Computers and Fluids 288 (2025) 106498

Contents lists available at ScienceDirect

Computers and Fluids

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

Parallel implementation and performance of super-resolution generative adversarial network turbulence models for large-eddy simulation

Ludovico Nista [a,*], Christoph D.K. Schumann [b], Peicho Petkov [c,d], Valentin Pavlov [c], Temistocle Grenga [e], Jonathan F. MacArt [f], Antonio Attili [g], Stoyan Markov [c], Heinz Pitsch [a]

[a] Institute for Combustion Technology, RWTH Aachen University, Aachen, 52056, Germany
[b] Department of Engineering, University of Cambridge, Cambridge, CB2 1PZ, United Kingdom
[c] National Centre for Supercomputing Applications, Sofia, 1113, Bulgaria
[d] Faculty of Physics, Sofia University St. Kliment Ohridski, Sofia, 1164, Bulgaria
[e] Department of Aeronautics and Astronautics, Faculty of Engineering and Physical Sciences, University of Southampton, Southampton, SO17 1BJ, United Kingdom
[f] Aerospace and Mechanical Engineering, University of Notre Dame, Notre Dame, IN 46556, USA
[g] School of Engineering, Institute for Multiscale Thermofluids, University of Edinburgh, Edinburgh, EH93FD, United Kingdom

ARTICLE INFO

Keywords:
Super-resolution generative adversarial networks
Synchronous data-parallel distributed training
Inference-coupled large-eddy simulations
High-performance computing
Turbulence closure modeling

ABSTRACT

Super-resolution (SR) generative adversarial networks (GANs) are promising for turbulence closure in large-eddy simulation (LES) due to their ability to accurately reconstruct high-resolution data from low-resolution fields. Current model training and inference strategies are not sufficiently mature for large-scale, distributed calculations due to the computational demands and often unstable training of SR-GANs, which limits the exploration of improved model structures, training strategies, and loss-function definitions. Integrating SR-GANs into LES solvers for inference-coupled simulations is also necessary to assess their *a posteriori* accuracy, stability, and cost. We investigate parallelization strategies for SR-GAN training and inference-coupled LES, focusing on computational performance and reconstruction accuracy. We examine distributed data-parallel training strategies for hybrid CPU–GPU node architectures and the associated influence of low-/high-resolution subbox size, global batch size, and discriminator accuracy. Accurate predictions require training subboxes that are sufficiently large relative to the Kolmogorov length scale. Care should be placed on the coupled effect of training batch size, learning rate, number of training subboxes, and discriminator's learning capabilities. We introduce a data-parallel SR-GAN training and inference library for heterogeneous architectures that enables exchange between the LES solver and SR-GAN inference at runtime. We investigate the predictive accuracy and computational performance of this arrangement with particular focus on the overlap (halo) size required for accurate SR reconstruction. Similarly, *a posteriori* parallel scaling for efficient inference-coupled LES is constrained by the SR subdomain size, GPU utilization, and reconstruction accuracy. Based on these findings, we establish guidelines and best practices to optimize resource utilization and parallel acceleration of SR-GAN turbulence model training and inference-coupled LES calculations while maintaining predictive accuracy.

1. Introduction

Direct numerical simulation (DNS) accurately resolves relevant turbulence time and length scales but is prohibitively costly for many engineering applications. In the large eddy simulation (LES) approach, spatially filtered equations are solved instead, which results in significant computational cost reductions [1] . In LES, we distinguish between resolved and subfilter-scale (SFS), for which accurate closure models are crucial to account for the interaction of resolved and unresolved scales [2]. Closure models must incorporate the unknown SFS fields into the evolution equations for the filtered fields consistently with the full-scale fields. Well-known closure models include the Smagorinsky model [3], dynamic models [4], and scale-similarity models [5], among others [6].

Given the availability of comprehensive DNS datasets, data-driven modeling has emerged as a promising method for both developing novel, and augmenting existing LES closures [7–10]. One prominent data-driven strategy involves employing a super-resolution (SR) model within the context of a deconvolution procedure. This approach aims to approximate the unfiltered field u_i by a deconvoluted field $u_i^\star$, ideally with $u_i^\star \approx u_i$, such that the statistical properties of $u_i^\star$ conditioned on the

* Corresponding author.
E-mail address: l.nista@itv.rwth-aachen.de (L. Nista).

https://doi.org/10.1016/j.compfluid.2024.106498
Received 19 August 2024; Received in revised form 16 November 2024; Accepted 25 November 2024
Available online 2 December 2024

filtered field, $\bar{u}_i$, are consistent with those of the ground truth unfiltered field u_i. The deconvoluted field is obtained from a filtered field by inverting a filtering operator $\mathcal{G}$,

$$u_i \approx u_i^{\bullet} = \mathcal{G}_l^{-1} * \bar{u}_i = \mathcal{G}_l^{-1} * \mathcal{G} * u_i \,, \tag{1}$$

where $\mathcal{G}_l^{-1}$ is an l^{th}-order approximate inverse of $\mathcal{G}$, and $*$ denotes a convolution. Due to the inherent non-uniqueness of deconvolution, $u_i^{\bullet}$ is not expected to exactly match u_i in a pointwise sense. Instead, the goal is for the deconvoluted field to reproduce the correct conditional statistics of the original field given the filtered field $\bar{u}_i$. The $\mathcal{G}^{-1}$ operation is performed in a data-driven manner using suitable neural network frameworks. These networks must be trained to reconstruct high-resolution (HR) fields from low-resolution (LR) fields, thus aiming to explicitly reconstruct SFS fields on finer auxiliary grids.

The effectiveness of the deconvolution approach depends on the accuracy of the approximate inverse operator $\mathcal{G}^{-1}$. Many initial studies performed *a priori* tests using deep convolutional neural networks (CNNs) for SR, showing their improved accuracy over analytical models for in-sample predictions (i.e., tested on data that statistically matches the training data) [11–15]. However, SR-CNNs designed for SR reconstruction usually require large amounts of labeled data (as they can generally be trained only using fully supervised learning), exhibit limited generalization to out-of-sample flow conditions, and can generate unphysical, blurred SR fields [12,16–18].

To overcome the limitations of the conventional SR-CNNs, researchers have turned to the use of implicit generative models including diffusion models [19] and generative adversarial networks (GANs) [20]. Shu et al. [21] introduced a diffusion model, which requires only HR data, and exhibits the capability to reconstruct SR data from both regular LR samples and sparsely measured samples. In GAN-based SR, the generator network, built upon a CNN, super-resolves the LR fields (e.g., filtered DNS or LES or experimental data of low spatial resolution) to generate HR counterparts. Simultaneously, the discriminator network engages in semisupervised learning, distinguishing between the generated HR field and the authentic, "ground truth" HR field (e.g., the DNS itself). Throughout the training, the generator learns to produce samples that are indistinguishable from genuine HR data. The discriminator, in turn, learns to judge the authenticity of these samples. As a result, both networks improve their predictions during the adversarial training process.

Many researchers have demonstrated the capabilities of such generative methods for turbulent modeling. The capability to accurately resolve high-frequency features has made GAN-based networks attractive for SFS modeling. Hassanaly et al. [22] applied the adversarial approach to sample conditional high-dimensional distributions to deconvolute atmospheric turbulence data. Kim et al. [17] addressed the challenge of reconstructing small-scale turbulence for sparsely paired LR and HR data, adopting a cycle-consistent GAN (CycleGAN). Bode et al. [23] proposed a physics-informed SR-GAN (PIESRGAN) for SFS turbulence reconstruction incorporating a loss function based on the continuity equation residual.

Several works have investigated the out-of-sample generalization capability of SR-GAN frameworks. In particular, the model's ability to extrapolate to higher and lower Reynolds numbers, as well as different Karlovitz numbers than those used for training has been evaluated. Findings indicated that the ratio between the LES filter width and the Kolmogorov length scale must be preserved for adequate generalization (i.e., a fixed SR upsampling window) [24–26]. Recently, Nista et al. [16] investigated filter kernels and sizes inconsistent with the training and have demonstrated the advantages of adversarial training on a priori out-of-sample reconstructions.

In general, SR-GANs have been recognized for their superior reconstruction performance over conventional supervised SR-CNNs, as well as algebraic models [16,17,27–29]. However, current SR-GANs' frameworks, specifically designed for turbulence modeling, are not sufficiently mature for large-scale applications. Among other challenges, there is a need to explore regularization techniques that embed physical information, to develop more efficient upsampling layers reducing memory overhead, to develop novel convolutional blocks to capture high-frequency turbulence, and to formulate physics-informed loss functions to ensure physically consistent reconstructed fields. However, training SR-GANs is computationally demanding due to the addition of a discriminator network, its often unstable training, and the balance needed between generator and discriminator. Therefore, there is a simultaneous need to drastically accelerate the training process to facilitate experimentation with diverse model structures, hyperparameter tuning, training strategies, and loss-function definitions as well as to ensure physically meaningful predictions.

Distributed training over many nodes has become crucial in this regard, as it significantly accelerates the training process for deep learning (DL) models and enables larger training datasets and more-complex model structures. This is particularly relevant in the context of graphics processing unit (GPU)-based computing. Pioneering works by Goyal et al. [30] and Jia et al. [31] demonstrated highly scalable DL training on dense GPU clusters, establishing connections between fundamental training hyperparameters, such as the learning rate and the batch size. Similarly, Krizhevsky et al. [32] successfully trained a CNN-based framework for image classification with more than 60 million parameters and significantly enhanced its performance using synchronous data transfer over distributed computational resources.

While multiple distributed training approaches (DTAs) have been proposed for conventional supervised-based CNNs, DTAs' studies for GANs remain limited. Moreover, existing DTAs' studies are predominantly tailored for image-processing and image-classification applications. DTA studies of frameworks developed for turbulence modeling are absent, in particular for SR-GANs. This highlights the need for investigating DTAs tailored specifically for turbulence applications, which exhibit major differences compared to standard image-processing applications. Moreover, finding a suitable DTA for SR-GANs' applications would not only focus on assessing the effective utilization of parallel processing to accelerate convergence but also quantify the computational requirements for obtaining *a priori* physically meaningful predictions. In particular, the well-known training instability and mode collapse associated with (SR-)GAN models must be considered.

In addition, much of the initial SR-GAN turbulence closure efforts have focused on improving *a priori* in- and out-of-sample reconstruction capabilities. Several recent studies have acknowledged the importance of *a posteriori* evaluation [33–36], which necessarily involves integration into LES solvers. While *a priori* tests have demonstrated remarkable model-fitting capabilities, they do not guarantee that the proposed models will be accurate or even stable in LES calculations, for which the grid spacing, numerical accuracy and stability, and modeling assumptions may need to be considered directly during the training phase [33,37]. In this regard, consistency between training and LES environments is fundamental for stability. This can be achieved, for instance, with either an SR-GAN framework, which employs precomputed LES fields and adversarial training, or with deep reinforcement learning frameworks, such as the one proposed by Kurz et al. [37], where the model is trained "online" through direct interaction with the LES solution dynamics.

