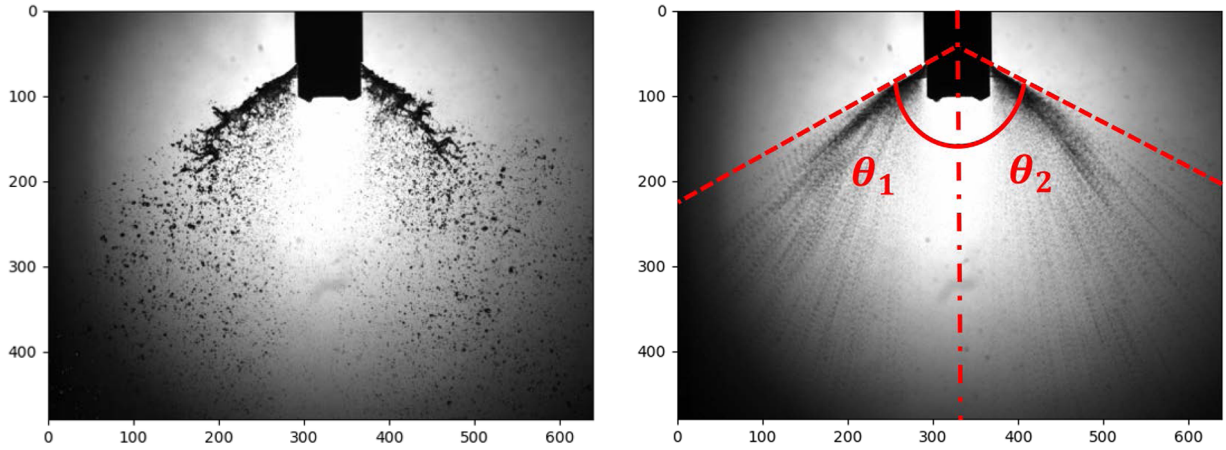
### A. Overview

Here, a *Generative Adversarial Networks with Physical Evaluators* (GAN-PE) framework is proposed. As shown in Fig. 4, the GAN-PE is composed of four parts: *generator* (*G*), *discriminator* (*D*), and two physical evaluators. The field solutions are generated by *G*, and the other three parts are employed to guarantee that the outputs catch the spray morphology and obey the operating conditions. The GAN is the base of the proposed network framework; the *G* captures the real spray data distribution, which corresponds to the operation conditions; and the *D* estimates the probability that a condition–sample pair came from the training data rather than *G*. There are also two evaluators designed to improve the performance of GANs. The first evaluator, *Mass Conservation Evaluator* ($E_{MC}$), is used to improve the generation robustness by calculating the ring error between output and the corresponding average target. The second evaluator, *spray angle evaluator* ($E_{SA}$), is used to improve the predictive accuracy in the specific operating conditions by comparing the output angle with the theoretical one. Fed with the outputs from *G*, the losses of *D*, $E_{MC}$, and $E_{SA}$ are calculated, respectively. After that, backpropagation is applied to adjust the U-net *Convolutional Neural Network* (CNN) of *G* to generate a new spray field that more satisfies the conditions and prior physics knowledge. After enough iterations, the network will be able to generate a "correct" spray field.

### B. GAN

#### 1. Generator

From inputs toward outputs, the network of *G* consists of two parts: encoding and decoding.[33] In the encoding process, the




(a) Raw image (b) Average image

**FIG. 3.** Data acquisition. The resolution of the images is 640 × 480. The raw monochrome pictures are processed as 8-bit gray images in which every pixel has a gray value and the range is 0–255. The average is obtained by calculating the mean gray value of raw images. The images are regarded as the 2D matrices whose dimensions are 640 × 480.

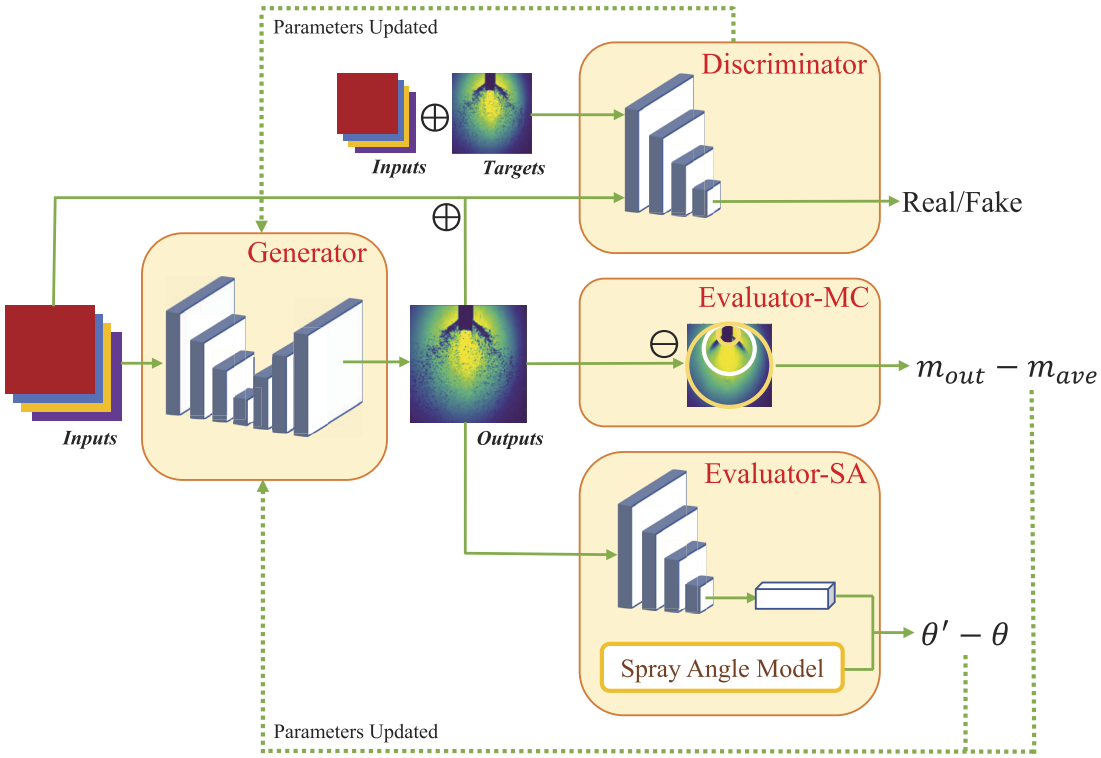



**FIG. 4.** Schematic of the proposed network framework. Fed with the operating conditions, the U-net generator outputs a field solution of the spray. The discriminator, mass conservation, and spray angle losses are utilized to update the generator by backpropagation.

