Available online at https://link.springer.com/journal/42241
http://www.jhydrodynamics.com
Journal of Hydrodynamics, 2020, 32(2): 234-246
https://doi.org/10.1007/s42241-020-0028-y

Deep reinforcement learning in fluid mechanics: A promising method for both active flow control and shape optimization*

Jean Rabault[1], Feng Ren[2], Wei Zhang[3], Hui Tang[2], Hui Xu[4, 5]
1. *Department of Mathematics, University of Oslo, Oslo, Norway*
2. *Research Center of Fluid Structure Interactions Department of Mechanical Engineering, Hong Kong Polytechnical University, Hong Kong, China*
3. *Science and Technology on Water Jet Propulsion Laboratory, Marine Design and Research Institute of China, Shanghai 200011, China*
4. *School of Aeronautic and Astronautic, Shanghai Jiao Tong University, Shanghai 200011, China*
5. *Department of Aeronautics, Imperial College London, London, UK*

(Received January 13, 2020 , Revised January 20, 2020 , Accepted January 22 , 2020, Published online April 30, 2020)

Abstract: In recent years, artificial neural networks (ANNs) and deep learning have become increasingly popular across a wide range of scientific and technical fields, including fluid mechanics. While it will take time to fully grasp the potentialities as well as the limitations of these methods, evidence is starting to accumulate that point to their potential in helping solve problems for which no theoretically optimal solution method is known. This is particularly true in fluid mechanics, where problems involving optimal control and optimal design are involved. Indeed, such problems are famously difficult to solve effectively with traditional methods due to the combination of non linearity, non convexity, and high dimensionality they involve. By contrast, deep reinforcement learning (DRL), a method of optimization based on teaching empirical strategies to an ANN through trial and error, is well adapted to solving such problems. In this short review, we offer an insight into the current state of the art of the use of DRL within fluid mechanics, focusing on control and optimal design problems.

Key words: Machine learning, deep reinforcement learning (DRL), flow control

Introduction

The application of Machine Learning and artificial neural networks (ANNs) to classical scientific and technical fields of research is advancing rapidly. This is a consequence of the efficiency of such methods on a variety of problems, ranging from image analysis[1-3] to the optimal control of robots[4-5]. These recent successes have highlighted the ability of ANNs to deal with the ingredients of non linearity, non convexity, and high dimensionality that are still today a challenge across a wide range of scientific disciplines.

In the present paper, we aim at providing a concise review of the current state of one of the Machine Learning techniques that is rapidly developing, namely deep reinforcement learning (DRL)[6-8], focusing on its use to solve problems arising in Fluid Mechanics. Our aim here is to provide a general overview of the field, point the reader to the relevant literature for more details, and and present the main challenges that remain to be solved for further application of this method. This short review was written as a complement to the presentation of the same title held at the International Symposium on High- Fidelity Computational Methods and Applications in Shanghai, December 2019[9], and it also shares a lot of material and ideas with the lecture held at the 2019 Flow/Interface School on Machine Learning and Data Driven Methods, KTH, Stockholm 2019[10] (the slides of this extended lecture are available at https://folk.uio.no/jeanra/Research/material_lecture_DRL_FM_Rabault_2019.pdf). The aim of this review is therefore to offer a self-contained summary of the material presented there.

The organization of the paper is as follows. First, a brief introduction to both ANNs and DRL is provided. Then, the application of DRL to both

* **Biography:** Jean Rabault (1990-), Male, Ph. D.,
E-mail: jeanra@math.uio.no
Corresponding author: Hui Xu,
E-mail: dr. hxu@sjtu.edu.cn

optimal control and optimal design is presented. Finally, we discuss future prospects and offer some words of conclusion.

1. A brief introduction to artificial neutral networks and deep reinforcement learning

1.1 *Artificial Neutral Networks and supervised learning*

ANNs are based on ideas proposed as early as the 1950s[11], which take their inspiration in a loose analogy to the functioning of the cortex of rats. More specifically, an ANN is defined as a set of neurons, where each neuron is a simple computational unit[12-13]. In general, only feedforward (i.e., acyclic) ANNs are considered, and a neuron is composed of two parts. First, it collects information from a series of inputs and performs a weighted sum of these, and second, it applies an activation function to the result obtained at the first stage. In this context, the set of weights of all the neurons of the network defines the parametrization of the model. Usually, an ANN is composed of a series of layers of neurons, where the outputs of all neurons of a layer are used as inputs to the next layer. This is summarized in Fig. 1. In addition, specific architectures can be used to reflect the structure of a given problem. The most famous example there is the Convolution layer[14], which considers the input to the layer as a N-dimensional structured data and generates as output a similarly structured data by application of a convolutional kernel on the input, following a regular pattern. Therefore, convolutional neural networks (CNNs) effectively apply the same local transformation on smaller parts of their input, which enforces both locality and translational invariance of the processing they perform.

From a formal mathematical point of view, the interest of ANNs lies in their property of being universal approximators[15]. What is meant there, is that an ANN using a suitable nonlinear activation function can, given that its size is large enough, perform an arbitrarily good approximation of any function from its input to its output layers. However, while this result holds in theory, in practice there are many challenges to performing effective learning. First, this theoretical result says nothing about which exact activation function to use, or how large the network should be. Therefore, a large body of literature has discussed both these questions. In practice, it is commonly agreed nowadays that a simple activation function, such as the rectified linear unit (ReLU), is often a good choice[16]. Similarly, the literature reports that the depth of ANNs is important for their efficiency and descriptive power[13]. Therefore, recent state of the art ANNs can feature up to over a hundred layers[2]. However, this increased depth comes with many challenges, in particular the risk for the signal propagating in the ANN to either explode or decay exponentially with increasing depth[17]. As a consequence, both regularization techniques (dropouts[18], batch normalization[19]), specific network initializations (Xavier initialization[17], He initialization[20]), and specific architectures (UNet[21], Resnets[2]) have been developed to help solve these challenges. These technical refinements, together with the use of backpropagation of errors as a training method[13], and some effective batch optimizers[22-23], are now the most common methodology used to train ANNs to model a given function or dataset. Following these advances, ANNs have now reached the point where, at least from the point of view of most practitioners, supervised learning, i.e., learning a function from a large database of labeled examples, works satisfactorily on many tasks such as image classification or speech analysis as long as enough data and computational power are available.

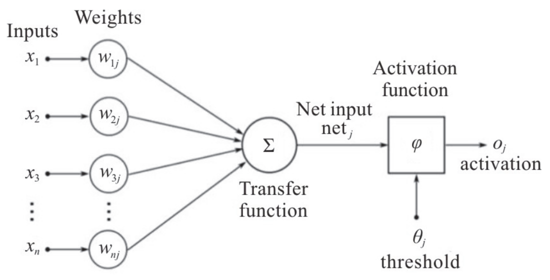

Fig.1(a) Illustration of the functioning of an artificial neuron

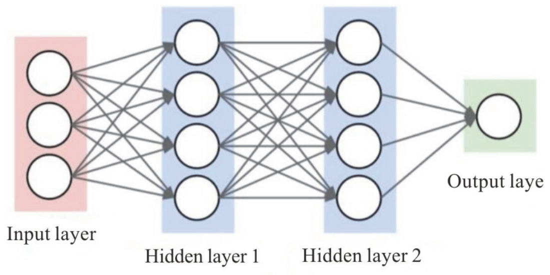

Fig. 1(b) (Color online) Neurons can be organized in layers to define a fully ANN

The use of ANNs is made especially easy thanks to the development of several high quality open source implementations. These include, to name but a few, Tensorflow[24] (a framework developed largely by Google), and Torch / pyTorch[25] (developed by Facebook). Thanks to these frameworks, using ANNs has become very easy and the practitioners can focus purely on their problem, rather than in the low-level implementation of all the technical features of modern ANNs.