A posteriori LES applications of data-driven SR-models can be computationally expensive compared to standard algebraic-based applications and may also benefit from massive parallel implementation and GPU-based acceleration. At the same time, the parallelization and domain decomposition approach might affect physical accuracy and hardware efficiency. With the ongoing development of GPU-accelerated exascale computing platforms, it is desirable to harness the performance potential of these powerful node architectures [38–40]. Efficient utilization of hybrid cluster architectures, equipped with both central processing units (CPUs) and GPUs, can accelerate computational fluid dynamics (CFD)-coupled data-driven applications, as recently demonstrated by the *phyDLL* library [41]. However, there is a paucity of

efficient parallelization strategies for data-driven SR applications and the integration of both CPUs and GPUs in CFD simulations in the literature.

This study investigates various parallelization approaches for an SR-GAN framework designed for turbulent flow reconstruction, focusing on the quantification of computational performance and physical accuracy during both the training process and inference-coupled simulations. It is important to note that the methods presented here are not specific to this network and could be extended to other SR-GAN frameworks available in the literature. Different DTAs are introduced in Section 4 and investigated in Section 6 with special emphasis not only on standard hardware scalability but also on the effects of DTAs on in-sample predictive capabilities for three-dimensional turbulence reconstruction. Different training parameters are considered such as the node (CPU/GPU) configurations, LR/HR training sizes, global batch size, and discriminator accuracy. Furthermore, a modular, parallel DL library called *superLES* is presented in Section 5. This library enables on-the-fly field exchange between the LES solver and coupled SR-GAN inference at runtime. The computational performance, scalability, and physical accuracy of SR-LES computations are investigated in Section 7 using a heterogeneous cluster architecture.

The overarching goal of this work is to provide guidelines and best practices for optimizing resource utilization to accelerate both the GAN-based SR training processes and inference-coupled SR-LES computations, while maintaining predictive accuracy in turbulence modeling.

2. Datasets and preprocessing description

A forced homogeneous isotropic turbulence (HIT) DNS dataset at Taylor-microscale Reynolds number $Re_\lambda \approx 140$, referred to as Re140, is used for SR training and testing, and for inference-coupled (*a posteriori*) validation. The DNS dataset was computed using the CIAO code, which is based on the numerical algorithms developed by Desjardins et al. [42] and has been used in several DNS and LES studies in the past (for instance, [43,44]). It solves the Navier–Stokes equations in a three-dimensional periodic domain using a conservative, semi-implicit, iterative algorithm for low Mach number flows [45], in which the momentum equation and a Poisson equation for the hydrodynamic pressure are updated using a fractional-step scheme [46]. In the calculations, density ρ and kinematic viscosity μ are taken to be constant. Spatial derivatives are obtained using second-order central differences on a staggered grid, in which velocity components are located at cell faces, and scalar quantities are located at cell centers. The sharp-spectral forcing proposed by Palmore et al. [47] is applied with cutoff wavenumber $\kappa_c = \pi/\Delta = 3$. CIAO employs the message-passing interface (MPI) standard for parallelization. The equations have been discretized on a grid of 512^3. The forced HIT DNS is initialized with a von Karman–Pao (VKP) spectrum [48], and performed with the restriction that $\kappa_{\max}\eta \geq 1$, where η is the Kolmogorov length scale and $\kappa_{\max}$ is the maximum discrete wavenumber. The maximum Courant–Friedrichs–Lewy (CFL) number is 0.5, and the time step Δt is fixed. Table 1 reports the simulation parameters of the dataset.

A statistically stationary state is obtained for the Re140 case after $t \approx 10\,\tau_L$, where $\tau_L = k/\epsilon$ is the eddy-turnover time, k is the turbulent kinetic energy (TKE), and ϵ is the TKE dissipation rate. After the initial transient, a total of 160 snapshots of the 3D velocity field $[u_i = (U, V, W)^T]$ were extracted every $0.5\,\tau_L$. To obtain LR input data, these instantaneous velocity fields were filtered using explicit filter kernels, such as box, Gaussian, and spectrally sharp filter kernels, of width $\Delta = 8\,\mathrm{dx}$, where dx is the DNS grid spacing. To ensure that the LR F-DNS fields are of the same dimensionality as the corresponding LES fields, a discrete downsampling operation is applied. This downsampling is treated independently from the filter kernel. The training, validation, and testing dataset is composed of HR data, i.e., DNS data, and the corresponding LR data, i.e., (downsampled) F-DNS data, for each kernel filter employed.

Table 1
Simulation parameters of training and testing DNS dataset. N is the number of mesh points, Re_t is the turbulent Reynolds number, l_t/L is the number of integral scales within the computational domain, dx/η is the mesh resolution relative to the Kolmogorov length scale η, κ_{max} is the largest wavenumber represented, and τ_L is the eddy-turnover time is second.

Case	N	Re_λ	Re_t	l_t/L	dx/η	$\kappa_{\max}\eta$	τ_L
Re140	512^3	140	3300	5.26	2.00	1.59	120

2.1. Patch-to-patch training strategy

For training, only 120 randomly selected snapshots were utilized. The remaining 40 snapshots were reserved for validation and testing purposes. A discussion regarding the optimal amount and the fidelity of training data is reported in Appendix. The 120 snapshots of 512^3 HR data and 64^3 LR data of the three-dimensional velocity components comprise a dataset of approximately 360 GB. The patch-to-patch training strategy alleviates the computational demand of loading large datasets into GPU memory. In this strategy, non-overlapping subboxes (SBs) are randomly extracted from each full-size snapshot. Section 6.1 investigates the effect of LR and HR subbox size on the network's predictive capabilities versus optimal computational utilization. During the SR-GAN training, the following LR SBs sizes are considered $n_{\mathrm{SB,LR}} \in [4, 6, 8, 12, 16, 32, 48]$, where $n_{\mathrm{SB,LR}}$ is number of mesh points per SB in each direction. The corresponding target $n_{\mathrm{SB,HR}}$ size is uniquely determined by the fixed upsampling factor.

Notably, during the *a priori* evaluation of test samples, no patch-based approach is used. Instead, the entire test sample domain is reconstructed continuously without relying on cropped SBs as during the training. This continuous reconstruction avoids potential artifacts associated with discontinuities between SBs as reported in Section 7.1. It is made possible by the larger memory pool available for CPU-based inference. Moreover, the 40 validation/testing snapshots were chosen to ensure statistical independence between the two sets of samples. The time separation between training/validation samples and testing samples spans more than $3\,\tau_L$, therefore the two sets of samples are expected to be statistically uncorrelated. Moreover, to optimize the network's performance, the velocity components of both high-resolution and low-resolution data used for training and testing were normalized using the global maximum and minimum values of the DNS data.

3. Generative adversarial network and loss function definition

The SR-GAN employed in this work is shown in Fig. 1 which is based on our previous work, where the generator was adapted to include additional upsampling and dense layers given the employed DNS-to-F-DNS ratio [24]. This configuration is adapted for small-scale turbulence reconstruction from the original ESRGAN framework for image reconstruction [49]. The performance of this framework in reconstructing turbulent flow fields has been thoroughly assessed in comparison to a standard supervised CNN-based framework [16]. Comparisons with some SR-GAN frameworks available in the literature were reported in [16], while comprehensive assessments of other SR-GAN frameworks are currently ongoing and beyond the scope of this work.

Our SR-GAN framework employs a conventional GAN framework (generator and discriminator) for training. The generator is a CNN derived from the original SRResNet [50]. This framework captures small-scale features by incorporating skip connections. The generator employs three-dimensional convolutional layers and leaky rectified linear unit (LReLU) activation functions. LReLUs are chosen for their enhanced computational efficiency compared to the standard rectified linear unit (ReLU) [51].

The residual-in-residual dense block (RRDB) is a key component of the framework, featuring residual connections and a series of densely

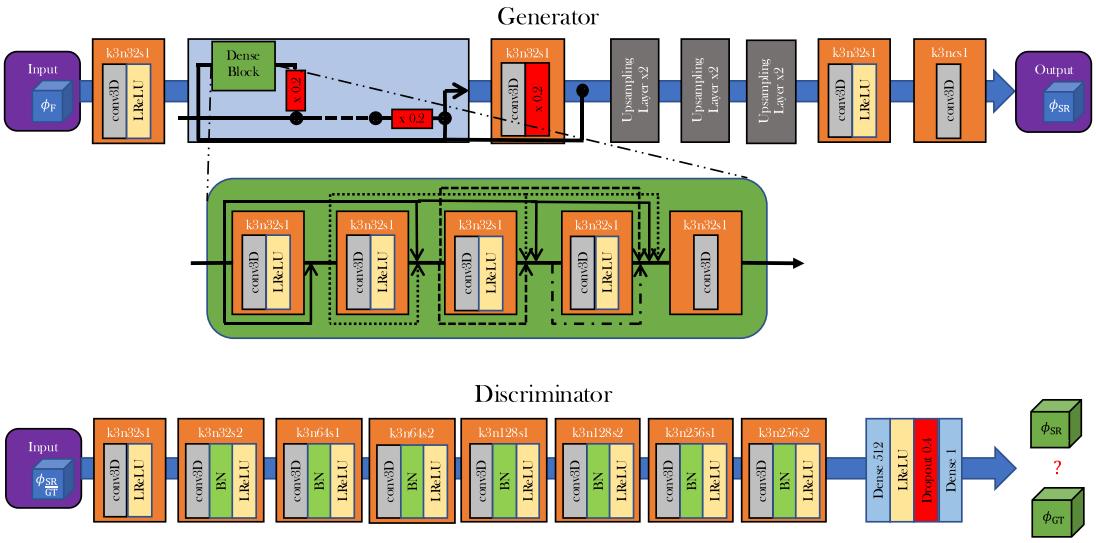

Fig. 1. The generator (above) and discriminator (below) structures employed by the SR-GAN framework employed in this work. In each, ϕ_F is the filtered input field, ϕ_{SR} is the super-resolved field, and ϕ_{GT} is the ground-truth (GT), i.e., DNS field. Each convolutional block contains kernels of size k, n filter maps, and s strides along each spatial dimension of the convolutional layer.

connected layer blocks (three in this context). This structure incorporates a residual-in-residual design [49]. Together, these elements enable the generation of super-resolved data through a deep network. This depth is crucial for learning complex transformations and represents a current state-of-the-art feature in super-resolution networks [52]. The generator further incorporates three upsampling layers, each increasing the input's spatial resolution by a factor of two in each spatial dimension. This is achieved through nearest-neighbor interpolation, replicating adjacent grid points, and a convolution layer to enhance the operation. The overall number of trainable parameters in the generator model is approximately 3 million. The discriminator (Fig. 1) is structured as a deep deconvolutional framework comprising fully connected layers, including convolutional layers and LReLU activation similar to the generator, with a binary classification output. The total count of trainable parameters for the discriminator is approximately 6 million.