operating conditions $L_{\text{open}}$, $T_{\text{gs}}$, $m_{\text{g}}$, and $m_{\text{l}}$ are resized as four feature channels of the input tensor for convolutional down-sampling with corresponding kernels.[34,35] After that, the matrices with a size of 128 × 128 are progressively reduced to 512 single-value vectors. Each layer of the network consists of a convolution operation, batch normalization, and a non-linear activation function. By the convolutional calculation, along with the increasing number of feature channels, the matrix size is down-sampled by a factor of 2. In this way, the information of operating conditions is translated into the extracted features in the next layer. In addition, skip-concatenations from input to output feature channels are introduced to ensure that operating condition information is available in the following up-sampling process for inferring the solution. Then, the decoding part works in an opposite way, which can be regarded as an inverse convolutional process mirroring the behavior of the encoding part. Along with the increase in spatial resolution, the spray fields are reconstructed based on the single-value vectors by up-sampling operations. For more details of the U-net architecture and convolutional block, including active function, pooling, and dropout, see Ref. 27.

The weighted loss function considering the following discriminator and evaluators is written as

$$\mathscr{L}(D, E_{\text{MC}}, E_{\text{SA}}) = \mathscr{L}_D + \alpha \mathscr{L}_{E_{\text{MC}}} + \beta \mathscr{L}_{E_{\text{SA}}}, \quad (1)$$

where $\mathscr{L}_D$, $\mathscr{L}_{E_{\text{MC}}}$, and $\mathscr{L}_{E_{\text{SA}}}$ are the loss terms that are calculated by $D$, $E_{\text{MC}}$, and $E_{\text{SA}}$, respectively. In addition, α and β are the constant hyperparameters that are manually tuned before training to adapt the scales of these loss terms. Here, the orders of magnitude of α and β are 2 and 1, respectively. After proper training, both generator and discriminator losses remain stable and the generator is able to map a spray sample from a random distribution to the desired one that obeys the physical knowledge and conditions.

### 2. Discriminator

The discriminator in a GAN is simply a classifier. It tries to distinguish real samples from the data created by the generator, i.e., fake data. The discriminator's training data come from two sources. One consists of the real data instances, here are the real experimental images. $D$ uses these instances as positive examples during training. The other are the fake data instances created by the generator. The discriminator uses these instances as negative examples during training. Then, $D$ is used to feed the possibility that samples come from the targets rather than generation distribution back to $G$. We use *Least Squares Generative Adversarial Networks* (LSGANs) settings to train the $D$ and the $G$ simultaneously.[36] This special type of GAN helps to remedy the gradient vanishing by using the least squares loss function instead of the sigmoid cross entropy loss function.[37]

Here, $D$ is modified by the encoder of the $G$, which means that the generating solutions from $G$ are down-sampled by the re-convolutional calculation so that the spray field information is concluded into the linear 1D tensor. Then, this 1D tensor will be used to

be trained to maximize the probability of assigning the correct label, real or fake, to both training targets and generating solutions. Similar to the work in Ref. 38, we use the input–output and input–target pairs to feed $D$ instead of only output and target in the random image generation tasks. The operating conditions and the outputs/targets are concatenated as the different feature channels in a 4D data tensor. In this way, $D$ not only discriminates between the real and fake but also helps to judge whether the outputs accord with the corresponding conditions.

The loss functions for LSGANs are defined as

$$\begin{aligned}\min_D V_{\mathrm{GAN}}(D) &= \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[(D(\boldsymbol{x})-b)^2] \\ &\quad + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[(D(G(\boldsymbol{z}))-a)^2], \\ \min_G V_{\mathrm{GAN}}(G) &= \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[(D(G(\boldsymbol{z}))-c)^2],\end{aligned} \tag{2}$$

where $\boldsymbol{x}$ is the training data and $\boldsymbol{z}$ is the input variables. In addition, $a$ and $b$ are the labels for fake data and real data, respectively; and $c$ denotes the value that $G$ wants $D$ to believe for the generated solutions. Here, we apply $a = 0$ and $b = c = 1$. Therefore, $\mathscr{L}_D$ is equal to the second part of Eq. (2).

## C. Evaluators

### 1. Mass conservation evaluator

As shown in Fig. 5, we assume that there are a few spherical volumes with different diameters that are tangent at the middle point of the upper boundary in both generating images and the average images. The idea originates from that the spray phenomenon obeys the mass conservation law. Because the propellants are injected from the two flanks of the central pintle and the width of the image's normal direction is fixed, the 3D effect is ignored and the spheres are simplified to the 2D rings. The mass fluxes of droplets through one specific ring in every instantaneous frame are equivalent. The spray field generation is indeed a 2D image reconstruction task. Therefore, the "mass" here is a broader concept, which also involves the background shadow, i.e., the quasi-mass in one position is represented by the gray value of the corresponding pixel. Following the definition of "L1loss" that is widely used in the machine learning community, we define a mass conservation loss here. The difference is that the former measures the sum of absolute errors between each element in the generation and target,[39] but ours first calculates a gray value summation of every element in one concerning ring and then compares absolute error between the corresponding rings in the generation and target and, at last, the summation of the errors.