1.2 *Deep reinforcement learning, policy gradient method, and the proximal policy optimization algorithm*

As a consequence of this success, ANNs and supervised learning can also be used as building blocks in more sophisticated algorithms. One such field of application that is encountering large success recently is the deep reinforcement learning (DRL) methodology[8]. In DRL, the ANN is used as a function approximator being part of a larger algorithm that performs learning through trial-and-error. For this, a formal framework is defined encompassing on the one hand the agent, i.e., the ANN together with algorithms that allow to perform the trials-and-errors and learning, and on the other hand the environment, i.e., the system to control. The interaction between the agent and the environment takes place through 3 standardized channels of interaction: a state observation s that can be noisy, partial, and stochastic, an action a that is the control applied on the environment, and a reward signal r provided by the environment to the agent that indicates how good the current state of the environment is. This interaction naturally takes place in a closed-loop fashion, and the aim of DRL is to maximize the obtained reward by judicious choice at each step of the control, given the state information. The general DRL framework is illustrated in Fig. 2. Several striking, high-profile successes have been obtained using DRL and its refinements in recent years, such as winning at the game of Go against the best human player[26-27], or effectively controlling server farms to reduce cooling electricity consumption[28].

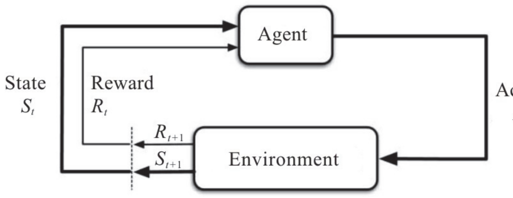

Fig. 2 Illustration of the DRL framework. A closed control loop is defined through the interaction between the environment and the agent through the state and action channels of communication, and the reward signal is used to drive the optimization process. In this process, an ANN is used as a function approximator and trained to optimize the reward

Different algorithms can be used to perform the learning through trial-and-error. Most of them fall in two large categories, namely Q-learning which is based on the Bellman equation[29-31], and policy gradient (PG)[32]. Both categories contain many algorithmic variants and refinements[33]. In the following, we will focus mostly on PG, which has provided several algorithms regarded as state-of-the-art for controlling processes where a continuous action space is present.

In the general framework of the policy gradient method, the aim is to optimize the discounted reward $\sum_{t\geq 0}\gamma^t r_t$, where γ is a discount factor between 0 and 1, typical $\gamma = 0.95$, and t indicates timestep number. For a value of γ close to 0, only the current reward is important, while for $\gamma = 1.00$ the agent does not prefer to obtain the reward at the current time step or later in time, therefore favoring optimization over a distant time horizon. The optimization takes place with regards to the set of weights Θ of the ANN, where the ANN is learning a mapping from state inputs to actions. The actions are described through probability density functions (pdfs), which are parametrized by the outputs of the network. Depending on the nature of the output (continuous, discrete, with compact support or not), different parametric pdfs can be used. A common choice for continuous output on compact support is for example to resort to a scaled beta distribution.

In order to formulate the optimization problem as a gradient descent, it is convenient at this point to introduce the concept of trajectory in the phase space, τ. A such trajectory is a sequence of pairs of state and actions, $\tau = (s,t)_t$, where $t = 0$. N is the step in the current trajectory. Following this definition, the value function to optimize can be written as

$$V(\Theta) = E\left[\sum_{t=0}^{H} R(s_t, a_t) \mid \pi_\theta\right] = \sum_{\tau} P(\tau,\Theta)R(\tau) \tag{1}$$

where $\pi_\theta(a \mid s)$ is the policy, i.e., the probability of taking action a given the state observation s. The policy is parametrized by an ANN with set of weights θ.

Therefore, one can take the gradient of this expression, and apply simple manipulations to obtain

$$\nabla_\Theta V(\Theta) = \sum_{\tau} \nabla_\Theta P(\tau,\Theta)R(\tau) \tag{2}$$

$$\nabla_\Theta V(\Theta) = \sum_{\tau} \frac{P(\tau,\Theta)}{P(\tau,\Theta)} \nabla_\Theta P(\tau,\Theta)R(\tau) \tag{3}$$

$$\nabla_\Theta V(\Theta) = \sum_{\tau} P(\tau,\Theta)\nabla_\Theta \lg[P(\tau,\Theta)]R(\tau) \tag{4}$$

The last formula can be interpreted as a Monte Carlo formula, and estimated through sampling of enough trajectories in the phase space

$$\nabla_{\Theta} V(\Theta) \approx \hat{g} = \frac{1}{M}\sum_{i=1}^{M} \nabla_{\Theta} \lg[P(\tau^{(i)}, \Theta)] R(\tau^{(i)}) \tag{5}$$

The physical interpretation is that the gradient estimated modifies the weights of the ANN so as to increase the probability of trajectories that have a high reward, and reduce the probability of trajectories that have a low reward. The gradient of the log probability can be derived following:

$$\nabla_{\Theta} \lg[P(\tau^{(i)}, \Theta)] = \nabla_{\Theta} \lg\left[\prod_{t=0}^{H} P(s_{t+1}^{(i)} \mid s_t^{(i)}, a_t^{(i)}) \pi_{\tau}(a_t^{(i)} \mid s_t^{(i)})\right] \tag{6}$$

$$\nabla_{\Theta} \lg[P(\tau^{(i)}, \Theta)] = \nabla_{\Theta}\left[\sum_{t=0}^{H} \lg P(s_{t+1}^{(i)} \mid s_t^{(i)}, a_t^{(i)}) + \sum_{t=0}^{H} \lg \pi_{\tau}(a_t^{(i)} \mid s_t^{(i)})\right] \tag{7}$$

$$\nabla_{\Theta} \lg[P(\tau^{(i)}, \Theta)] = \nabla_{\Theta} \sum_{t=0}^{H} \lg \pi_{\tau}(a_t^{(i)} \mid s_t^{(i)}) \tag{8}$$

where we have used the fact that the transition probabilities are only functions of the environment and its dynamics, and are therefore not related to the weights of the ANN. The gradient of the policy can be directly related to the weights of the ANN through backpropagation of errors, remembering that the output of the ANN is used to parametrize the distribution from which actions are sampled.

Therefore, combining the formulas (5) and (8) provides an algorithm for performing DRL following the Policy Gradient method. Usually, deploying such algorithms takes place in two phases. First, during training, one can sample trajectories in the phase space under the policy provided by the ANN, by randomly sampling the pdf provided by the ANN at each time step. This, therefore, naturally introduces stochasticity in the trajectories taken and drives exploration. Second, once training has been performed, one can follow the optimal policy by choosing as action at each time step the peak of the pdf produced by the ANN. This exploitation of the ANN to produce a (believed to be) optimal policy is referred to in the literature as “deterministic mode” or “single runner mode”.