The generator's loss function ($\mathcal{L}_{\text{GEN}}$), already employed in the previous work of Bode et al. [23], is defined as a linear combination of pixel loss (L_{pixel}), pixel gradient loss (L_{gradient}), continuity loss ($L_{\text{continuity}}$), and the contribution of the adversarial (discriminator) loss ($L_{\text{adversarial}}$) [53],

$$\begin{aligned}
\mathcal{L}_{\text{GEN}} &= \beta_1\, L_{\text{pixel}} + \beta_2\, L_{\text{gradient}} + \beta_3\, L_{\text{continuity}} + \beta_4\, L_{\text{adversarial}}\,,\\
L_{\text{pixel}} &= \text{MSE}(\phi_{\text{SR}}, \phi_{\text{DNS}})\,,\\
L_{\text{gradient}} &= \text{MSE}(\nabla\phi_{\text{SR}}, \nabla\phi_{\text{DNS}})\,,\\
L_{\text{continuity}} &= \text{MSE}(\nabla\cdot\phi_{\text{SR}}, 0)\,,\\
L_{\text{adversarial}} &= -\mathbb{E}[\log(\sigma(D(G(\phi_{\text{F}})) - \mathbb{E}[D(\phi_{\text{DNS}})]))]\\
&\quad - \mathbb{E}[\log(1-\sigma(D(\phi_{\text{DNS}}) - \mathbb{E}[D(G(\phi_{\text{F}}))]))].
\end{aligned} \tag{2}$$

The coefficients $\beta = [0.89, 0.06, 0.05, 6\cdot 10^{-5}]$ were chosen through hyperparameter tuning, such that the absolute value of each term in $\mathcal{L}_{\text{GEN}}$ is of the same order. It is important to note that these β parameters may not be universally applicable and could be case-dependent. The ϕ_{SR} indicates the super-resolved field, while the ϕ_{DNS} is the ground-truth (GT), i.e., the DNS field. The mean-squared error (MSE) is computed between the reconstructed and GT fields and is applied separately to all elements when tensor quantities are considered. The operator $\mathbb{E}[\cdot]$ represents the mathematical expectation, $\sigma(\cdot)$ denotes the sigmoid function, and $D(\phi_{\text{DNS}})$ and $G(\phi_{\text{F}})$ refer to the outputs of the discriminator and generator, respectively. The initial component of the adversarial loss function prompts the discriminator to accurately categorize HR fields as real, whereas the subsequent term motivates the generator to generate SR fields capable of deceiving the discriminator into categorizing them as genuine HR [53]. The L_{pixel} loss function is inherently dimensionless because both the input and output fields are normalized beforehand. The L_{gradient} and $L_{\text{continuity}}$ loss functions are normalized using the Kolmogorov length scale η. This normalization ensures that the choice of grid spacing or velocity magnitude does not affect the loss functions, enhancing the generalizability and scalability of the nondimensional β parameters across comparable setups.

The discriminator's loss function is given by

$$\begin{aligned}
\mathcal{L}_{\text{DISC}} &= \mathbb{E}[\log(\sigma(D(\phi_{\text{DNS}}) - \mathbb{E}[D(G(\phi_{\text{F}}))]))]\\
&\quad + \mathbb{E}[\log(1-\sigma(D(G(\phi_{\text{F}})) - \mathbb{E}[D(\phi_{\text{DNS}})]))],
\end{aligned} \tag{3}$$

which is based on the relativistic average GAN loss function proposed by Jolicoeur et al. [53].

4. Parallelization of the training process

Training SR-GAN frameworks designed for turbulence modeling on a single GPU is challenging due to high computational demands and memory requirements from the framework itself, compounded by large dataset sizes. Distributed training, especially via data parallelism, is preferred for its scalability and efficiency in handling large datasets [54]. In the training framework employed in this work, each GPU stores clones of the generator and the discriminator. Each clone handles a separate dataset portion, as illustrated in Fig. 2. Allocating both the generator and discriminator on the same GPU enhances communication efficiency during training, as frequent communication between these components is required for adversarial learning.

During the distributed training process, the entire dataset is initially divided into portions equally sized (sharded training datasets) based on available workers and preloaded into memory for fast access. Each worker trains its copy of the SR-GAN model exclusively on the training sub-dataset allocated to it. Synchronous data parallelism, chosen for ease of implementation, synchronizes the forward pass across in time all workers, resulting in different loss outputs and gradients because each worker operates on distinct training sub-datasets. Once all workers have computed their gradients, these gradients are communicated using the Horovod library [55]. Horovod is an open-access decentralized framework, with bandwidth-optimal communication protocols over the MPI for worker communication and NVIDIA collective communications libraries (NCCLs), where workers (GPUs) exchange parameters without the need for a parameter server. This enables fast gradient aggregation and averaging. Once gradients have been synchronized and updated,

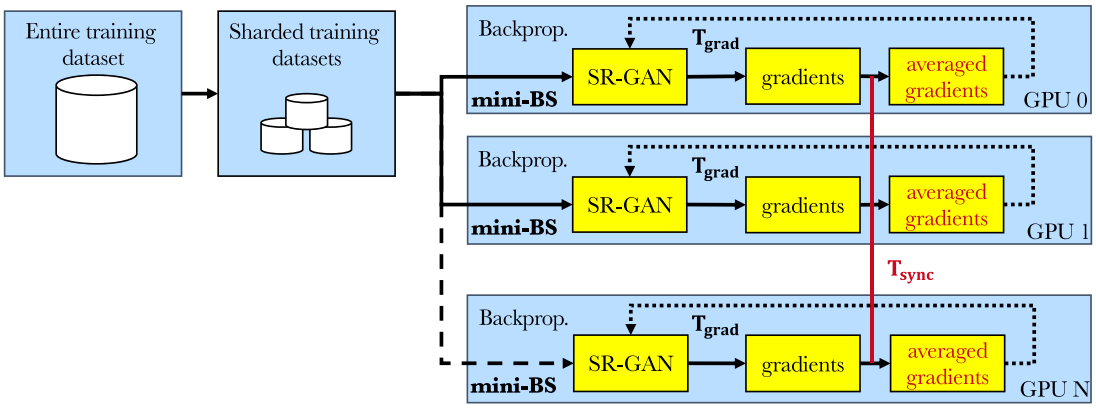

Fig. 2. Synchronous data-parallelization strategy based on the Horovod library [55]. The mini-batch size (mini-BS) refers to the number of fields processed in a single iteration by a worker (GPU) during training. The global batch size refers to the total number of fields considered in each optimization iteration across all workers in a distributed training framework. T_{grad} indicates the execution time to compute the local gradient estimate, while T_{sync} refers to the execution time required to average the gradient estimations and synchronize model parameters across all workers (GPUs).

each worker independently conducts backpropagation to update its model weights for subsequent forward passes. Synchronous data parallelism ensures that all workers are training on a consistent model state (i.e., same model's weights) and contributes to the overall convergence of the model during training.

The Horovod library exhibits exceptional scalability performance. However, a central challenge remains in the optimization of the neural networks themselves. Most optimization methods are variations of stochastic gradient descent (SDG) that approximate gradients within a batch of fields extracted from the entire training dataset. As training is distributed across numerous workers, the global BS increases, necessitating the parallelization of the SDG operation. Consequently, the total amount of training data processed increases with the number of workers, thereby accelerating training and enhancing gradient update effectiveness. These local batches per worker are termed mini-batch sizes (mini-BSs).

The most common synchronous data-parallelism distributed training approach, and specifically considered in this work, is to maintain a fixed mini-BS per worker as in the single-training approach. Thus, the global BS scales with the number of workers employed. This approach is preferred in the literature not only to improve GPU utilization [30] but also to reduce the number of parameter updates, thus increasing scalability [56]. Following Goyal et al. [30], a linear scaling rule is applied that increases the learning rate proportionally to mini-BS increments. This enhances computational performance and enables larger gradient-descent steps, which can accelerate convergence. However, increasing the global BS might have a considerable impact on the learning and generalization capabilities of the model, as SDG optimizers require a certain amount of noise produced by the rather small sizes of mini-BS [57]. This is particularly relevant for SR-GANs where adversarial training plays a crucial role in enhancing image super-resolution capabilities.

Experiments using various mini-BS and inter-GPU communicators, different combinations of LR and HR training subboxes, and effective approaches to accelerate the training convergence across multiple computing nodes reported in Section 6 were conducted using the DEEP-ESB partition[1] of the Jülich supercomputing center (JSC). This partition is tailored for intensive applications and code parts with regular control and data structures, with a particular focus on efficient parallel scalability. Details of the hardware are outlined in Table 2. The GPUs employed achieve a peak HBM2 memory bandwidth of 900 GB/s per GPU. The ADAM optimizer is initialized with a learning rate of 10^{-4} for every single GPU training conducted and was selected based on prior successful implementations [24,26]. Single-precision (FP32) training was employed to mitigate potential rounding errors, ensuring numerical stability throughout the computations, as shown in recent studies [16,58].

Table 2
Hardware details of the Jülich supercomputing center's DEEP-ESB partition used for distributed training and interference-coupled LES.

Partition	DEEP-ESB
Total Nodes	75
Network	IB-EDR (100 Gb/s)
CPU Cores/Node	8x Intel Cascade Lake (2.5 GHz)
Accelerators/Node	1x NVIDIA V100 (32 GB)
RAM/Node	48 GB

Due to its adversarial nature, GAN training is challenging and convergence can be inhibited by parameter oscillations and destabilization [20]. Therefore, the generator is pre-trained in a fully supervised manner, utilizing only the pixel-loss contribution in Eq. (2), which is similar to training a fully-supervised SR-CNN. This initial pre-training phase is conducted for a sufficient number of epochs, after which the loss function and statistics computed on the reconstructed field do not change substantially. Hence, the pre-training is considered converged. The pre-trained generator serves as a starting point for the subsequent GAN training process, and it is ensured that each training investigation begins with identical generator configuration weights obtained from pre-training. The GAN discriminator is not pre-trained during this phase. Additional details are reported in Appendix.

To improve the SR-GAN framework's performance and generalizability to out-of-sample inputs obtained from implicit (unknown) filter kernels (e.g., to genuine LES data), an additional partially unsupervised training phase is executed [16]. In this phase, both labeled data (F-DNS/DNS fields) and precomputed LES fields (i.e., without the corresponding HR fields) with and without SFS closure were provided as LR training inputs. The discriminator network has been pre-trained to recognize authentic DNS fields and, in this phase, provides feedback to the generator to learn how to accurately reconstruct high-frequency details and small-scale structures similar to DNS. This additional training phase enables the SR-GAN framework to significantly improve its extrapolative capabilities and robustness compared to traditional SR-GAN training approaches for turbulence closure and is particularly relevant for the inference-coupled SR-LES investigations.