The mass conservation error, i.e., the loss term from $E_{\mathrm{MC}}$, is defined as

$$\mathscr{L}_{E_{\mathrm{MC}}} = \begin{cases} \sum_{k=1}^{m}\left|\sum_{i=1}^{n} x_i - \sum_{j=1}^{n} y_j\right|_k, & \mathscr{E} \geq \mathscr{E}_{\mathrm{thr}}, \\ 0, & \mathscr{E} < \mathscr{E}_{\mathrm{thr}}, \end{cases} \tag{3}$$

where $x$ and $y$ are the gray values in generated images and average targets, respectively. In addition, $m$ is the number of concerning rings and $n$ is the number of data points in one concerning ring in the matrix. $\mathscr{E}$ is the error between the output and average matrix and defined as

$$\mathscr{E} = \sum_{i=1}^{N}|x_i - y_i|, \tag{4}$$

where $N$ is the number of all the data points in the matrix. To improve the generation randomness and in view of the error caused

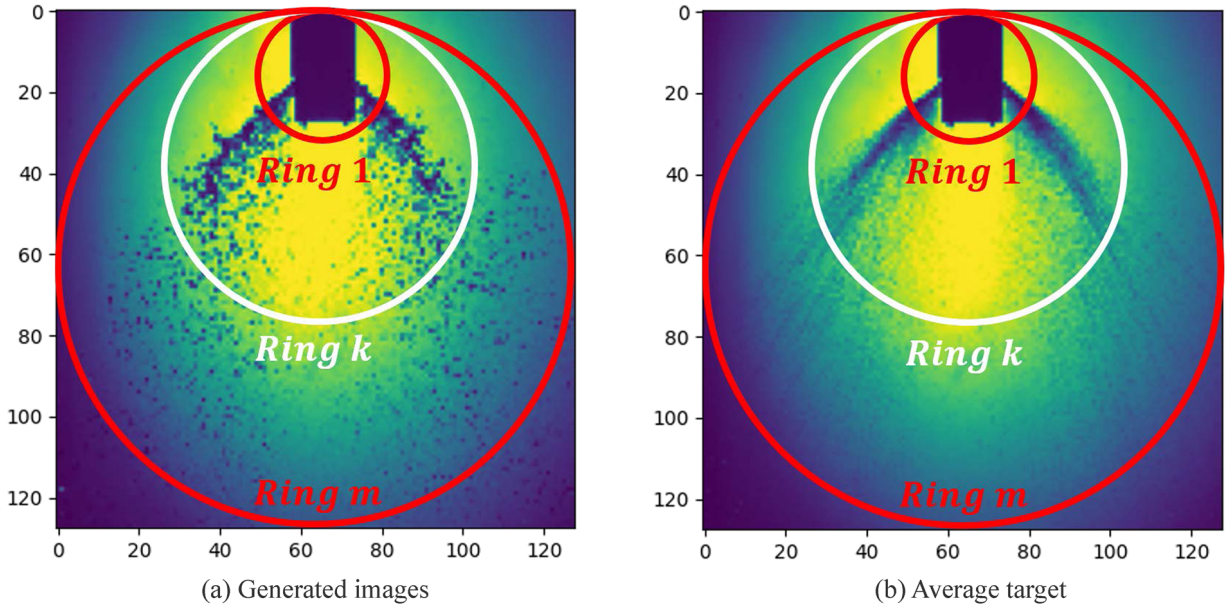



**FIG. 5.** Mass conservation. The resolution of the images is 128 × 128. The rings remain tangent at the central point of the top edge.

by light transmission and reflection, the loss threshold $\mathscr{E}_{thr}$ is introduced herein. Once the loss is less than the threshold, this loss term will be ignored.

### 2. Spray angle evaluator

The present evaluator is composed of two parts: one is the theoretical model of the spray angle and the other is a CNN encoder to estimate the spray angles from the generated field solutions.

The schematic diagram of the theoretical model of the spray angle is shown in Fig. 6. Several basic hypotheses must be declared before carrying out the theoretical analysis: (a) An element of fluid emerging from the jet exit is assumed to have the constant length and width equal to the jet exit length $L_{open}$ and width $W$, as it moves along the trajectory; (b) liquid jet deformation, evaporation, and droplet dispersion are ignored; (c) the fluid element has a constant y-component velocity and an initial angle equal to the central propellant deflection angle $\varphi$; (d) the spray angle is assumed to be equal to the slope of the liquid jet at the position where it passes through the gas film; and (e) surface tension, gravity, friction, heat transfer, and phase change are ignored.

Following the definition in Ref. 40, the aerodynamic drag $F_d$ of the liquid element in the x-direction is

$$F_d = \frac{1}{2} C_d \rho_g (v_g - v_l \cos\varphi)^2 WH, \tag{5}$$

where $C_d$ is the drag coefficient. $v_g$ and $v_l$ are the gas velocity and initial liquid element velocity, respectively. According to Newton's second law in the x-direction, there is

$$\frac{1}{2} C_d \rho_g (v_g - v_l \cos\varphi)^2 WH = m_l a = \rho_l W L_{open} H \frac{dv_x}{dt}, \tag{6}$$

where $m_l$ and $a$ are the mass and acceleration of the liquid element, respectively. $v_x$ is the x-component of the liquid element velocity. It

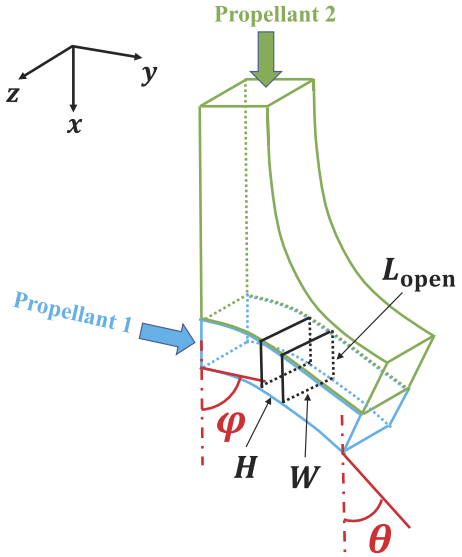



**FIG. 6.** Schematic of the spray angle model. $L_{open}$ and $W$ are the length and width of the radical rectangular section, respectively. $H$ is the axial height of the liquid element.

can be rewritten as

$$\frac{dv_x}{dt} = \frac{C_d}{2} \frac{\rho_g}{\rho_l} \frac{(v_g - v_l \cos\varphi)^2}{L_{open}}. \tag{7}$$