While in theory the method presented here could be applied as is, in practice a number of refinements are used to make the convergence more robust and stable. These refinements can include, depending on the exact DRL algorithm, the use of a memory replay buffer[34], the use of a separate network (critic network) to learn an estimator of the actualized reward[35], the imposition of limitations to the policy updates to avoid overfitting lucky events[32, 36], and corrections to the formulas (5) and (8) following importance sampling to allow the use of slightly off-policy data while training, therefore improving sample efficiency. Such improvements can be combined to formulate more efficient methods, such as the proximal policy optimization (PPO)[36], which is currently often regarded as the state-of-the-art algorithm for performing control of continuous action domain processes.

In addition to these low-level algorithmic improvements, high-level improvements can also be implemented. For example, one can enhance the exploration abilities of the PPO by implementing surprise or curiosity mechanisms[37]. For this, the ANN is asked to not only choose the next control value, but also to predict what the expected state will be after the control has been applied. Thereafter, the reward is enriched as $r_t' = r_t + s_t$, where s_t is the surprise value, i.e., a measure of the discrepancy between the state effectively reached and the state predicted by the ANN. As a consequence, the ANN builds an intrinsic reward that drives it to prioritize exploring trajectories on which the prediction of the next state following control is of bad quality. Such a region where the prediction by the ANN is in bad agreement with the next state indicates that either the corresponding region was not or poorly explored, or that complex dynamics not yet understood by the ANN take place in this domain of the phase space. Therefore, the curiosity-driven ANN is pushing itself to investigate in more details regions of the phase space where its knowledge is inaccurate, which participates to effective exploration. Unfortunately, this can lead to the ANN being stuck exploring again and again regions of true randomness, similar to the cognitive phenomenon of procrastination in human psychology[38]. Therefore, more sophisticated curiosity mechanisms have been developed, such as curiosity through reachability, that preserve the desirable properties of curiosity mechanisms while alleviating negative aspects such as procrastination. Curiosity is not the only way to enhance learning with high-level techniques. For example, authors have reported that resorting to world models[39], i.e., forcing an ANN to learn a dynamic model of the environment and training subsequently on it in addition to the main environment (an idea which is inspired by the mechanisms of dreaming in human psychology), can also improve learning. Finally, one can also take advantage of human expert knowledge, and improve the initial phases of training through showing examples to the ANN, or by carefully crafting which initial state the ANN starts to control the system from[40]. This in turn can reduce the number of steps necessary to reach a desirable state of the system, effectively cutting on the cost of exploration.

At this stage it should be pointed out that, while an intrinsically local gradient descent approach is used to train the ANN, the method as a whole performs, by contrast, a global optimization that is well adapted to fully non linear, non convex, high dimensionality systems. Indeed, the true mechanism behind the collection of data and therefore the discovery of new strategies lies in the collection of trajectories in the phase space performed during training, which presents a large stochasticity. Therefore, given that enough trajectories are sampled, the PPO algorithm has the property to explore the behavior of the environment even past low reward barriers (which are the analogy for DRL to high potential energy barriers in classical physics).

As a consequence of these strengths the PPO algorithm (or similar DRL / PG algorithms), together with large amounts of computational power to allow the sampling of a large number of trajectories in the phase space, is at the origin of several successes of the DRL approach on complex systems. While there is little interest to use DRL on systems where an optimal, analytical or semi-analytical control algorithm is known, many realistic problems of interest cannot be solved effectively using such traditional approaches in which case DRL and PPO are natural methods to investigate. This efficiency is not specific to a given field of research, which is clearly reflected in the literature since DRL methods have been found to be successful in a wide variety of domains. In the following, we will focus on applications to Fluid Mechanics, and more specifically to optimal flow control and optimal design.

Before presenting applications of DRL to fluid mechanics we want to mention that, similarly to what was described for supervised learning, high quality open source implementations of the most recent DRL algorithms such as PPO are available freely. Such implementations include, to name but a few, Tensorforce[41], and stable-baselines[42]. Both packages are implementing a wide variety of Q-learning and Policy Gradient methods and provide a number of helper functions to make deployment easier. In addition, these libraries are built on top of tensorflow, which makes them computationally effective. From the point of view of the practitioner, all what is needed to use these packages is to implement the application-specific environment. For this, it is enough to fill-in templated interface classes. For example, tensorforce provides an environment class from which custom environments should be derived. The main structure of the custom environment class should be as follows:

```
from    tensorforce.environments
import   Environment
class MyEnvironment ( Environment ) :
        def   states (self ) :
                """Returns the state space specificati on.
                """
        def   actions (self):
                """Returns the action space specificat
                ion . """
        def   reset (self):
                """Resets the environment to start a new
                episode .
              Returns:    dict[state]: Dictionary contain
              ing initial state(s) and auxiliary informati
              on ."""
        def   execute (self , actions) :
                """Executes the given action(s) and adv
                    ances the environment by one step.
             Args: actions (dict[action]): Dictionary
             containing action(s) to be executed
             Returns: ((dict[state], bool |0|1|2, float)):
             Dictionary containing next state (s),
             terminal information, and observed
             reward."""
```

The structure of this class closely follows the DRL framework presented in Fig. 2. At the start of each episode (i.e., trajectory in the phase space), the reset method is called to initialize the environment and derive the initial state. Following this initialization, the execute method can be called as many time as requested to enforce the closed-loop interaction. Finally, a number of additional functions and example scripts are provided to the user on for example the Tensorforce repository https://github.com/tensorforce/tensorforce, to help perform both training and single run exploitation. For a complete overview, see either the tensorforce documentation https://tensorforce.readthedocs.io/en/latest/, or one of the active flow control codes released as open source https://github.com/jerabaul29/Cylinder2DFlowControlDRL.

2. Application to control

Control of intermediate to high Reynolds number flows is a natural field of application for DRL. Indeed, the complexity of such flows is usually too high for traditional mathematical methods to provide positive results and, therefore, the field of flow control is famously difficult to apprehend with analytical tools[43]. As a consequence, two main applications of DRL can be found there. The first one is the high-level control of large systems, such as the swimming of a fish-like geometry, or the flying of a bird-like object. The second one is the finer grained, more detailed active flow control at the small scale of flow instabilities and separation.

Arguably, this first domain of application (i.e., the high-level control of large systems) is at the interface between robotics, biomimetism, and fluid-mechanics. Several high-profile works have been published on this topic recently. To name but a few, the swimming of fish schoolings has been investigated through simulations and revealed how the vortex alley generated by a leader fish can be exploited by a follower fish to reduce energy consumption and increase swimming speed[44]. This, in turn, may find applications for robotic swarms. Another high-profile application of DRL to high-level control problems can be found in Reddy et al.[45]. There, the authors used DRL to train an algorithmic pilot that should keep a glider in the air by taking advantage of thermic ascendant currents. This is an application that may prove useful for increasing the flight duration of light high-performance aerial unmanned vehicles (AUVs). To continue with AUVs, a recent work has highlighted the potential of DRL to replace also their attitude control algorithms (usually implemented using a PID controller or similar), which leads to increased flight performance and maniability[46]. Such enhanced performance using DRL algorithms has also been observed when controlling quadcopter drones hoovering in challenging conditions, such as low-altitude translation where the drone interacts with its own turbulent wake[47]. Finally, a recent work demonstrated the applicability of DRL for high-level trajectory planning in complex environments where a turbulent velocity map distorts the motion of an active drifter[48].