[1] https://www.fz-juelich.de/en/ias/jsc/systems/prototype-systems/deep_system

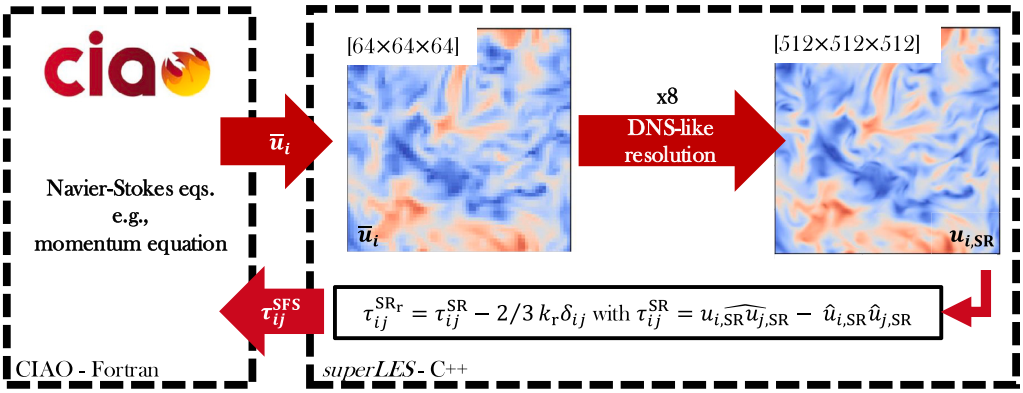

Fig. 3. Qualitative representation of the operations performed by the *superLES* library and its coupling with the CIAO solver.

5. *superLES* library for inference-coupled SR-LES simulations deployment

An SR DL framework has been developed that embeds an in-situ coupling mechanism between CFD solver and SR model through a modular implementation. This framework is used to conduct inference-coupled (*a posteriori*) LES computations. Here, *a posteriori* tests are conducted by coupling CIAO to a novel SR library, called *superLES*, that implements the SR-GAN framework introduced in Section 3. However, *superLES* is designed to be model agnostic, and the various SR-GAN frameworks proposed in the literature can be implemented into *superLES*.

5.1. Hybrid coupling between CIAO CFD solver and superles library

A high-performance hybrid coupling between the CIAO solver and the *superLES* library is developed. As both CIAO and *superLES* frameworks share the same data structure, the interface is designed to take advantage of the computing capabilities of hybrid (heterogeneous) architectures (CPUs-GPUs) of modern clusters, enabling direct MPI communication between CPU and GPU resources. The coupling includes different communication paradigms (shared, split, etc.), which align with the requirements of CFD solvers. These allow for data transfer and processing between massively parallel CFD solvers and distributed DL inference. The *superLES* provides interfaces in FORTRAN 2003 and C++ to be accessible to a wide variety of CFD solvers and DL frameworks such as TensorFlow [51] and PyTorch [59], using the open neural network exchange (ONNX) format and runtime environment [60]. MPI communication management, function wrappers, and user-defined inputs (e.g., weight configuration, upsampling ratio, etc.) are included. Moreover, the library is model-agnostic, meaning that the settings of neural network models and simulations are passed directly from the input file. This allows for a high degree of flexibility in experimenting with different neural network models without recompiling the CFD code.

As sketched in Fig. 3, at every CIAO iteration, the resolved three-dimensional velocity components are transferred to the *superLES* library. Ghost cells (or halo regions) and boundary information are also included. The domain decomposition and resources employed determine which MPI rank the information is sent to. At this stage, the three-dimensional velocity components are stored in the system memory. After the inference performed by the *superLES* library on GPUs (described in Section 5.2), each component of the τ_{ij}^{SR} tensor is explicitly offloaded from the GPUs to the CPUs and transferred back to the CFD solver.

5.2. SR inference and Reynolds stress tensor computation

Based on the separation of concerns principle of software engineering, the DL inference performed using the *superLES* library is encapsulated in a generic C++ library. This library is based on the well-known machine learning framework TensorFlow [51], ensuring modularization and abstraction of DL processes, thereby enhancing maintainability and scalability. Moreover, the library supports the decoupling of various DL model structures and different computational grid sizes. This is fundamental as it allows predictions with arbitrary DL model shapes. These can be tailored to the memory configuration of the available hardware platform without constraining simulation computational domain sizes.

At every time step, a LES (LR) field, originally stored in the CIAO code, is transferred to the *superLES* library. The field is normalized using the same normalization algorithm and values used for the training. The generator configuration weights of the SR-GAN model are stored in the native format of ONNX. This ecosystem establishes open standards for implementing DL applications and software tools. After loading the generator model structure and its configuration weights, the LES field is super-resolved up to the SR size, defined by the number of upsampling layers inside the SR-GAN model. In this work, the mesh size is increased by a factor of eight in each direction. The SR field, $u_{i,SR}$, is stored in an auxiliary, finer computational grid defined only by the C++ environments and is denormalized as described in Section 2. The Reynolds stress tensor based on the SR velocity field, τ_{ij}^{SR}, is evaluated as

$$\tau_{ij}^{SR} = u_{i,SR}\widehat{\odot u}_{j,SR} - \hat{u}_{i,SR} \odot \hat{u}_{j,SR} \tag{4}$$

where the $\hat{\cdot}$ operator indicates a combination of box-filtering and downsampling that is consistent with the upsampling factor. This filtering operation differs from the classical dynamic approach proposed by Germano et al. [4], as it involves both filtering and downsampling of the SR field to match the original resolution of the LES. The $\odot$ denotes element-by-element multiplication, performed with MPI/OpenMP for CPU-based platforms and MPI/CUDA (for NVIDIA GPU only) for hybrid CPU–GPU cluster architectures. This process is described in more detail in Section 5.3. The same applies to the filtering and algebra operations. Following the computation of the τ_{ij}^{SR} tensor, the residual kinetic energy, k_r, is evaluated, and the trace-free τ_{ij}^{r} tensor is returned to the CFD solver. The source term, $\partial\tau_{ij}^{SR}/\partial x_i$, is computed directly by the CIAO solver to preserve the numerical accuracy of gradient calculations.

5.3. Hybrid parallelization for heterogeneous cluster architectures

The multiple programs–multiple data implementation enables execution of the coupling of the CIAO solver to SR inference with *superLES* in a parallel environment. The coupling with the ONNX runtime environment allows SR model inference to be executed using either CPUs or

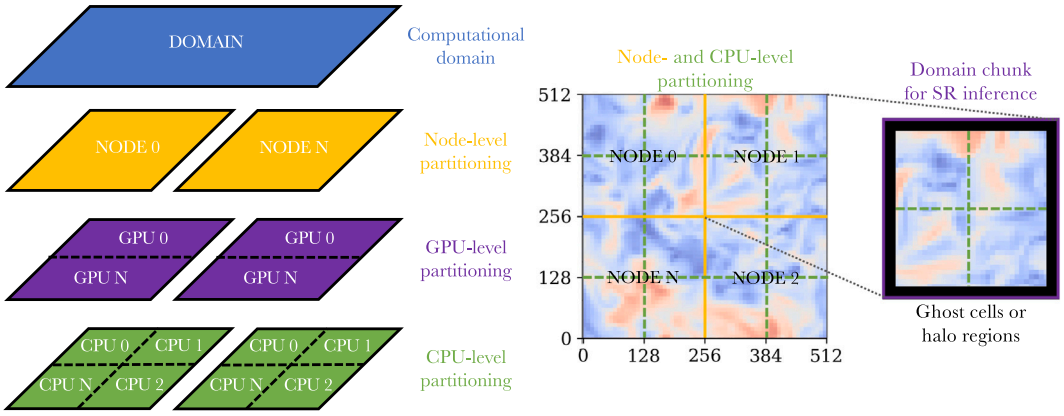

Fig. 4. Hierarchical domain decomposition between CIAO solver (CPU-level partitioning) and the *superLES library* library (GPU-level partitioning) (left) and qualitative representation of the SR domain chunk per node, highlighting the ghost cells (or halo regions) employed (right).

GPUs, depending on the allocated resources. The hierarchical domain decomposition between the CIAO solver (CPU-level partitioning) and the *superLES* library (GPU-level partitioning) is shown in Fig. 4.

The *superLES* library is developed to be easily integrated into various parallel CFD solvers. Two operation modes are supported according to the users' hardware configuration. These are referred to as *shared* and *split*. In the *shared* mode, the *superLES* library is integrated into the CFD code operation space, e.g., the same node(s), where the SR inference is performed. This mode is particularly suitable for heterogeneous architectures in which computing nodes include GPUs with ONNX runtime support. In this way, the CFD solver executes on the CPUs while the *superLES* library operates on the corresponding GPUs available to the same node. As both tasks happen within the same rank, the bottleneck introduced by the data transfer is reduced. This mode is employed for the inference-coupled SR-LES investigations carried out in Section 7.

On the other hand, *split* mode divides the MPI communicator ranks into two halves: one for the main CFD code and the other for the DL inference. This mode suits modular supercomputing architectures like cluster/booster frameworks, where clusters have powerful CPUs and boosters have dedicated accelerators. The main CFD code runs on cluster nodes, while DL inference operates on corresponding booster nodes. This mode also offers two process affinities: scatter and compact. Scatter mode overlaps the execution of native CFD and DL inference on the same booster node (also equipped with CPUs), useful for parallel execution. Compact mode allows executing CFD code on one subset of nodes and DL inference on another, accommodating different hardware architectures within the same parallel environment. These modes provide flexibility for optimizing performance based on hardware and computational needs.

5.4. Domain decomposition

The computational complexity associated with SR methods necessitates the decomposition of the LES velocity field into chunks. Decomposition enables the parallel computation on each chunk, leveraging available computing resources to accelerate the solution process. The *superLES* library can operate in two modes: *single* mode maintains a one-to-one correspondence between CFD-domain decomposition and SR-domain decomposition, ensuring identical decomposition between CPUs and GPUs. Conversely, in *aggregate* mode shown in Fig. 4, CPU-level subdomains are merged and sent to the GPU, allowing for flexibility in the number of CPUs and GPUs utilized. This mode is employed for the inference-coupled SR-LES investigations carried out in Section 7.

Moreover, padding and stride operations during convolutional inference can significantly affect smoothness requirements, potentially generating spurious solutions at the boundaries between SR domain chunks across diverse computational resources. The size of ghost cells, known as halo regions and depicted as an additional black border surrounding a computational domain in Fig. 4 (right), is crucial in SR-GAN applications. These halo regions serve as tunable parameters to mitigate boundary effects, ensuring smoother and more accurate reconstructions.