Then, the integration of Eq. (7) with respect to time is

$$\int_0^t \frac{dv_x}{dt} dt = \int_0^t \frac{C_d}{2} \frac{\rho_g}{\rho_l} \frac{(v_g - v_l \cos\varphi)^2}{L_{open}} dt, \tag{8}$$

and then,

$$v_x|_0^t = \frac{C_d}{2} \frac{\rho_g}{\rho_l} \frac{(v_g - v_l \cos\varphi)^2}{L_{open}} t. \tag{9}$$

Since $v_x(0) = v_l \cos\varphi$, there is

$$v_x = \frac{C_d}{2} \frac{\rho_g}{\rho_l} \frac{(v_g - v_l \cos\varphi)^2}{L_{open}} t + v_l \cos\varphi. \tag{10}$$

As $v_x = dx/dt$, the integration with respect to time is

$$x = \frac{C_d}{4} \frac{\rho_g}{\rho_l} \frac{(v_g - v_l \cos\varphi)^2}{L_{open}} t^2 + v_l \cos\varphi t, \tag{11}$$

where $x$ is the x-coordinate of the element's trajectory. Since the y-component velocity remains constant and $v_y = v_l \sin\varphi = y/t$, where $y$ is the y-coordinate, the mathematical expression of the element trajectory is derived as

$$x = \frac{C_d}{4} \frac{\rho_g}{\rho_l} \frac{(v_g - v_l \cos\varphi)^2}{L_{open}} \left(\frac{y}{v_l \sin\varphi}\right)^2 + \frac{v_l \cos\varphi y}{v_l \sin\varphi}. \tag{12}$$

For the collision between the gas sheet and the rectangular liquid jet, the momentum ratio is

$$C_{TMR} = \frac{\dot{m}_l v_l}{\dot{m}_g v_g} = \frac{\rho_l v_l^2 A_l}{\rho_g v_g^2 A_g} = \frac{\rho_l v_l^2 W L_{open}}{\rho_g v_g^2 WH} = \frac{\rho_l v_l^2 L_{open}}{\rho_g v_g^2 H}. \tag{13}$$

Therefore, Eq. (12) is expressed in terms of the momentum ratio $C_{TMR}$ as follows:

$$x = \frac{C_d}{4 C_{TMR} H \sin^2\varphi} \left(1 - \frac{v_l \cos\varphi}{v_g}\right)^2 y^2 + \frac{\cos\varphi}{\sin\varphi} y. \tag{14}$$

At the position of penetrating through the gas sheet where $y = H$, the trajectory expression is modified to

$$x = \left[\frac{C_d}{4 C_{TMR} \sin^2\varphi} \left(1 - \frac{v_l \cos\varphi}{v_g}\right)^2 + \frac{\cos\varphi}{\sin\varphi}\right] y, \tag{15}$$

and the slope of the liquid jet $\theta$ is obtained as

$$\theta = 90^\circ - \arctan\left[\frac{C_d}{4 C_{TMR} \sin^2\varphi} \left(1 - \frac{v_l \cos\varphi}{v_g}\right)^2 + \frac{\cos\varphi}{\sin\varphi}\right]. \tag{16}$$

The theoretical model assumes that the liquid jet does not deform, but in reality, it will deform under aerodynamic forces, which results in a reduction of the effective momentum of the

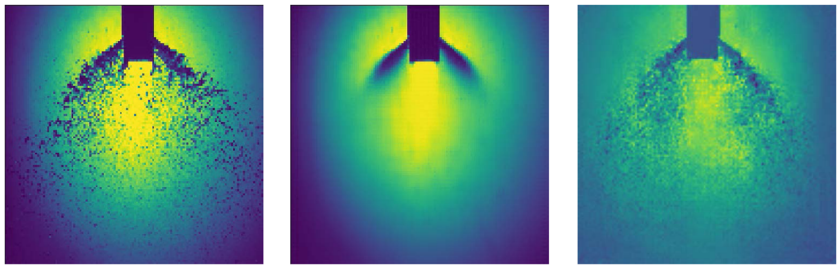



**FIG. 7.** Comparison of the CNN generator and original GAN. The operating condition is $L_{open} = T_{gs} = 4$ mm, $m_l = 35.81$ g/s, and $m_g = 22.17$ g/s.

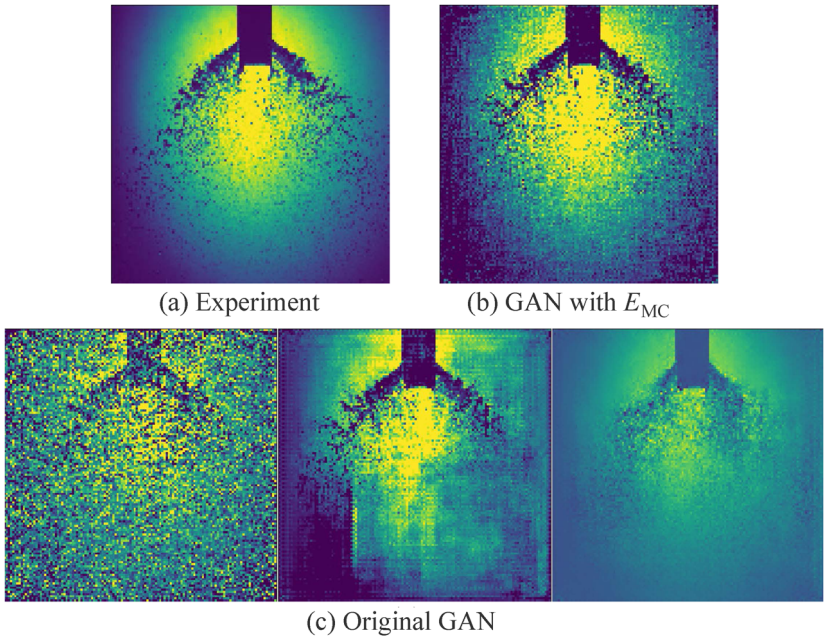



**FIG. 8.** The comparison between the results of with or without $E_{MC}$. Compared with the results without $E_{MC}$, GAN with $E_{MC}$ has a clearer and more regularizing reconstruction of the spray field.