The second domain of application is interested in performing active control of flow instabilities and separation. This, in turn, can lead to positive effects such as reduced drag and reduced lift fluctuations in engineering designs. Several recent works have, there also, highlighted the potential of DRL. The first results served as a proof-of-concept of the methodology and illustrated possible gains on a very simple flow configuration, namely, the 2-D Karman alley behind a cylinder at a moderate Reynolds number of $Re = 100$ [49]. There, it was shown that DRL could learn accurate control laws able to sensibly reduce drag by using small jets on the top and bottom of the cylinder. The effect of control on the flow configuration is illustrated in Fig. 3. Since the control can be applied at the location of the flow separation, the mass flow rate necessary to change the flow configuration and reduce drag can be as low as a fraction of a percent of the incoming mass flow rate intersecting the cylinder. This example is illustrative of the potential of DRL compared with other methods, such as the adjoint or parameter grid search. Compared with the adjoint, only a limited insight into the system is necessary to perform control, and as little as 5 probes are sufficient to apply a sensible strategy. Moreover, once training has been performed, obtaining the next control from a series of probes measurements is very computationally inexpensive - as it is enough to just propagate the data of the probes through the ANN. Compared with a grid search, which could for example investigate the optimum jet forcing of the form $A\cos(\omega t)$, where A and ω are the parameters to optimize, the DRL strategy is far more refined. Indeed, the control law found by the DRL agent is composed of two phases: first, a few cycles of relatively large actuations are performed to change the configuration of the flow. By contrast, after the configuration of the flow has been modified,

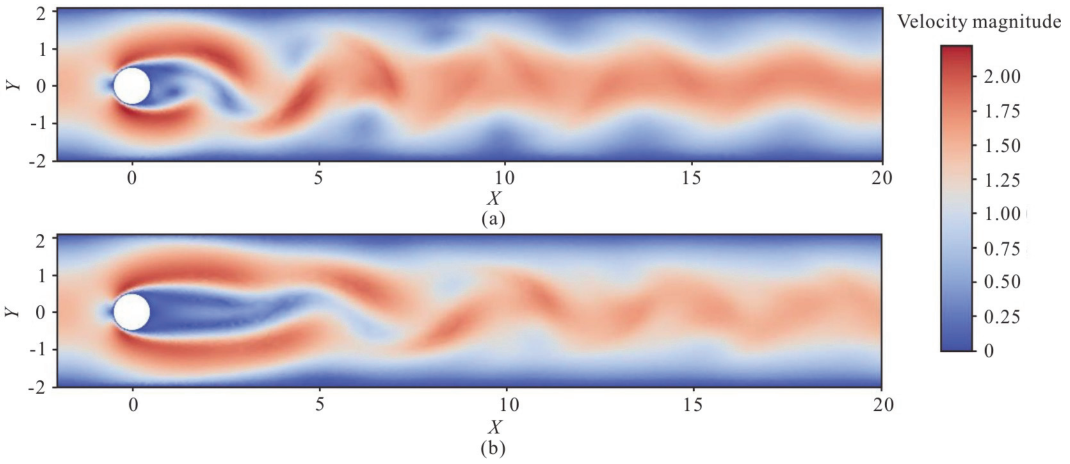

Fig. 3 (Color online) Illustration (velocity magnitude snapshot) of the effect of the active flow control strategy found by the PPO algorithm on the flow configuration behind a 2-D cylinder at Reynolds number 100. The flow is altered in a way similar to what would be obtained with boat tailing, effectively reducing drag and lift fluctuations. Figure reproduced from Rabault et al.[49]

the amplitude of the control is much reduced. This is illustrated in Fig. 4. The frequency of the control also changes as time evolves, and the final vortex shedding frequency in the reduced drag configuration is modified compared with that of the baseline flow.

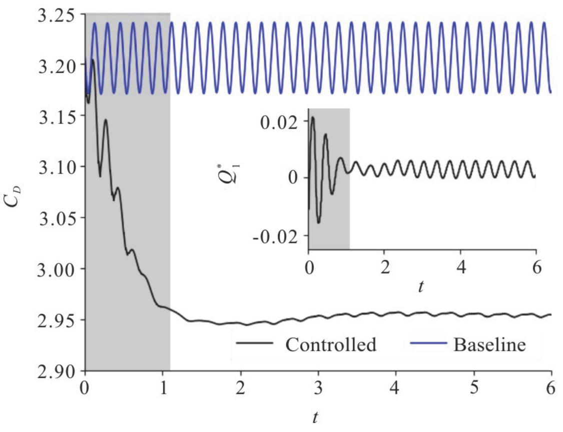

Fig. 4 (Color online) The control law applied by the PPO agent on the simulation presented in Fig. 3. Two phases are clearly visible in the control law. The normalized mass flow rate Q is only a few percents of the mass flow rate intersecting the cylinder. Figure reproduced from Rabault et al.[49]

Therefore, it appears that DRL can be used as an experimental tool to study the properties of complex flows. Going to higher Reynolds numbers leads to more and more complex dynamics, and underlines even better the adaptability of DRL algorithms (ongoing work, see preliminary reports by Ren et al.[50]). In particular, DRL can effectively find control strategies that adapt to stochastic, noisy systems where the closed-loop control allows to react to its dynamics. In order to control more complex cases, it becomes important to speed up the training phase. Luckily, this can be easily performed. Indeed, one can observe from Eq. (5) that the Monte Carlo sampling of trajectories in the phase space can be performed independently on the different trajectories. This, in turn, means that several simulations can be run completely independently of each other in a perfectly parallel manner for collecting trajectories in the phase space. This results in large performance gains in cases where the simulation of the environment is the limiting factor in terms of computational resources, which is the case when CFD simulations are used to train the DRL agent. Speedups of up to 60 times have been reported when applying such a parallelization technique[51]. In addition, this technique has the potential to provide even larger performance gains for more challenging flows. Indeed, the more complex the system to control, the more trajectories in the phase space are needed to compute the policy gradient for the update of the ANN.

Some additional important technical considerations are also presented in Rabault and Kuhnle[51]. In particular, it is highlighted that the frequency of the action update by the controller should be carefully adapted to the system to control. This is easily understandable from simple physical considerations. Indeed, the control frequency should be at least high enough to control the underlying phenomena, which translates following the Nyquist criterion to the condition that $f_a > 2f_s$, where f_a is the frequency of the action update, and f_s the underlying typical frequency of the system to control. Similarly, the frequency of the action update cannot be too high. Indeed, the exploration of the control law by the DRL agent is based on trial-and-error. Therefore, if the frequency of the action update is too high, the DRL algorithm will effectively force the system to control with high frequency white noise at the beginning of training. In this case, the probability that the white noise generated has some features that perform some effective control on the system is very low, and the trial-and-error attempts will likely fail to produce any progresses. By contrast, if the frequency of the action update is slightly more moderate, the probability of a "lucky attempt" that does affect the system in a positive way is much higher. This leads to a sweet spot for the action update frequency, typically around $f_a \approx 10 f_s$. If this frequency of update is different from the numerical time step of the simulation, one needs to apply interpolation of the control between action updates. In the case of a system presenting multiscale properties, such a clear frequency f_s cannot be easily defined. In this case, it is possible to extend the output of the policy with one additional value that lets the ANN choose also the duration until the next control update.