6. Performance analysis of distributed training approach

The computational performance of the distributed training approach is investigated next. The Re140 dataset is used for the analysis. The evaluation focuses on three key indicators: (1) *physical accuracy*, which quantifies computational requirements for obtaining physically meaningful predictions, (2) *hardware efficiency*, which measures the raw number of training SBs processed per wall-clock time by the GPUs, and (3) *processing efficiency*, that assesses the effective utilization of parallel processing to accelerate convergence across distributed computing nodes.

6.1. Physical accuracy

Physically meaningful predictions can only be obtained if the model learns effectively during the training. In the context of large computational domains, patch-to-patch training is commonly adopted, which may lead to variations in model performance. Table 3 summarizes the different combinations of LR and HR SBs used for SR-GAN training given the GPUs' memory limitation. Depending on the input LR SBs size, the corresponding target HR SBs size is uniquely determined by the fixed upsampling factor. The size of the LR/HR pair l_{SB} is non-dimensionalized using the Kolmogorov length scale η. Due to network framework, it is usually advantageous to scale the maximum mini-BS as a power of two. Moreover, with decreasing SB size, the total number of SBs extracted over the entire computational domain increases.

Each LR/HR pair configuration represented in the table is trained individually on a single GPU. This choice follows insights from Section 6.3, which suggest that using multiple GPUs for training might negatively affect predictive performance [61]. Therefore, hyperparameter tuning for learning rates and implementing an adaptive mini-BS scheduler are considered to mitigate any influence from specific training parameters [57,62]. Training iterations continue until no further improvement is observed in the averaged pointwise cross-correlation between τ_{12}^{SR} and τ_{12}^{DNS}, referenced to $\langle\tau_{12}^{SR}\rangle$.

Fig. 5 presents the $\langle\tau_{12}^{SR}\rangle$, alongside the root-mean-squared error (RMSE) of the reconstructed velocity field, as assessed in-sample on the same testing dataset with the network trained on the various LR/HR SBs pairs detailed in Table 3. The use of a consistent box filter kernel and size between training and testing samples is maintained throughout. Both metrics exhibit an asymptotic behavior. When the LR/HR

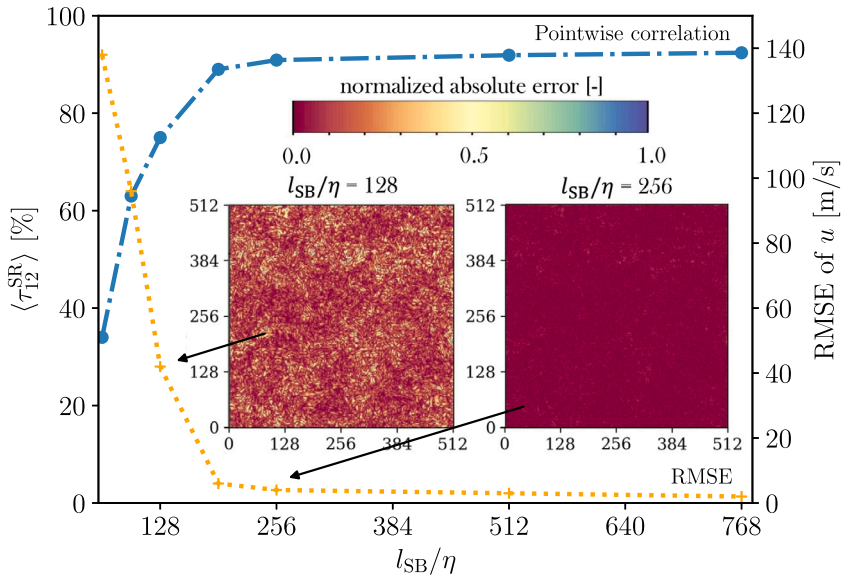

Fig. 5. Averaged pointwise cross-correlation between τ_{12}^{SR} and τ_{12}^{DNS}, indicated as $\langle \tau_{12}^{SR} \rangle$ (blue line), and RMSE of the reconstructed velocity field (orange line) for each LR/HR SBs pairs (referred to as l_{SB}/η) indicated in Table 3 evaluated on the testing dataset. 2D centerline slices of the absolute error of the SR fields versus DNS are shown in the center. The normalized absolute error is computed as $\hat{E} = E/\max(E_{DNS,SR_{128}}, E_{DNS,SR_{256}})$, where $E = \left|u_{DNS} - u_{SR_{128,256}}\right|$, and the subscripts 128 and 256 indicate the corresponding l_{SB}/η ratio.

Table 3
Overview of patch-to-patch training parameters used for training: HR and LR SBs training sizes, number of Kolmogorov length scales η per l_{SB} (l_{SB}/η), number of SBs extracted from the entire dataset, maximum mini-BS (multiple of 2), and time per epoch for training on a single GPU. OOM refers to out-of-memory. The configurations considered for the analyses conducted in Section 6.3 are highlighted in bold.

$N_{SB,HR}$	$N_{SB,LR}$	l_{SB}/η	# of SB	Max mini-BS	Time/epoch [mins]
512^3	64^3	1024	120	*OOM*	N/A
384^3	**48^3**	**768**	**284**	**1**	**75.87**
256^3	**32^3**	**512**	**960**	**2**	**76.32**
128^3	**16^3**	**256**	**7680**	**16**	**77.58**
96^3	**12^3**	**192**	**18 204**	**64**	**78.06**
64^3	8^3	128	61 440	128	79.83
48^3	6^3	96	145 635	512	81.63
32^3	4^3	64	491 420	1024	82.03

SBs encompass at least roughly 192η, the RMSE of the reconstructed velocity field diminishes, and consequently, higher $\langle \tau_{12}^{SR} \rangle$ are achieved. Conversely, as the SB size decreases below 192η, the reconstructed field deviates notably from the DNS field. This suggests that the model's performance varies significantly based on the LR and HR SB combinations used for training, contingent upon the number of η contained in each training SB. This qualitative trend is further demonstrated through centerline slices of the normalized absolute error, demonstrating that errors are introduced predominantly at the small scales, as shown in Fig. 5 (center).

To further understand the impact of training SB size on the model's learning capability, Fig. 6 compares the instantaneous TKE spectra (left) and the probability density function (PDF) of the normalized velocity gradients (right) for the F-DNS, DNS, and SR fields. The comparison is performed using two of the previous training configurations, specifically $l_{SB}/\eta = 128$ and $l_{SB}/\eta = 256$. Notably, the model trained with SBs smaller than 192η fails in recovering the correct TKE. This leads to significant deviations in the SFS ranges, resulting in both over- and under-prediction. This is also seen for turbulence intermittency, through the PDF of the normalized velocity gradients, corroborating the findings. Smaller LR/HR SB pairs tend to underestimate large negative gradients, as shown in the inset of Fig. 6 (right), and slightly overestimate large positive gradients, while larger LR/HR SB pairs exhibit only marginal deviations from the DNS solution. It is important to note that the threshold value of $l_{SB}/\eta \approx 192$ may be specific to this particular configuration, characterized by Reynolds number, and filter size, and it should not be assumed to hold universally without further verification across different conditions.

This demonstrates that varying the input and output patch sizes exclusively results in significant differences in predictive accuracy and, consequently, in the model's learning capability. These differences occur despite a consistent training dataset and equal filter kernel and size between training and testing datasets. Larger patch sizes prove advantageous for training deep SR-GAN frameworks as they enable a larger receptive field. This increased field size allows the network to capture a broader range of scales, crucial for effectively modeling both large and small-scale structures. When the integral length scale l_t is considered as a reference length scale per SB, accurate predictions require training SB to encompass at least one l_t, which may also not be universal. Additionally, larger patch sizes might affect the resilience of the network to overfit, resulting in more robust and generated fields with less distortion. Additionally, given the significant influence of the discriminator on SFS structure reconstruction [16], larger output patch sizes might not only enhance the generator's performance but also enables the discriminator to better evaluate the generated field, providing more accurate feedback to the generator, which leads to a higher correlation between the reconstructed fields and the ground truth.

6.2. Hardware efficiency

To enhance the model's learning capability, careful attention must be given to the LR/HR training SBs configuration. Equally important in a distributed training approach is hardware efficiency, which is primarily influenced by GPU architectures, memory bandwidth, and inter-GPU communication. In synchronous data parallelism, an iteration during the training runtime involves two main operations: computing the local gradient estimate (T_{grad}) and averaging the gradient estimation and synchronizing model parameters across all GPUs (T_{sync}), as depicted in Fig. 2.

The hardware efficiency is investigated by using a LR/HR pair of training SBs of 16^3 (input) and 128^3 (output), respectively (compare Table 3). Fig. 7 (left) shows the number of training SBs processed per minute with a growing mini-BS as a function of the number of GPUs. It can be seen that maximizing mini-BS enhances system throughput by

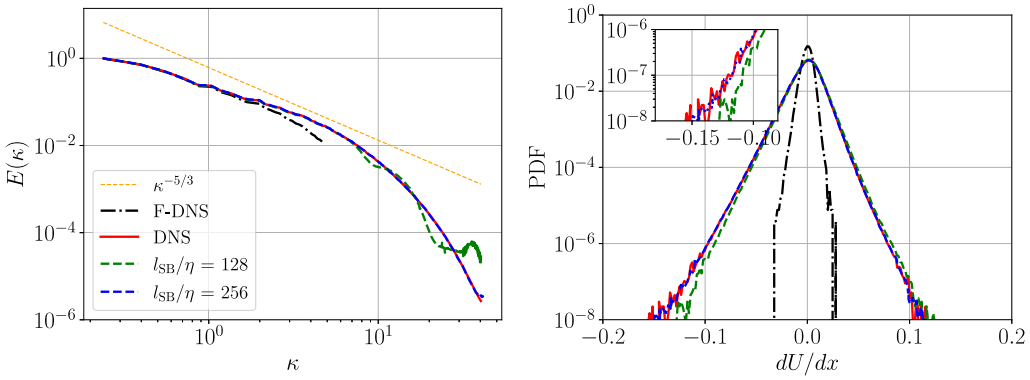

Fig. 6. In-sample instantaneous TKE spectra (right) and PDFs of the normalized velocity gradients (left) for two different LR/HR training SB configurations, specifically for l_{SB}/η = 128 and l_{SB}/η = 256. The orange line represents Kolmogorov's −5/3 power law. The inset highlights the models' performance in reconstructing the large negative gradients.