**TABLE II.** Spray angle estimation by the manual measurement and CNN.

| | $L_t$ = 80% | | | $L_t$ = 60% | | | $L_t$ = 40% | | |
|---|---|---|---|---|---|---|---|---|---|
| $C_{TMR}$ | MM | CNN | Error (%) | MM | CNN | Error (%) | MM | CNN | Error (%) |
| 0.52–0.56 | 31.27 | 29.70 | 5.0 | 29.31 | 28.04 | 4.3 | 26.05 | 27.57 | 5.8 |
| 0.79–0.86 | 36.32 | 35.10 | 3.4 | 33.12 | 34.54 | 4.3 | 31.59 | 32.61 | 3.2 |
| 0.98–1.22 | 40.67 | 41.98 | 3.2 | 34.89 | 35.60 | 2.0 | 34.67 | 35.15 | 1.4 |
| 1.29–1.34 | 41.98 | 40.37 | −3.8 | 39.21 | 39.34 | 0.3 | 36.36 | 37.0 | 1.75 |
| 1.50–1.70 | 44.37 | 45.99 | 3.7 | 42.87 | 43.37 | 1.2 | 39.43 | 38.08 | −3.4 |
| 1.98–2.04 | 45.28 | 47.45 | 4.8 | 43.43 | 44.62 | 2.7 | 41.01 | 41.47 | 1.1 |
| 2.56–2.69 | 49.88 | 49.43 | −0.9 | 47.34 | 45.46 | −4.0 | 44.64 | 44.39 | −0.6 |
| 2.90–3.22 | 50.36 | 50.02 | −0.7 | 48.13 | 47.07 | −2.2 | 44.98 | 45.64 | 1.5 |
| 3.33–3.39 | 51.32 | 52.92 | 3.1 | 48.12 | 48.75 | 1.3 | 46.01 | 46.64 | 1.4 |
| 3.83–4.12 | 52.18 | 52.55 | 0.7 | 48.84 | 51.09 | 4.6 | 47.64 | 48.98 | 2.8 |
| 4.50–4.88 | 54.39 | 53.92 | −0.9 | 49.34 | 50.57 | 2.5 | 48.80 | 49.60 | 1.7 |
| 5.12 | 54.55 | 55.96 | 2.6 | 49.41 | 50.75 | 2.7 | ··· | ··· | ··· |

liquid jet. Consequently, the liquid jet deformation factor $\gamma$, which is obtained through the experimental results, is introduced to modify the spray angle theoretical model. In this way, Eq. (16) is modified to

$$\theta = \gamma\left\{90^\circ - \arctan\left[\frac{C_d}{4C_{TMR}\sin^2\varphi}\left(1 - \frac{v_l\cos\varphi}{v_g}\right)^2 + \frac{\cos\varphi}{\sin\varphi}\right]\right\}. \quad (17)$$

Note that the spray angle model derived above has been demonstrated by the experimental results. In fact, when there is no central propellant deflection and $\varphi = 90^\circ$, Eq. (17) could be simplified as $\theta = \gamma[90^\circ - \arctan(C_d/4C_{TMR})]$. It formally corresponds to the experimental fitting model $\theta = C_1 \arctan(C_2 C_{TMR})$ of the liquid–liquid pintle injector in Ref. 41, where $C_1$ and $C_2$ are the fitting parameters.

In the field of medical image analysis, the machine learning approach, especially the deep neural networks, has been employed for automated scoliosis assessment.[42–44] In these publications, the x-ray images are fed into the neural network estimator and the spinal Cobb angles are obtained. Similarly, inside the $E_{SA}$, there is a well-trained spray angle estimator to output the angle values from the predictive images. The architecture of this down-sampling CNN is like $D$, except the addition of one fully connected layer in the end to output the estimated spray angle $\theta'$.

The loss term from $E_{SA}$ is calculated as

$$\mathscr{L}_{E_{SA}} = |\theta' - \theta|. \quad (18)$$

## IV. RESULTS

### A. Model validation

Figure 7 shows the generated results in one typical operating condition of the CNN generator and original GAN, which consist of only $G$ and $D$. The L1 loss used by the CNN generator compares the difference between the generations and targets. This absolute error loss performs very well in some steady or mean state field prediction tasks, such as the work in Refs. 21 and 27. However, when the training cases have a multi-modal distribution, this loss will fail. In our spray field prediction task, although the morphology under one specific operating condition is similar, the detailed droplets are distinguishable. Therefore, the instantaneous spray field solution has many possibilities, which are like the various frames at different times. However, the L1 loss averages all the possibilities and produces a very blurry average image instead. However, the discriminator in GANs, which can be regarded as the loss of generator, is not an explicit loss function. Instead of the pixel-wise loss, $D$ is an approximation loss, which discriminates between the real and fake data distributions and guides the spray generation with detailed morphology.

However, in the training process of GANs, the generator and discriminator have to be balanced trained and the convergence is often an unstable state. The discriminator and generator are always in a seesaw battle to undercut each other. Similar to the work in Ref. 45, we maintain a dynamic ratio between the number of gradient descent iterations on the discriminator and the generator using *Exponential Moving Average* (EMA).

For the spray simulation task, the discriminator has difficulties capturing the detailed feature of the small droplets. The $\mathscr{L}_D$ has a possibility of becoming less meaningful through the training process. $G$ will update itself based on the random feedback and the quality of generation may collapse. The $G$ outputs low-quality images through many epochs, and some of them show faint spray patterns in the background but are easily identified as fake. It will be very easy for the discriminator to distinguish the targets and generation so the values of the loss from $D$ drop to zero rapidly.