Another important point also highlighted in Rabault and Kuhnle[51] is that the DRL practitioner should be careful in choosing a reward function that does not push the DRL agent to perform degenerate, or "cheating", optimization. The ability of DRL algorithms to take advantage of weaknesses of the environment to artificially improve their reward in unexpected ways is well known from experimentations with video games[52-53]. Similar challenges are observed in the case of flow control. For example, an effective reduction of the drag value can be obtained by strong biased blowing upwards or downwards on the sides of a cylinder, but in this case a large value of lift is produced at the same time as drag is reduced due to the bending of the wake. However, this is not really a strategy that performs flow control (or not only flow control). Therefore, adding a lift penalization to the reward function is found to be necessary in order to observe effective, unbiased flow separation control in the case of the 2-D cylinder.

In addition to these considerations about the application of DRL, the results of Belus et al.[54]; Rabault et al.[55] illustrate the importance of using the locality and invariance properties of the underlying environment in order to make learning more tractable. This is especially true for systems that present a control domain of large dimensionality. Indeed, as highlighted by Belus et al.[54], failing to take these properties into account can lead to prohibitive exploration and learning costs, effectively making DRL training impossible, even when quite simple systems are considered. By contrast, satisfactorily reproducing locality and invariance in the architecture of the ANN, for example through the use of convolutional networks or by using cloned ANNs to control different parts of the environment as shown in Fig. 5, allows to successfully discover effective control strategies even when large systems are considered.

Following these promising results, several groups are now working towards obtaining effective control laws in more challenging situations. These include, pushing higher the Reynolds number in simulations similar to the 2-D cylinder control of Rabault et al.[49] (see the preliminary presentations of Ren et al.[50] for example on this aspect), and performing control of more realistic 3-D flows. This approach should probably be adopted also in future applications of DRL: first, one has to start with a case of reduced complexity to apprehend some of the potential pitfalls of DRL and how to mitigate them, before moving to the full complexity problem.

While the application of DRL to active flow control in simulations starts to be well established and to diffuse in the fluid mechanics community (see the work of other groups on different configurations and problems: control of a chaotic system[56], control of Rayleigh-Benard convection[57]), no physical experiments have been presented yet, to the best of the knowledge of the authors, in the context of fluid mechanics (of course, many applications have been presented in the context of robotics and control of industrial systems). This may be related to several challenges. First, building an experiment is nowadays more time consuming and complex than setting up a simulation using an open source simulation package. Second, applying DRL to experiments effectively sets demanding requirements on the underlying electronics and computer infrastructure. Indeed, the network prediction and training must be performed within tight time constraints to follow up with the physical phenomena. While this is in theory not a specially problematic difficulty, since the availability of low-latency real time OSs, fast FPGAs, and effective neural network accelerators such as GPUs is well established, in practise this demands that experimental fluid mechanics teams extend their technical skillset which, while not a fundamental challenge, may take some time. On this aspect, it would be very useful if some groups with dedicated hardware competence in the Fluid Mechanics or robotics community could release designs of effective, GPU or FPGA-accelerated systems adapted to the training of real-world sys-

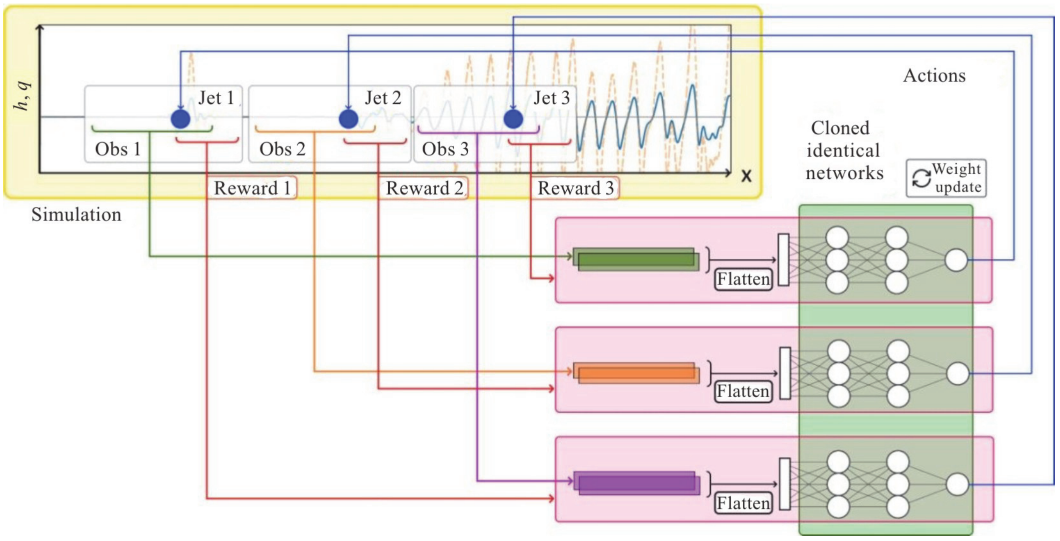

Fig. 5 (Color online) In the case when systems with large control space dimensionality are considered, exploration and training can reach prohibitive costs. However, such systems also usually feature some properties of locality and invariance. This can be exploited effectively, either by using convolutional networks, or by resorting to cloned ANNs as shown in this figure. By using cloned ANNs, it is possible to both apply the same strategy on all outputs, therefore reducing the cost of exploration, and to simultaneously increase the volume of reward generated, therefore, providing unambiguous signal during training and improving training quality. Figure reproduced from Belus et al.[54]

tems in a similar way as open source software packages are released. Third, and maybe more fundamental, it is usually more challenging to acquire large amounts of real-time, high accuracy data within an experiment than in a simulation environment. For example, the main training of Rabault et al.[49] uses 151 velocity probes located both around the cylinder and further in its wake. Performing such large data acquisition in real time may be challenging even with high speed PIV in a carefully crafted experiment, and it will definitely be very challenging, if doable at all, in a realistic setup corresponding to an industrial application. However, there also some solutions exist. In particular, Rabault et al.[49] also shows that the number of probes can be drastically reduced with only a limited decrease in the control efficiency, at least in the situation investigated. Preliminary results (ongoing work in our group) even indicate that, in a case similar to Rabault et al.[49] aiming at controlling the flow around a 2-D cylinder at moderate Reynolds numbers, having as state observation a few pressure probes located at the surface of the cylinder is enough to derive a meaningful strategy. This, in turn, is very achievable in a real-world setup. Finally, the last difficulty is that one needs to be able to implement physical actuators that can follow up with the control strategy in order to apply the control to a real world system. While there are many candidates for such actuators and the literature usually considers that control algorithms, rather than physical actuators, are the main limitation to performing active flow control[58-59], this is yet another technical domain that needs to be mastered to perform a physical experiment.

Therefore, ideally, the numerical simulations and the experiments should be complementary of each other. The numerical simulations can be used as easily set-up benchmark cases, to be tested and well understood before real-world implementation. In addition, it is well established that ANNs can, given suitable care is taken during training, be trained or at least pre-trained in simulations before transferring the strategies learnt to the real world[60] (see https://www.youtube.com/watch?v=aTDkYFZFWug&feature=youtu.be for a nice illustration). These abilities of transfer learning and mitigation of the reality gap between simulations and the real world are one more strength for DRL compared with other methodologies. The strength of experiments, by contrast with simulations, will be both to present indisputable real-life validation of the strategies and methodologies revealed by simulations, and to open the way to full scale Reynolds numbers applications. Indeed, real-life experiments will be the only way to generate large amount of training data in situations as costly to simulate as realistic full-scale flows.