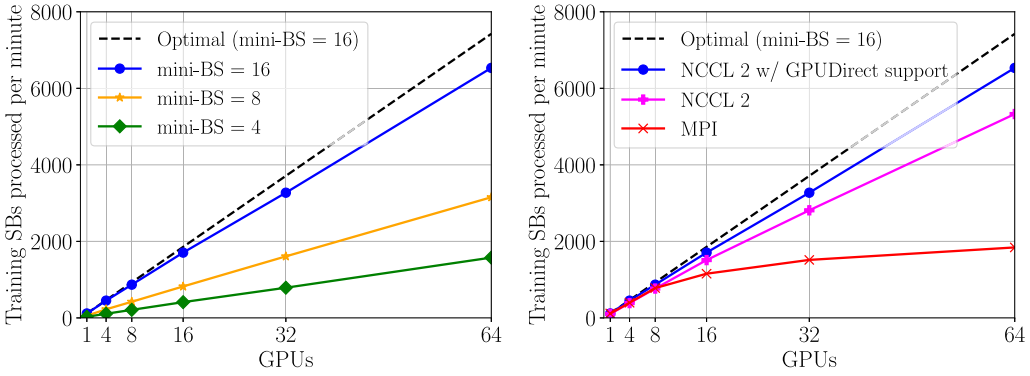

Fig. 7. Number of training SBs processed per minute as a function of growing mini-BS (left) and using various inter-GPU collective communicators (right) as the number of utilized GPUs increases. The mini-BS is fixed to 16 for the communicators' variation.

more effective usage of computational resources. This is explained by the fact that T_{grad} depends on many factors, such as input/output patch size, network size, GPU performance, and number of training fields, while T_{sync} is influenced by gradient size, network performance (bandwidth), and is typically shorter when the GPUs are co-located within the same physical node. As the mini-BS increases, T_{grad} also increases due to the higher number of fields processed for each iteration's runtime, while T_{sync} is typically independent on the mini-BS. Consequently, each iteration's runtime is constrained by the T_{sync}. Maximizing the mini-BS improves scalability yielding a higher T_{grad}/T_{sync} ratio. A higher ratio indicates that a larger proportion of the iteration's runtime is dedicated to gradient computation, rather than waiting for synchronization, meaning that computational resources are more efficiently utilized, with less idle time spent waiting for synchronization across GPUs.

Minimizing T_{sync} is crucial for ensuring optimal scalability, as it increases with the number of GPUs. This can be achieved by optimal topology-aware inter-GPU communication primitives seamlessly integrated into applications. Fig. 7 (right) depicts the number of SBs processed per minute for various GPU-node connections using a fixed mini-BS of 16 and different collective communicators. CPUs are employed with MPI communication, while GPUs are employed with NCCL 2 and NCCL 2 with GPUDirect RDMA in the Horovod library. The GPUDirect tool with RDMA communicators enables direct reading and writing to/from GPU memory, thereby reducing redundant memory copies, CPU overheads, and latency. This leads to a significant performance uplift, especially with multiple GPUs across several nodes, as is demonstrated by the nearly linear scalability of up to 64 GPUs. Conversely, using the MPI standard leads to substantial performance drops compared to NCCL-based setups. In this case, the speed up notably declines beyond 8 GPUs as T_{sync} nearly approaches T_{grad}.

6.3. Processing efficiency

Maximizing mini-BS and implementing efficient GPU-to-GPU communication protocols have been shown to significantly improve GPU performance and hardware efficiency. On the other hand, based on the analysis performed in Section 6.1, the LR/HR pair combinations are limited to the ranges $N_{SB,LR} \in [12^3, 16^3, 32^3, 48^3]$ and corresponding $N_{SB,HR} \in [96^3, 128^3, 256^3, 384^3]$, as highlighted in Table 3. These configurations result in a varying number of the training SBs and mini-BSs per GPU (thereby affecting global BSs for distributed training). While models trained with these pairs demonstrate comparable predictive capabilities on a single worker, single-GPU training is undesirable due to its lengthy duration of approximately a week per training run (Table 3). Therefore, it is crucial to evaluate an effective approach to accelerate training convergence across distributed computing nodes without compromising the inherent predictive accuracy. Synchronous data parallelism, as described in Section 4, is investigated here, where the mini-BS per GPU is constant, while the global mini-BS scales with the number of GPUs employed. We employ a linear scaling rule for the learning rate and a warm-up scheme. This distributed training strategy is referred to as scaling.

Fig. 8 (left) shows the training wall time required by the SR-GAN to achieve $\langle \tau_{12}^{SR} \rangle$ above 90% on the testing dataset when employing the scaling approach. This metric ensures that the statistics of the reconstructed fields closely match those presented in Figs. 5 and 6.

All four LR/HR training configurations exhibit comparable training wall times when using up to 8 GPUs. However, performance significantly deteriorates with the addition of more GPUs when employing larger LR/HR configurations, such as $32^3 \rightarrow 256^3$ and $48^3 \rightarrow 384^3$. This results in either divergence, where the generator fails to produce meaningful physical predictions, or longer training times, making the

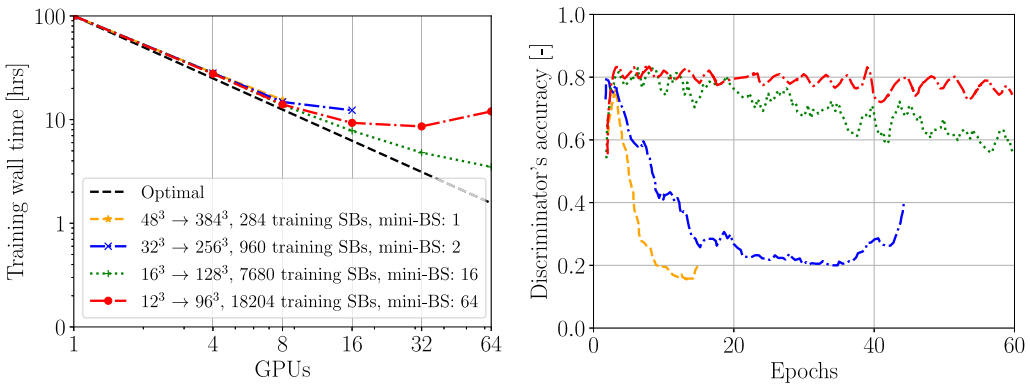

Fig. 8. Left: Training wall time required to achieve the target $\langle \tau_{12}^{SR} \rangle$ above 90%. The bold highlighted LR/HR training SB configurations from Table 3 were utilized. The global batch size is increased corresponding to the number of GPUs, keeping the mini-BS fixed. The linear scaling rule for the generator's and discriminator's learning rate and a warm-up scheme are applied. If no wall time is reported, it indicates training divergence. Right: Normalized validation accuracy of the discriminator across epochs during the training with 32 GPUs. The initial 2 epochs are excluded since the discriminator is not pre-trained.

DTA inefficient. On the other hand, when smaller LR/HR training configurations are employed, for example, $16^3 \rightarrow 128^3$, the training process remains stable and demonstrates almost linear scalability. This is evidenced by the wall training time decreasing nearly in direct proportion to the number of GPUs used. Interestingly, for the smallest LR/HR configuration, i.e., $12^3 \rightarrow 96^3$, utilizing more than 16 GPUs does not yield further reductions in wall training time. In fact, beyond this point, the training time using 64 GPUs is comparable to that using only 8 GPUs, negating any possible acceleration from additional workers.

The size of the LR/HR SBs not only influences the number of SBs extracted from the entire training dataset but also determines the maximum mini-BS. There appears to be a global BS threshold beyond which model quality deteriorates. Utilizing a large number of GPUs implies a large global BS, which, as recent studies have shown, can lead to poor generalization properties and overfitting [57]. This is particularly relevant when the training dataset size is limited compared to a standard image-processing training dataset size, as the model's updates might be too frequent, causing it to overfit to the limited data. Despite the application of the linear scaling and warm-up rules for the learning rate to mitigate this issue, as suggested by recent research [30], managing this balance remains challenging.

Furthermore, the GAN's optimization process depends on the interaction between the generator and discriminator, which is fundamentally different from standard supervised CNN-based SR methods. In the context of distributed training, this interaction might introduce additional complexities, for which is crucial to understand how the failure of one network affects the overall performance and how this interaction behaves under distributed training conditions.

The SR-GAN models' performance is strongly related to how well the discriminator is performing its task of distinguishing real fields from fake ones, referred to as discriminator accuracy. Fig. 8 (right) shows the discriminator accuracy obtained on the validation dataset for the same LR/HR training configurations when 32 GPUs are employed for the training. High discriminator accuracy (close to 1) indicates that the discriminator successfully distinguishes between real and generated images, which consequently puts pressure on the generator to improve the generated field. Conversely, when the discriminator accuracy is too low for too long, the discriminator is not learning effectively, thus providing inaccurate feedback to the generator. Optimal behavior is achieved when the discriminator accuracy slowly diminishes as the number of epochs increases, reaching a value of roughly 0.5. In this scenario, the discriminator has an equal probability of classifying images as real or fake, indicating a balanced but challenging environment for the generator.

For the larger LR/HR configurations, the discriminator accuracy drops dramatically after the first few epochs, leading to a scenario where the discriminator quickly becomes unable to distinguish between real and generated images. This sudden drop suggests that the generator is overwhelming the discriminator early in the training, resulting in the generator producing images that still contain artifacts or lack finer details. In such cases, the limited amount of training data per GPU as well as the small global batch size might cause the discriminator to overfit to the training dataset, leading to memorization rather than generalization on the training dataset. Consequently, the discriminator provides unhelpful and unrealistic feedback to the generator, causing the generator training to diverge.

With the smaller LR/HR configurations, the discriminator accuracy is initially very high as it provides feedback to the generator. Accuracy slowly diminishes over the training, meaning that the discriminator maintains its ability to distinguish real from generated images for a longer period (epochs). This gradual decline indicates a more balanced training process where both the generator and discriminator are improving concurrently. The slower reduction in accuracy allows the generator to learn more effectively from the feedback provided by the discriminator, resulting in higher-quality generated fields over the training. In this scenario, the larger number of training fields per GPU leads to more diverse training data, helping the discriminator generalize better. However, when the global batch size exceeds a certain threshold, such as in the $12^3 \rightarrow 96^3$ configuration, the discriminator's learning capacity diminishes, as indicated by a four times smaller mean discriminator accuracy slope compared to the $16^3 \rightarrow 128^3$ LR/HR configuration, which might be related to the optimizer employed.

The analysis suggests that the distributed training approach is significantly influenced by the number of GPUs employed, even with sufficiently large LR/HR training configurations to ensure physical accuracy, maximized mini-BS and inter-GPU connections to enhance hardware performance. This is due to the indirect impact on the stochastic optimization, affected by both mini-BS and number of training SBs variations. It is evident that there is a coupled interaction between mini-BS, learning rate, and number of training SBs. Additionally, in the context of SR-GAN distributed training, these three linked optimization parameters not only influence generator learning but also significantly impact the discriminator, thereby affecting overall generator performance. It is therefore particularly relevant to consider these factors when employing the synchronous data-parallelism distributed training approach to SR-GANs.

7. Performance analysis for in-sample *a posteriori* LES

Integrating SR-GAN inference into a high-performance CFD solver presents challenges including efficient data exchange, synchronizing the model across multiple ranks, and ensuring parallel scalability. We examine the computational performance and predictive accuracy of the coupled CIAO solver and *superLES* library for LES of forced HIT with

Table 4
Influence of input chunk size N_{in} on *a priori* SR-GAN inference. Bold input sizes are used for subsequent inference-coupled LES.