The comparison of generated spray images from different frameworks demonstrates the superior performance of the mass conservation evaluator as shown in Fig. 8. In some generated images,

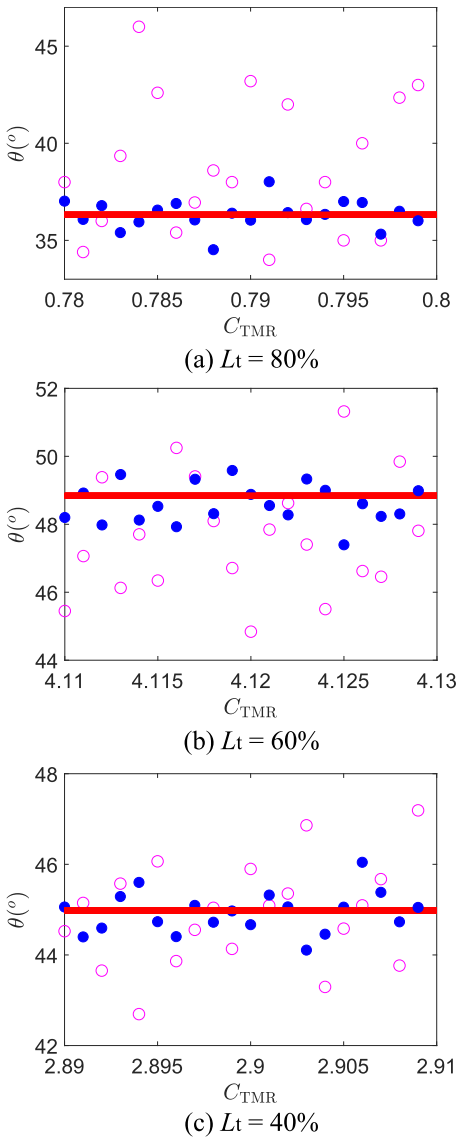



**FIG. 9.** Comparison of the predicted spray angle between the model with and without $E_{SA}$. The red lines indicate the experimental results. The blue point markers indicate the angle values predicted by the model with $E_{SA}$. The magenta circle markers indicate the ones predicted by the model without $E_{SA}$.

the background does not agree with the real target; the introduction of $\mathscr{L}_{E_{MC}}$ helps $G$ identify the position and intensity of the droplet as well as the background shadow.

In the machine learning field, the parallel training usually is conducted with a batch form. Inputting a certain number of samples to the neural networks in a batch, the solutions are parallel generated and compared to the targets and an error is calculated. In our framework, when obtaining the predicted spray field, all the outputs, ten generated images, in one training batch will be averaged to one image. Then, this average image will be used to feed the down-sampling CNN to obtain the spray angle of this batch. To validate the spray angle estimator, we use the average images from experiments as test samples to output the angle values. Table II shows the comparison of spray angles obtained by *Manual Measurements* (MM) and CNN estimator. With the increase of $C_{TMR}$, the deviations between the two tend to be smaller. The error of all the test cases is less than 5.8%.

After this CNN estimator is trained, it only takes some milliseconds to output an angle value, which is according to the operating conditions of this batch. Then, the estimated value is employed in $\mathscr{L}_{E_{SA}}$ to update $G$. Due to the quick decrease at the beginning of the training, this error no longer affects $G$, only except for the abnormal generated solutions with angles diverged awfully from the theoretical model.

As shown in Fig. 9, the results predicted by the model without $E_{SA}$ have large value ranges. The difference between maximum and minimum angles in a small momentum ratio level, $C_{TMR} \in [0.78, 0.80]$, even approach 12°. Meantime, the predicted angles have an "even-bias." Compared with the experimental measurements, all values are closer to the mean value of the whole domain. For small momentum ratios, as shown in Fig. 9(a), most of the predicted angle values are larger than the experimental results. For large momentum ratios, as shown in Fig. 9(b), most of the values are smaller than the experimental results. With the help of $E_{SA}$, this phenomenon is reduced and all three groups with different momentum ratios have the predicted angles closer to the experimental results. In addition, the predicted values of spray angles have narrower ranges.

## B. Predictions

The literature showed that the macroscopic morphology study is important to characterize a spray.[46,47] Here, the simulated spray morphology is analyzed and compared with the experimental results.

Figure 10 compares the simulated and experimental spray morphology under different operating conditions. The typical spray morphology, i.e., liquid column formation, breakup of the column, and lateral expansion of the spray, can be clearly noted. The

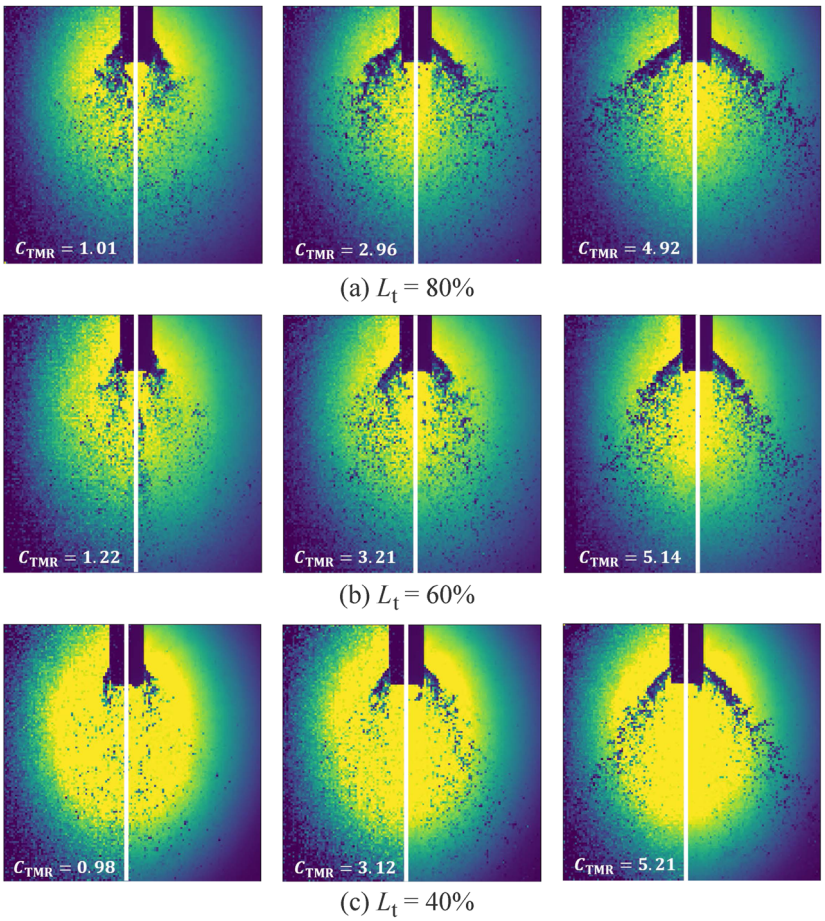



**FIG. 10.** Simulated and experimental spray morphology under different throttling levels and momentum ratios. The left half part is the simulated spray obtained by GAN-PE, and the right half part is the corresponding experimental high-speed image with the same size scale.