3. Application to shape optimization

As a second possible domain of application for DRL within fluid mechanics both shape optimization and, more generally, the workflow of a CFD engineer designing a part can be considered. This is especially attractive when sophisticated geometries have to interact with complex nonlinear flow features, which can be a challenge to adjoint-based methods due to the existence of many local minima.

However, applications in this domain are still relatively less developed than in the case of flow control. This may be explained in part due to the existence of other well established non-gradient-based methods for these tasks, such as evolution-based algorithms. In addition, it is slightly less clear how DRL should be applied to such problems. Indeed, one can either attempt to perform “one shot optimization” where the ANN has to directly describe the shape it believes to be optimal, or “iterative optimization” where the ANN is allowed to incrementally deform the shape to make it evolve towards an optimal geometry. Since DRL is usually more efficient in closed-loop setups, as this is how the algorithms were initially thought of and designed, the second approach may be more promising on the long term. However, implementing such iterative method puts harder requirements on for example the mesh deformation and topology checking, and as a consequence the applications of DRL to shape optimization that the authors are aware of are based on one shot optimization using parametric models.

The first such application considers aerodynamic optimization of a missile-like object flying within given specifications[61]. The DRL algorithm starts from a reasonable missile shape, and is charged to perform limited changes to the geometry of the missile in order to improve the flight performance. The geometry changes are characterized through a series of design parameters such as wing dimensions or missile length, as visible in Fig. 6. In this work, it is shown that while a policy gradient DRL algorithm (in this case deep deterministic policy gradient (DDPG)) is beaten by other trial-and-error methods when applied as is out-of-the-box, a slightly tuned version of this algorithm introduced by the authors (SL-DDPG) beats these other techniques in terms of both speed of convergence and quality of the optimal configuration found, as also visible in Fig. 6.

In the second application of DRL to shape optimization that the authors are aware of[62], the PPO algorithm is charged with creating a shape that maximizes lift at moderate Reynolds numbers. By contrast to the first application, this time the DRL algorithm has to perform one-shot optimization starting from a geometry that does not produce any lift, in this case a cylinder discretized with second-order

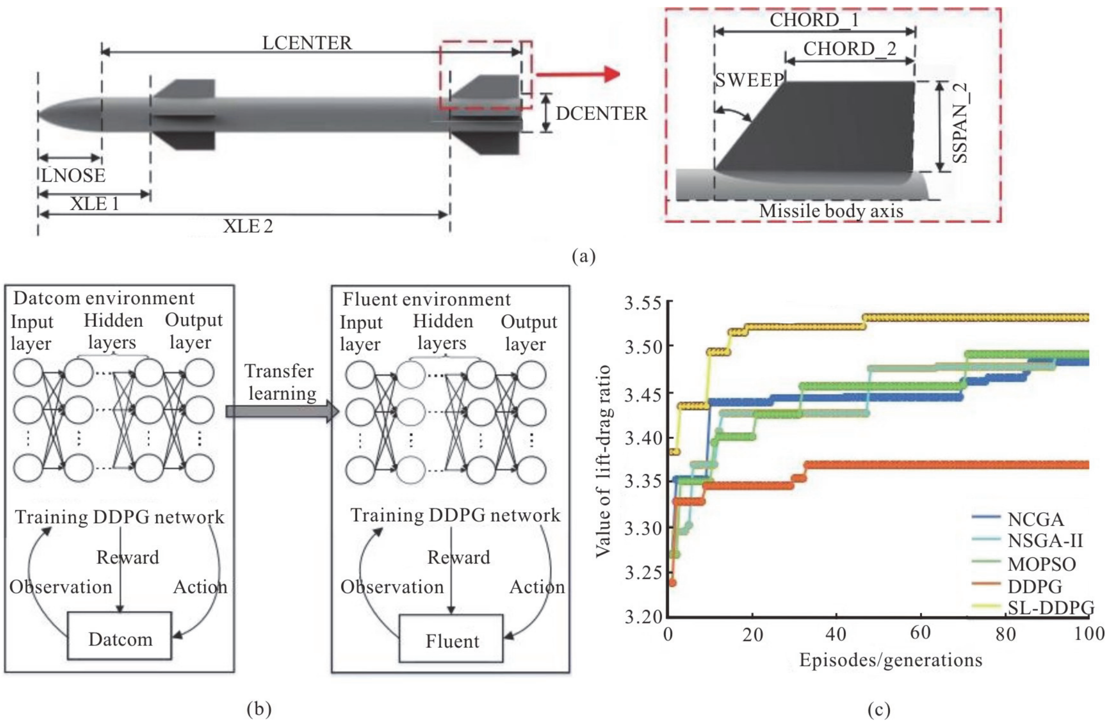

Fig. 6 (Color online) Illustration of the optimization parameters used for maximizing the flight performance of a model missile. The results indicate that the tuned DDPG method is both competitive in terms of absolute performance as well as sample effective compared with traditional optimization methods. Figure reproduced from Yan et al.[61].

splines. There also, it is found that the DRL algorithm successfully performs optimization upon unambiguous parametrization of the geometry to optimize by splines, which is obtained by limiting the free domain of the defining points to separate quadrants in order to avoid overlap and self-crossing of the shape boundary. The parameters defining the splines are also re-normalized to a typical range of $[-1,1]$, where the ANNs perform best. In addition, non-meshing shapes that can be obtained for example by using too high local curvatures of the shape are punished by applying a negative reward and immediately starting the next optimization step. Following these additions to the methodology, the DRL algorithm is there also able to produce shapes that generate large values of lift. Unsurprisingly, the shapes obtained are wing-like, with an exact morpho- logy that depends on the number of spline points let free to be used in the optimization.

While these works confirm that DRL also has some potential in shape optimization, several additional improvements should be investigated in the future. First, making the DRL algorithm aware of the details of the flow configuration by giving it access to the flow fields should help to build a high-level understanding of the underlying problem. Second, one can expect that using a parametrization of the shape being optimized that is not case-specific, for example by letting the ANN deform the mesh of the shape rather than just change a few characteristic coefficients, would allow to offer a more general optimization context and lead to some possibilities for generalization and transfer learning, which are known strengths of ANN-based algorithms. However, this may require additional complexity in the implementation of the environments, since mesh morphing techniques are more complex to implement than parametric shape descriptions.

4. Future prospects and consclusions

The interest in using DRL techniques for Fluid Mechanics applications is rapidly growing, following the many successes of this technique in a wide range of domains of science. This is for example visible through the number of DRL-centered publications at the APS/DFD meeting. While in 2018 only 1 talk was focused on DRL, in 2019 over 10 talks were specifically mentioning DRL in their title (data collected by searching after DRL key words in the titles of talks at the APS/DFD from the official websites: http://meetings.aps.org/Meeting/DFD18/SearchAbstract and http://meetings.aps.org/Meeting/DFD19/SearchAbstract).

The reason for this interest lies in the potentialities of DRL to successfully analyze situations that are challenging for more classical methods due to the combination of non-linearity, non-convexity, and high dimensionality, that can be found in many complex problems. The ability to perform well on such challenging problems comes from the nature of DRL algorithms. Indeed, since they are performing exploration based on a trial-and-error approach, they are not easily trapped in suboptimal local extrema and are, therefore, well adapted to systems with nonlinear stochastic dynamics and partial observations.