N_{in}	N_{out}	l_{SB}/η	RMSE	$\langle\tau_{12}^{SR}\rangle$
64^3	**512^3**	**1024**	**2.38**	**92.2%**
32^3	**256^3**	**512**	**2.98**	**91.9%**
16^3	**128^3**	**256**	**3.14**	**91.1%**
8^3	64^3	128	36.76	64.3%
4^3	32^3	64	138.49	32.7%

$\Delta = 8\,\mathrm{dx}$ (LES mesh size $N = 64^3$), initialized with the VKP spectrum, and using spectrally sharp forcing. The *superLES* library embeds the SR-GAN generator pretrained using $l_{SB}/\eta = 256$ and 64 GPUs, given that this configuration exhibits the best trade-off in accuracy, hardware, and processing efficiency (Section 6).

7.1. Influence of domain and halo size on a posteriori accuracy

Optimal performance of the *superLES* framework is obtained in *aggregate-shared* mode (Section 5), for which the input GPU chunk size decreases with increasing compute node count. This influences the model's predictive accuracy by reducing the number of SBs available for inference.

Table 4 reports the *a priori* in-sample RMSE and $\langle\tau_{12}^{SR}\rangle$ of the reconstructed field for different input chunk sizes, N_{in}, using the same SR-GAN model. As shown in the table, accurate predictions require a certain domain size — input domains smaller than this (8^3 and 4^3) cause the SR-GAN to produce inaccurate SR fields. This trend is corroborated *a posteriori* in Fig. 10, which presents the averaged TKE spectra (left) and the PDF of normalized velocity gradients (right). An input chunk size of 4^3 results in significant accuracy degradation compared to a chunk size of 16^3, as the SR-GAN model effectively behaves as though no SFS modeling is applied. Additionally, the SR-GAN model tends to overestimate the occurrence of large positive and negative gradients. Conversely, when the input chunk size is 16^3, the SR-GAN model yields good predictions, with both the TKE and velocity gradient PDFs aligning more closely with the F-DNS. This highlights the SR-GAN model's sensitivity to input chunk size, confirming its critical role in maintaining accuracy which is consistent with the *a priori* findings. However, since the parallel decomposition sets the per-GPU SR subdomain size (for fixed LES domain size and upsampling ratio), the size of the SR input domain (input chunk size) sets an upper bound on the number of GPUs that may be employed for a given LES domain size.

The size of the halo regions, n_{halo}, needed for domain decomposition also influences reconstruction accuracy. Table 5 shows the RMSE of reconstructed velocity fields for different halo sizes, normalized by the corresponding RMSE without halo regions (no domain decomposition), for the $N_{in} = 16^3$ and $N_{in} = 32^3$ cases. (The $N_{in} = 64^3$ case does not use domain decomposition.) Larger domains have fewer halo regions and generally give better results: for each halo size, the $N_{in} = 32^3$ domains have smaller reconstruction errors than $N_{in} = 16^3$. Similarly, larger halo regions generally improve predictive accuracy, with the greatest accuracy achieved when fully half of the input chunk size is provided as a halo region—which of course significantly increases reconstruction cost.

Fig. 9 shows 2D slices of the absolute error of the SR fields reconstructed using the default halo size of the CIAO solver, $n_{halo} = 2$, and the best balance for SR inference $n_{halo} = 8$. The minimum input chunk size $N_{in} = 16^3$ is employed to maximize the number of GPU-to-GPU chunk boundaries in the domain. When $n_{halo}/n_{in} = 1/8$ ($n_{halo} = 2$) is employed, discontinuities appear at the chunk boundaries, which significantly reduce the accuracy of the SR reconstruction. Conversely, using the larger $n_{halo}/n_{in} = 1/2$ ($n_{halo} = 8$) virtually eliminates these interface errors. This behavior is further highlighted in Fig. 10, where

Table 5
RMSE of reconstructed velocity fields for different halo sizes, normalized by the error for continuous reconstruction (no halos). The directional number of input points is n_{in}, such that $N_{in} = n_{in}^3$, and n_{halo} is the number of points per halo region. For example, for $n_{in} = 32$, a normalized halo of 1/16 corresponds to $n_{halo} = 2$.

N_{in} \ n_{halo}/n_{in}	1/16	1/8	1/4	1/2
32^3	1.77	1.52	1.09	1.00
16^3	N/A	2.63	2.00	1.00

Table 6
Resources employed for strong scaling tests on DEEP-ESB.

# nodes	# CPUs	CPU chunk size	# GPUs	Input GPU chunk size
1	8	32^3	1	64^3
2	16	$32 \times 32 \times 16$	2	$64 \times 64 \times 32$
4	32	$16 \times 16 \times 32$	4	$32 \times 32 \times 64$
8	64	16^3	8	32^3
16	128	$8 \times 16 \times 16$	16	$16 \times 32 \times 32$
32	256	$8 \times 8 \times 16$	32	$16 \times 16 \times 32$
64	512	8^3	64	16^3

a smaller halo region significantly diminishes predictive capabilities. Specifically, adopting a ratio of $n_{halo}/n_{in} = 1/8$ leads to more dissipative dynamics compared to $n_{halo}/n_{in} = 1/2$. This is evident in the stronger decay of TKE relative to the F-DNS and the reduced frequency of high-magnitude velocity gradients.

7.2. Computational performance of the superLES library

We next evaluate the strong-scaling performance of the SR-LES and the hybrid CFD–SR-GAN coupling. On each node of the DEEP-ESB partition, we allocate eight CPUs to the CIAO solver and one GPU to the *superLES* library. The relationships between the number of allocated nodes, CPUs, GPUs, and CPU and GPU chunk sizes are tabulated in Table 6. We limit our analysis to 64 nodes based on our findings in Section 7.1 and note that performance conclusions could be limited to the particular CFD solver and SR implementation. This section employs halo regions of size $n_{halo}/n_{in} = 1/2$ and a minimum input SR size of $N_{in} = 16^3$. Strong scaling is evaluated by comparing the speed-up against the execution time on a single node.

Fig. 11 depicts the computational time of the *superLES* library's operations for the *a posteriori* LES. The *superLES* library achieves good scalability up to 16 nodes, with averaged speed-up of around 90% across all operations with increasing node count. SR-GAN inference is the primary computational cost driver, followed by the evaluation of τ_{ij}^{SR}. The CPU–GPU data transfer time is about two orders of magnitude lower. As these operations are not runtime limiting, their influence on scalability can be considered negligible. Notably, memory transfer to CPUs takes approximately twice as long as that to GPUs due to additional CPU-domain decomposition overhead. Moreover, there is a slight reduction in performance attributed to increased halo-transfer overhead with node increments between 2 and 4 nodes.

However, the performance significantly diminishes, dropping by 50% when utilizing more than 16 nodes, as shown in Fig. 11. This is caused primarily by the diminishing input SR-chunk size with increasing resources, which reduces GPU utilization. This results in a higher halo-to-SR-chunk size ratio and increases the time needed for data transfer relative to inference, making the use of GPUs less efficient. Therefore, careful balancing of CPU (CIAO solver) and GPU (*superLES* library) operations is crucial for maximizing computational efficiency (in addition to producing physically accurate predictions). The present software architecture favors hybrid CPU–GPU nodes with more CPUs and fewer GPUs per node but with relatively large memory pools. This type of node would limit domain decomposition, thus reducing the data exchange of the halo regions, while ensuring physical accuracy and optimizing GPU utilization.

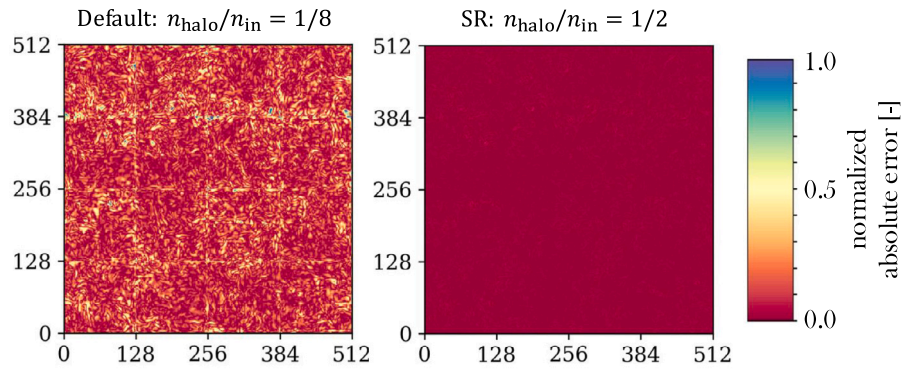

Fig. 9. Influence of halo size: CIAO solver default (left) and best balance for SR inference (right). This analysis is conducted using an input GPU chuck size $N_{in} = 16^3$.

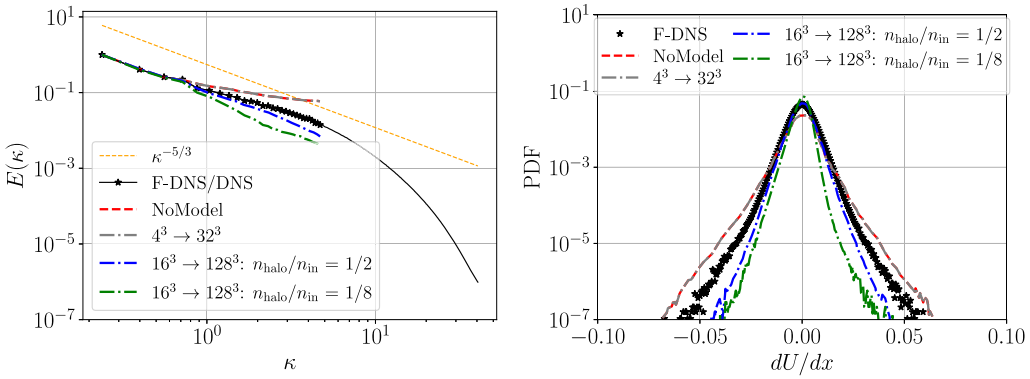

Fig. 10. In-sample *a posteriori* averaged TKE spectra (right) and PDFs of the normalized velocity gradients (left) across varying computational setups, including different GPU chunk sizes (Table 6) and halo region sizes' (Table 5). The orange line represents Kolmogorov's -5/3 power law.