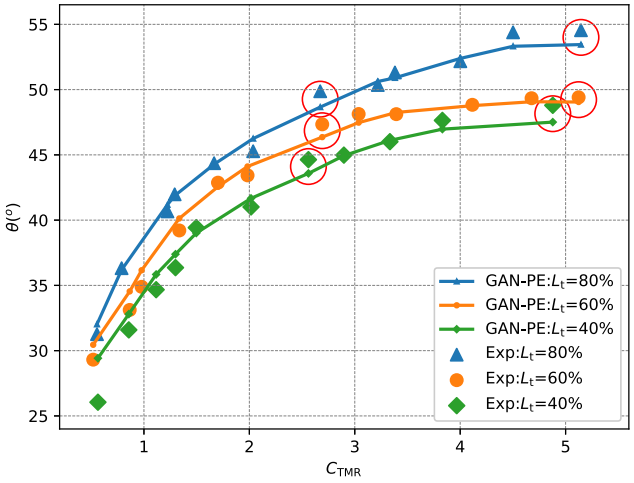



**FIG. 11.** The comparison of the spray angle between GAN-PE and experiments. The red circles show the cases that are out of the learning domain.

generated spray field shows that the droplets experience a size reduction before approaching a uniform tiny size distribution because of the impingement and collision. As shown in Fig. 10, the generated spray is fairly comparable with the experimental results; meanwhile, the hollow-cone-shaped profiles with a specific spray angle are well reproduced. For those cases that are out of the training domain, the last column in the figure, the generation also presents a similar good quality compared with those insides. Imperfectly, the background still represents grainy and the values of adjacent data points are not as continuous as those in the real images. However, all in all, the simulation succeeds to report the macroscopical morphology of the spray.

Figure 11 shows the curves of the spray angle vs momentum ratio at different throttling levels; it can be observed that the GAN-PE results coincide with the experimental results well in a wide momentum ratio range. According to theoretical analysis that has been explained in Sec. III C, the spray angle is mainly determined by the momentum ratio, and the simulated solutions also represent this. Due to the difficulties of $E_{SA}$ to estimate the small angles, the predictions of the cases with small momentum ratios have relatively large errors. The values of different throttling levels are close to each other. For the test cases that are not in the learning domain, the results have a small deviation and all the predicted angle values are less than the manually measured ones. It is because these test cases are not constrained by the targets so the prediction has a trend to approach the mean value of the adjacent operating points, which is happened to be larger.

## V. CONCLUSION

In this paper, we proposed a novel deep learning framework constrained by physical evaluators to directly predict spray solutions based on generative adversarial networks. The normal discriminator and the mass conservation and spray angle evaluators are used to constrain the CNN to generate the spray solution, including macroscopical morphology and spray angle. The former evaluator is able to improve the training convergence and the latter one helps to obtain more accurate spray angles that are consistent with the operating conditions. It is noteworthy that the related network architecture and spray problem are generic and the proposed framework is potentially suitable for other fluid field simulations that have proper prior physics knowledge. Further research will be carried out for spray droplet size analysis and prediction with the present network framework.

## ACKNOWLEDGMENTS

This article was supported by the TUM Open Access Publishing Fund. Hao Ma was supported by the China Scholarship Council (Grant No. 201703170250). Chi Zhang was supported by Deutsche Forschungsgemeinschaft (Grant Nos. DFG HU1527/6-1 and DFG HU1527/10-1). The authors thank Xiangyu Hu, Nils Thuerey, Yujie Zhu, and Wei Wang for beneficial discussions and Anlong Yang for the experimental facilities.

## DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

## REFERENCES

[1] G. Dressler and J. Bauer, "TRW pintle engine heritage and performance characteristics," in *36th AiAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit* (AIAA, 2000), p. 3871.
[2] S. Heister, "Pintle injectors," in *Handbook of Atomization and Sprays* (Springer, 2011), pp. 647–655.
[3] K. Sakaki, H. Kakudo, S. Nakaya, M. Tsue, R. Kanai, K. Suzuki, T. Inagawa, and T. Hiraiwa, "Performance evaluation of rocket engine combustors using ethanol/liquid oxygen pintle injector," in *52nd AIAA/SAE/ASEE Joint Propulsion Conference* (AIAA, 2016), p. 5080.
[4] W. R. Hammock, Jr., E. C. Currie, and A. E. Fisher, "Apollo experience report: Descent propulsion system," NASA Technical Note D-7143 (1973).
[5] B. Bjelde, P. Capozzoli, and G. Shotwell, "The SpaceX Falcon 1 launch vehicle flight 3 results, future developments, and Falcon 9 evolution," in *59th International Astronautical Congress* (Space Exploration Technologies, 2008).
[6] M. J. Casiano, J. R. Hulka, and V. Yang, "Liquid-propellant rocket engine throttling: A comprehensive review," J. Propul. Power **26**, 897–923 (2010).
[7] K. Radhakrishnan, M. Son, K. Lee, and J. Koo, "Lagrangian approach to axisymmetric spray simulation of pintle injector for liquid rocket engines," Atomization Sprays **28**, 443–458 (2018).
[8] M. Son, K. Yu, K. Radhakrishnan, B. Shin, and J. Koo, "Verification on spray simulation of a pintle injector for liquid rocket engine," J. Therm. Sci. **25**, 90–96 (2016).
[9] M. Son, K. Radhakrishnan, Y. Yoon, and J. Koo, "Numerical study on the combustion characteristics of a fuel-centered pintle injector for methane rocket engines," Acta Astronaut. **135**, 139–149 (2017).
[10] J. Yu and J. S. Hesthaven, "Flowfield reconstruction method using artificial neural network," AIAA J. **57**, 482–498 (2019).
[11] H. Ma, Y.-x. Zhang, O. J. Haidn, N. Thuerey, and X.-y. Hu, "Supervised learning mixing characteristics of film cooling in a rocket combustor using convolutional neural networks," Acta Astronaut. **175**, 11–18 (2020).
[12] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, "Machine learning for fluid mechanics," Annu. Rev. Fluid Mech. **52**, 477 (2019); arXiv:1905.11075v3.
[13] J. Ling, A. Kurzawski, and J. Templeton, "Reynolds averaged turbulence modelling using deep neural networks with embedded invariance," J. Fluid Mech. **807**, 155–166 (2016).
[14] E. J. Parish and K. Duraisamy, "A paradigm for data-driven predictive modeling using field inversion and machine learning," J. Comput. Phys. **305**, 758–774 (2016).