As a consequence, DRL is especially promising for at least two applications within Fluid Mechanics that correspond well to its formal framework, namely active flow control and shape optimization. So far, applications for both cases have been performed on relatively simple benchmarks. However, several groups are working towards the use of DRL in more challenging configurations, and the good results obtained on the simplest cases are so far promising for further application of this methodology. In addition, starting with simple configurations allows to quickly test different ways to use DRL, and to gain experience on the main points one should be careful about when trying to apply DRL to a new problem. In particular, at least four important factors were highlighted by the works performed so far. First, DRL can be relatively data-hungry, but luckily the algorithm is naturally parallel which allows to obtain large speedups given that enough computational resources are available. Second, renormalization of data to a range close to $[-1,1]$ where the discontinuity of the neurons is active is key to the good functioning of these methods. Third, the choice of the reward function is both of key importance for the success of optimization, and sometimes challenging. Indeed, the DRL algorithms are very efficient at finding flaws or shortcuts in the environments, and therefore they easily optimize naive reward functions in unexpected and unwanted ways. Fourth, the application of DRL to large systems with many control values can become challenging due to the curse of dimensionality on the control space dimension, which translates into prohibitive exploration costs. Luckily, exploiting locality and invariants in the environment to control allows for effective mitigation of this difficulty.

Following these progresses, it is now possible to envision the control of higher Reynolds number, 3-D flows that are closer to real-world systems. In addition, the use of DRL in real-world experiments rather than simulations appears more likely as the strengths of the method are revealed through numerical benchmarks. Ultimately, one can hope that DRL will become both a tool of practical importance for industrial applications, and a methodology that allows to investigate general properties of flows through an empirical approach.

Finally, we note that the Fluid Mechanics community is by no ways alone in investigating the use of DRL on complex physical or mechanical systems. Arguably, many of the applications mentioned in this review are at the interface between fluid mechanics, robotics, biomimetism, and computer science. As a consequence, there is much knowledge and experience to gain from adopting a multidisciplinary approach, at least when reading through the DRL literature. This is particularly true when general questions around DRL are investigated, such as the possibility for transfer learning or bridging the reality gap between training in simulations and application of the policies found thereafter to the real world. Similarly, it is probably important for the Fluid Mechanicists to be careful to not ignore tools and software packages released by these other communities. For example, many excellent resources and implementations are available open source around the PPO algorithm, and it is probably a good idea to resort to these well-tested and benchmarked code bases, and help provide feedback and improve them when necessary, rather than re-developing from scratch new tools that largely re-implement the same features. As a consequence, a key enabler to further DRL applications will likely be the ability of the community to commit to further sharing code and benchmarks as open source, so as to reduce the entry barrier for new groups who want to participate in DRL research. On this aspects, the openness of the DRL community, that has been widely sharing code and software packages, must be acknowledged. We hope such openness can continue also thanks to Fluid Mechanicists themselves, for example with the sharing of effective hardware designs to make DRL easier to use in real-world experiments with hard time constraints.

Acknowledgements

This work was supported by the National Numerical Wind Tunnel Project (Grant No. NNW2019ZT4-B09), the National Natural Science Foundation of China (Grant Nos. 91852106, 91841303).

References

[1] Krizhevsky A., Sutskever I., Hinton G. E. Imagenet classification with deep convolutional neural networks [C]. *Advances in Neural Information Processing Systems*, Lake Tahoe, USA, 2012, 1097-1105.

[2] He K., Zhang X., Ren S. et al. Deep residual learning for image recognition [C]. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, USA, 2016, 770-778.

[3] Rabault J., Kolaas J., Jensen A. Performing particle image velocimetry using artificial neural networks: A proof-of-concept [J]. *Measurement Science and Technology*, 2017,

28(12): 125301.

[4] Kober J., Bagnell J. A., Peters J. Reinforcement learning in robotics: A survey [J]. *The International Journal of Robotics Research*, 2013, 32(11): 1238-1274.

[5] Gu S., Holly E., Lillicrap T. et al. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates [C]. *2017 IEEE international conference on robotics and automation (ICRA), IEEE*, Singapore, 2017, 3389-3396.

[6] Schmidhuber J. Deep learning in neural networks: An overview [J]. *Neural networks*, 2015, 61: 85-117.

[7] Lillicrap T. P., Hunt J. J., Pritzel A. et al. Continuous control with deep reinforcement learning [EB/OL]. arXiv preprint, 2015, arXiv:1509.02971.

[8] Sutton R. S., Barto A. G. Reinforcement learning: An introduction [M]. Cambridge, USA: MIT press, 2018.

[9] Rabault J., Zhang W., Xu H. Deep reinforcement learning in fluid mechanics: a promising method in both active flow control and shape optimization [C]. *International Symposium on High Fidelity Computational Methods and Applications*, Shanghai, China, 20129.

[10] Rabault J. Deep reinforcement learning applied to fluid mechanics: materials from the 2019 flow/interface school on machine learning and data driven methods [C]. *International Symposium on High Fidelity Computational Methods and Applications*, Shanghai, China, 2019.

[11] Rosenblatt F. The perceptron: A perceiving and recognizing automation [R]. New York, USA: Cornell Aeronautical Laboratory, 1957, Report 85-60-1.

[12] Goodfellow I., Bengio Y., Courville A. 2017 The Deep Learning Book [M]. Cambridge, USA: MIT Press.

[13] LeCun Y., Bengio Y., Hinton G. Deep learning [J]. *Nature*, 2015, 521(7553): 436-444.

[14] LeCun Y., Bengio Y. Convolutional networks for images, speech, and time series (The handbook of brain theory and neural networks) [M]. Cambridge, USA: MIT Press, 1998.

[15] Hornik K,, Stinchcombe M., White H. Multilayer feedforward networks are universal approximators [J]. *Neural Networks*, 1989, 2(5): 359-366.

[16] Glorot X., Bordes A., Bengio Y. Deep sparse rectifier neural networks [C]: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, Rome, Italy, 2011, 315-323.

[17] Glorot X., Bengio Y. Understanding the difficulty of training deep feedforward neural networks [C]. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Chia Laguna Resort, Italy, 2010, 249-256.

[18] Srivastava N., Hinton G., Krizhevsky A. et al. Dropout: A simple way to prevent neural networks from overfitting [J]. *The Journal of Machine Learning Research*, 2014, 15(1): 1929-1958.

[19] Ioffe S., Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift [EB/OL]. arXiv preprint, 2015, arXiv:1502.03167.

[20] He K., Zhang X., Ren S. et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification [C]. *Proceedings of the IEEE International Conference on Computer Vision*, Santiago, Chile, 2015, 1026-1034.

[21] Li X., Chen H., Qi X. et al. H-DenseUNet: Hybrid densely connected UNet for liver and tumor segmentation from CT volumes [J]. *IEEE Transactions on Medical Imaging*, 2018, 37(12): 2663-2674.

[22] Kingma D. P., Ba J. Adam: A method for stochastic optimization [EB/OL]. arXiv preprint, 2014, arXiv: 1412.6980.

[23] Duchi J., Hazan E., Singer Y. Adaptive subgradient methods for online learning and stochastic optimization [J]. *Journal of Machine Learning Research*, 2011, 12(7): 2121-2159.

[24] Abadi M., Barham P., Chen J. et al. Tensorflow: A system for large-scale machine learning [C]. *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI'16)*, Savannah, USA, 2016, 265-283.

[25] Paszke A., Gross S., Massa F. et al. PyTorch: An imperative style, high-performance deep learning library [C]. *Advances in Neural Information Processing Systems*, Vancouver, Canada, 2019, 8024-8035.

[26] Silver D., Schrittwieser J., Simonyan K. et al. Mastering the game of go without human knowledge [J]. *Nature*, 2017, 550(7676): 354-359.

[27] Silver D., Hubert T., Schrittwieser J. et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play [J]. *Science*, 2018, 362(6419): 1140-1144.

[28] Knight W. Google just gave control over data center cooling to an AI [EB/OL]. 2018, https://www.technologyreview.com/s/611902/google-just-gave-control-over-data-center-cooling-to-an-ai/.

[29] Bellman R. A Markovian decision process [J]. *Journal of Mathematics and Mechanics*, 1957, 6(4): 679-684.

[30] Bellman R. E., Dreyfus S. E. Applied dynamic programming [M]. Princeton,USA: Princeton University Press, 2015.

[31] Van Hasselt H., Guez A., Silver D. Deep reinforcement learning with double q-learning [C]. *Thirtieth AAAI Conference on Artificial Intelligence*, Phoenix, USA, 2016.

[32] Schulman J., Levine S., Abbeel P. et al. Trust region policy optimization [C]. *International Conference on Machine Learning*, Guangzhou, China, 2015, 1889-1897.

[33] Garnier P., Viquerat J., Rabault J. et al. A review on deep reinforcement learning for fluid mechanics [EB/OL]. arXiv preprint, 2019, arXiv:1908.04127.

[34] Schaul T., Quan J., Antonoglou I. et al. Prioritized experience replay [EB/OL]. arXiv preprint, 2015, arXiv:1511.05952.

[35] Pinto L., Andrychowicz M., Welinder P. et al. Asymmetric actor critic for image-based robot learning [EB/OL]. arXiv preprint , 2017, arXiv:1710.06542.

[36] Schulman J., Wolski F., Dhariwal P. et al. Proximal policy optimization algorithms [EB/OL]. arXiv preprint, 2017, arXiv:1707.06347,.

[37] Achiam J, Sastry S. Surprise-based intrinsic motivation for deep reinforcement learning[J]. arXiv preprint, 2017, arXiv:1703.01732.

[38] Savinov N., Raichuk A., Marinier R. et al. Episodic curiosity through reachability [EB/OL]. arXiv preprint, 2018, arXiv:1810.02274.

[39] Ha D., Schmidhuber J. World models [EB/OL]. arXiv preprint, 2018, arXiv:1803.10122.

[40] Salimans T., Chen R. Learning Montezuma's revenge from a single demonstration [EB/OL]. arXiv preprint, 2018, arXiv:1812.03381.

[41] Tensorforce: A tensorflow library for applied reinforcement learning [EB/OL]. 2017, https://tensorforce.readthedocs.io/en/latest/.

[42] Stable baselines [EB/OL]. 2018, https://github. com/hill-a/stable-baselines.

[43] Duriez T., Brunton S. L., Noack B. R. Machine learning

control-taming nonlinear dynamics and turbulence [M]. Cham, Switzerland: Springer International Publishing, 2017.

[44] Novati G., Verma S., Alexeev D. et al. Synchronised swimming of two fish [J]. *Bioinspiration and Biomimetics*, 2017, 12(3): 036001.

[45] Reddy G., Celani A., Sejnowski T. J. et al. Learning to soar in turbulent environments [J]. *Proceedings of the National Academy of Sciences*, 2016, 113(33): E4877-E4884.

[46] Bøhn E., Coates E. M., Moe S. et al. Deep reinforcement learning attitude control of fixed-wing UAVs using proximal policy optimization [C]. *2019 International Conference on Unmanned Aircraft Systems (ICUAS), IEEE*, Atlanta, USA, 2019, 523-533.

[47] Hwangbo J., Sa I., Siegwart R. et al. Control of a quadrotor with reinforcement learning [J]. *IEEE Robotics and Automation Letters*, 2017, 2(4): 2096-2103.

[48] Biferale L., Bonaccorso F., Buzzicotti M. et al. Zermelo's problem: Optimal point-to-point navigation in 2D turbulent flows using reinforcement learning [J]. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 2019, 29(10): 103138.

[49] Rabault J., Kuchta M., Jensen A. et al. Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control [J]. *Journal of Fluid Mechanics*, 2019, 865: 281-302.

[50] Ren F., Rabault J., Tang H. Active flow control of flow past a circular cylinder at moderate reynolds number using deep reinforcement learning [C]. *International Symposium on High Fidelity Computational Methods and Applications*, Shanghai, China, 2019.

[51] Rabault J., Kuhnle A. Accelerating deep reinforcement learning strategies of flow control through a multi-environment approach [J]. *Physics of Fluids*, 2019, 31(9): 094105.

[52] Clark J., Amodei D. Faulty reward functions in the wild [EB/OL]. 2016, https://openai.com/blog/ faulty-reward-functions/.

[53] Baker B., Ingmar K., Markov T. et al. Emergent tool use from multi-agent interaction [EB/OL]. arXiv preprint, 2019, arXiv:1909.07528.

[54] Belus V., Rabault J., Viquerat J. et al. Exploiting locality and translational invariance to design effective deep reinforcement learning control of the 1-dimensional unstable falling liquid film [J]. *AIP Advances*, 2019, 9(12): 125014.

[55] Rabault J., Belus V., Viquerat J. et al. Exploiting locality and physical invariants to design effective deep reinforcement learning control of the unstable falling liquid film [C]. *The 1st Graduate Forum of CSAA and the 7th International Academic Conference for Graduates, NUAA*, Nanjing, China, 2019.

[56] Bucci M. A., Semeraro O., Allauzen A. et al. Control of chaotic systems by deep reinforcement learning [EB/OL]. arXiv preprint, 2019, arXiv:1906.07672.

[57] Corbetta A., Beintema G., Biferale L. et al. Reinforcement learning versus linear control of Rayleigh-Bénard convection [C]. *American Physical Society, Division of Fluid Dynamics Meeting*, Philadelphia, USA, 1998.

[58] Collis S. S., Joslin R. D., Seifert A. et al. Issues in active flow control: Theory, control, simulation, and experiment [J]. *Progress in Aerospace Sciences*, 2004, 40(4-5): 237-289.

[59] Cattafesta III L. N., Sheplak M. Actuators for active flow control [J]. *Annual Review of Fluid Mechanics*, 2011, 43: 247-272.

[60] Hwangbo J., Lee J., Dosovitskiy A. et al. Learning agile and dynamic motor skills for legged robots [EB/OL]. arXiv preprint, 2019, arXiv:1901.08652.

[61] Yan X., Zhu J., Kuang M. et al. Aerodynamic shape optimization using a novel optimizer based on machine learning techniques [J]. *Aerospace Science and Technology*, 2019, 86: 826-835.

[62] Viquerat J., Rabault J., Kuhnle A. et al. Direct shape optimization through deep reinforcement learning [EB/OL]. arXiv preprint, 2019, arXiv:1908.09885.