[15]K. Duraisamy, G. Iaccarino, and H. Xiao, "Turbulence modeling in the age of data," Annu. Rev. Fluid Mech. **51**, 357–377 (2019); arXiv:1804.00183v3.
[16]X. Jin, P. Cheng, W.-L. Chen, and H. Li, "Prediction model of velocity field around circular cylinder over various Reynolds numbers by fusion convolutional neural networks based on pressure on the cylinder," Phys. Fluids **30**, 047105 (2018).
[17]V. Sekar, Q. Jiang, C. Shu, and B. C. Khoo, "Fast flow field prediction over airfoils using deep learning approach," Phys. Fluids **31**, 057103 (2019).
[18]N. Omata and S. Shirayama, "A novel method of low-dimensional representation for temporal behavior of flow fields using deep autoencoder," AIP Adv. **9**, 015006 (2019).
[19]A. B. Farimani, J. Gomes, and V. S. Pande, "Deep learning the physics of transport phenomena," arXiv:1709.02432 (2017).
[20]M. D. Ribeiro, A. Rehman, S. Ahmed, and A. Dengel, "DeepCFD: Efficient steady-state laminar flow approximation with deep convolutional neural networks," arXiv:2004.08826 (2020).
[21] N. Thuerey, K. Weißenow, L. Prantl, and X. Hu, "Deep learning methods for Reynolds-averaged Navier–Stokes simulations of airfoil flows," AIAA J. **58**, 25–36 (2020).
[22]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," J. Comput. Phys. **378**, 686–707 (2019).
[23]M. Raissi, A. Yazdani, and G. E. Karniadakis, "Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations," Science **367**, 1026–1030 (2020).
[24]P. Holl, V. Koltun, and N. Thuerey, "Learning to control PDEs with differentiable physics," arXiv:2001.07457 (2020).
[25]L. Sun, H. Gao, S. Pan, and J.-X. Wang, "Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data," Comput. Methods Appl. Mech. Eng. **361**, 112732 (2020).
[26]L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, "DeepXDE: A deep learning library for solving differential equations," SIAM Review **63**, 208–228 (2021).
[27]H. Ma, X. Hu, Y. Zhang, N. Thuerey, and O. J. Haidn, "A combined data-driven and physics-driven method for steady heat conduction prediction using deep convolutional neural networks," arXiv:2005.08119 (2020).
[28]I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial netowrks," arXiv:1406.2661.
[29]J. Kim and C. Lee, "Deep unsupervised learning of turbulence for inflow generation at various Reynolds numbers," J. Comput. Phys. **406**, 109216 (2020).
[30]J.-L. Wu, K. Kashinath, A. Albert, D. Chirila, H. Xiao *et al.*, "Enforcing statistical constraints in generative adversarial networks for modeling chaotic dynamical systems," J. Comput. Phys. **406**, 109209 (2020).
[31] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International Conference on Machine Learning* (PMLR, 2017), pp. 214–223.
[32]K. Sakaki, H. Kakudo, S. Nakaya, M. Tsue, K. Suzuki, R. Kanai, T. Inagawa, and T. Hiraiwa, "Combustion characteristics of ethanol/liquid-oxygen rocket-engine combustor with planar pintle injector," J. Propul. Power **33**, 514–521 (2017).
[33]O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention* (Springer, 2015), pp. 234–241.
[34]Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," in *IEEE Transactions on Neural Networks and Learning Systems* (IEEE, 2021).
[35]A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," Artif. Intell. Rev. **53**, 5455–5516 (2020).
[36]X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision* (IEEE, 2017), pp. 2794–2802.
[37]T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," arXiv:1710.10196 (2017).
[38]M. Mirza and S. Osindero, "Conditional generative adversarial nets," arXiv:1411.1784 (2014).
[39]A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "PyTorch: An imperative style, high-performance deep learning library," arXiv:1912.01703.
[40]P.-K. Wu, K. A. Kirkendall, R. P. Fuller, and A. S. Nejad, "Breakup processes of liquid jets in subsonic crossflows," J. Propul. Power **13**, 64–73 (1997).
[41] J. Freeberg and J. Hogge, "Spray cone formation from pintle-type injector systems in liquid rocket engines," in *AIAA Scitech 2019 Forum* (AIAA, 2019), p. 0152.
[42]J. Yang, K. Zhang, H. Fan, Z. Huang, Y. Xiang, J. Yang, L. He, L. Zhang, Y. Yang, R. Li *et al.*, "Development and validation of deep learning algorithms for scoliosis screening using back images," Commun. Biol. **2**, 390 (2019).
[43]Y. Cai, L. Wang, M. Audette, G. Zheng, and S. Li, *Computational Methods and Clinical Applications for Spine Imaging* (Springer, 2020).
[44]H. Wu, C. Bailey, P. Rasoulinejad, and S. Li, "Automated comprehensive adolescent idiopathic scoliosis assessment using mvc-net," Med. Image Anal. **48**, 1–11 (2018).
[45]Y. Xie, E. Franz, M. Chu, and N. Thuerey, "tempoGAN: A temporally coherent, volumetric GAN for super-resolution fluid flow," ACM Trans. Graphics **37**, 1–15 (2018).
[46]M. Luo, "Experimental and numerical study of cryogenic flashing spray in spacecraft application," Ph.D. thesis, Technische Universität München, 2018.
[47]M. Luo, Y. Wu, and O. J. Haidn, "Temperature and size measurements of cryogenic spray droplets with global rainbow refractometry," J. Propul. Power **35**, 359–368 (2019).