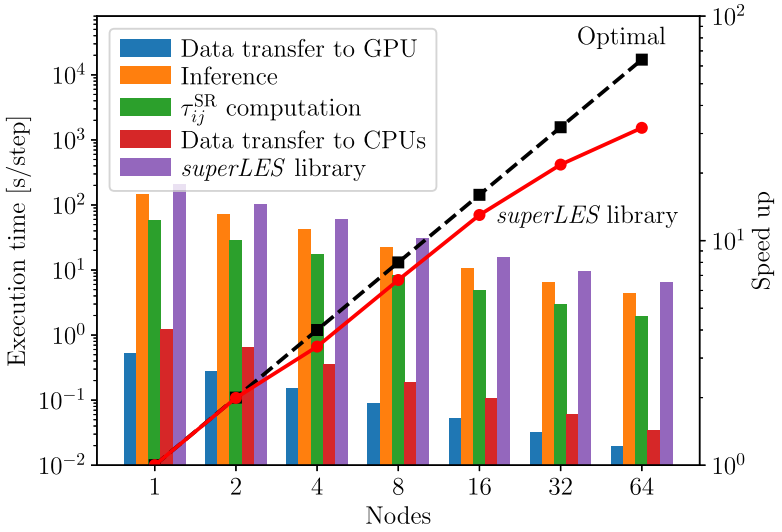

Fig. 11. Computational time of *superLES* library operations (bars) and strong scaling speed-up (lines) for increasing DEEP-ESB node counts. The execution time does not consider the CIAO solver.

8. Conclusion

We have examined the computational performance and accuracy requirements of distributed, hybrid CPU–GPU SR-GAN training and inference-coupled SR-LES. The analysis is conducted using an SR-GAN framework adapted for small-scale turbulence reconstruction and covers model training requirements, including input subbox size and mini-batch size for distributed training, and *a posteriori* testing considerations, including model-input overlap (halo) size requirements, hybrid CPU–GPU parallel scalability, and CPU/GPU load balancing. The methods introduced here are applicable to other SR-GAN frameworks employing similar training strategies.

The patch-to-patch model-training strategy extracts randomly located, nonoverlapping subboxes from a full-domain snapshot to alleviate storage requirements on limited GPU memory. The influence of training subbox size relative to η is considered, which affects both predictive accuracy and computational cost. In-sample tests reveal that training SBs need to be at least 192η in size to minimize errors in the reconstructed velocity field and hence the modeled subgrid stress. Models trained with smaller subboxes fail to accurately recover the

correct TKE, leading to significant deviations in the unclosed terms. This demonstrates that the input and output patch sizes are crucial for super-resolution predictive accuracy. Larger patches are overall advantageous, as they provide a larger receptive field and can enhance the two networks' (generator and discriminator) resilience to overfitting. However, the threshold value of $l_{\mathrm{SB}}/\eta \approx 192\eta$ may be specific to the Reynolds number, and filter size used in this study. It is therefore recommended that this value not be generalized to other conditions without additional verification across varying configurations and flow parameters.

Equally crucial is the need for DTAs to optimize GPU utilization, prioritizing computation time for local gradient estimation over the time allocated to gradient averaging and synchronizing model parameters across all GPUs. It is shown that larger minibatch sizes improve scalability, with larger proportions of runtime dedicated to gradient computations than model synchronization overhead. The synchronization time is further reduced using a GPU-based communicator (NCCL) that reduces memory copies and the associated latency.

It is likewise important to assess the influence of GPU resource allocation on training wall-time required to achieve convergence. We consider LR/HR subbox configurations that ensure physical accuracy while fixing the maximum mini-BS. Hence, the global batch size scales proportionally with the number of GPUs utilized. The wall time required for training convergence is accelerated when using up to 8 GPUs, but performance significantly deteriorates when the number of GPUs is further increased. In distributed training, the upper boundary for the LR/HR training configuration that can effectively be utilized is decreased compared to serial training. Beyond this boundary, training configurations fail to produce meaningful predictions or require increased training times. This is due to the LR/HR subbox size determining both the number of subboxes extracted and the maximum minibatch size. Beyond a certain global batch-size threshold, the model quality deteriorates, as the model's updates are too frequent relative to the number of subboxes, leading to local overfitting.

The SR-GAN model's performance is also strongly related to the discriminator's ability to distinguish real fields from generated ones. The discriminator's accuracy drops dramatically for larger subboxes. This is also caused by overfitting to the (local) training dataset due to the limited training data per GPU. The analysis underscores the significant influence of the number of GPUs employed by the DTA, impacting the stochastic optimization through variations in minibatch sizes and the number of training subboxes. Additionally, the discriminator's learning capacity plays a crucial role in indirectly affecting the generator's overall performance. It is therefore essential to consider the discriminator's behavior when implementing synchronous, distributed-data SR-GAN training for turbulence modeling.

Finally, we present inference-coupled, *a posteriori* tests of SR-GAN model accuracy, stability, and cost for realistic LES calculations. Importantly, this requires integrating the SR-GAN library with the LES solver. The *superLES* library introduces a modular coupling strategy for distributed, hybrid CPU–GPU parallelism. This allows the LES solver to exchange fields on-the-fly with the SR-GAN inference engine at runtime on heterogeneous cluster architectures. The library is developed to be easily integrated into different parallel CFD solvers, allowing for various operation modes depending on the user's hardware configuration. For the hardware we tested, the SR-chunk size varies with the allocation size. Similarly to the *a priori* investigations, the SR-GAN model's performance varies significantly with the SR-chunk size. To achieve meaningful results, accurate predictions require a certain domain size, similar to the findings of the *a priori* in-sample analysis. This presents an upper bound to the computational resources that may be employed. Additionally, domain decomposition in parallel SR-LES computations introduces boundary effects, which are particularly relevant to the convolutional operations performed by the generator. Larger SR-chunk sizes are preferable because they reduce the number of boundaries between boxes: by allocating half of the SR-chunk size to halo regions — significantly more than the default halo size in a typical CFD solver — discontinuities can be effectively prevented. Those findings are corroborated through *a posteriori* analysis in terms of averaged TKE spectra and PDF of the normalized velocity gradients.

A strong scaling study of the *superLES* library demonstrates good scalability for 16 hybrid CPU–GPU nodes. Profiling indicates that the SR-inference operation is runtime-limiting, followed by the computation of the τ_{ij}^{SR}. Scalability heavily deteriorates beyond 16 nodes as the input SR chunk size decreases with increasing number of resources. This results in a higher halo-to-SR size ratio, which increases the time required for data transfer relative to inference, thus limiting GPU utilization. Therefore, careful consideration of the hardware configuration and its implications for DTA is crucial not only for achieving accurate physical predictions but also for maximizing computational efficiency.

In conclusion, SR-GAN training can be successfully accelerated using a synchronous, distributed-data approach. However, a careful balance of training subbox sizes, global batch sizes, and discriminator feedback is essential to ensure meaningful predictions and maintain the model's learning capability. The use of modular libraries is essential to integrate with existing CFD solvers. DL modeling approaches impose additional computational requirements compared to traditional LES modeling, as physical constraints potentially impose limitations on the maximum number of computational resources that can be effectively employed.

In future work, we aim to extend these analyses to more complex configurations and different SR-GAN frameworks, higher Reynolds numbers, and larger filter sizes, and enhance the adaptability of SR-LES across diverse compute architectures. This includes optimizing for various operational modes, integrating with GPU-capable CFD solvers, and conducting a comprehensive assessment of the SR-GAN model's performance in comparison to other modeling strategies.

In order to enhance the reproducibility of this study, provide clarity on technical aspects, and facilitate faster development, access to our GIT repository will be granted upon request.

CRediT authorship contribution statement

Ludovico Nista: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Investigation, Conceptualization. **Christoph D.K. Schumann:** Writing – review & editing, Writing – original draft, Methodology, Investigation, Formal analysis. **Peicho Petkov:** Software, Methodology, Investigation, Data curation. **Valentin Pavlov:** Software, Resources. **Temistocle Grenga:** Writing – review & editing, Supervision, Methodology, Formal analysis. **Jonathan F. MacArt:** Writing – review & editing, Visualization, Supervision, Methodology, Formal analysis. **Antonio Attili:** Writing – review & editing, Supervision, Methodology, Formal analysis, Conceptualization. **Stoyan Markov:** Supervision, Resources, Project administration, Funding acquisition. **Heinz Pitsch:** Writing – review & editing, Supervision, Resources, Project administration, Funding acquisition.

Funding

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation program under the Center of Excellence in Combustion (CoEC) project, Grant Agreement No. 952181, and from the German Federal Ministry of Education and Research (BMBF) and the state of North Rhine-Westphalia for supporting this work as part of the NHR funding.

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.

Acknowledgments

The authors gratefully acknowledge the computing resources from the DEEP-EST project, which received funding from the European Union's Horizon 2020 research and innovation program under Grant Agreement No. 754304. We thank Dr. R. Sedona and Dr. G. Cavallaro for their support in the porting of the application to DEEP-EST. Part of the computations were performed with computing resources granted by RWTH Aachen University under project rwth1480.

The authors thank F. Fröde and F. Orland for their exceptional support and contributions to this research project.

Appendix. Training data requirements for SR-GAN

For successful SR-GAN training, a high-quality and well-balanced dataset is crucial. In this context, using genuine DNS fields as a reference is instrumental for accurately reconstructing small-scale features, as the discriminator must learn to distinguish these authentic HR fields from SR fields resulting from the generator. The ability of the discriminator to distinguish between these two fields is especially significant during the partially unsupervised training phase, where the discriminator's feedback guides the generator by assessing the quality of SR fields based solely on its capacity to identify genuine DNS fields which is acquired during pre-training.

Sufficiency depends on the quantity of training data is critical to enhance reconstruction accuracy and prevent training divergence. The required volume of data depends on several factors, including the model architecture, upsampling factor, and the complexity of the physics embedded within the data. Complex physical phenomena often involve intricate interactions and fine-scale structures that are challenging to capture accurately. When such complex physics is present in the training data, as is the case in turbulent flows, the model requires a large dataset to reproduce (generator) or identify (discriminator) these nuanced features. Deep architectures, with their higher number of trainable parameters, also require extensive data to prevent overfitting. Similarly, the demand for data scales with the upsampling factor, as the learning process becomes increasingly challenging the smaller the scales to be generated relative to the input field provided.

Supervised pre-training of the generator before adversarial training is beneficial, as mode collapse is prevented and a baseline level of performance established. This pre-training phase is especially important when training data is limited, as the discriminator learns to distinguish between SR and DNS more quickly than the generator is able to make adjustments, which upsets the balance of the inherent competition of the two networks in GAN training, leading to poor convergence [63]. Identifying an adequate amount of training data to initialize the generator is crucial for balanced training dynamics. Training stability and reconstruction accuracy are balanced by choosing the right amount of training data. In this work, the number of snapshots used was carefully chosen based on prior studies [26,63]. However, this specific amount may not universally apply across different data configurations, architectures, or physical complexities.

Furthermore, the availability of highly-resolved DNS data for training is constrained to relatively low Reynolds numbers due to the high computational costs associated with DNS. Without robust generalization capabilities, the SR-GAN framework remains limited to physical conditions for which DNS data exists. The SR-GAN framework's extrapolation potential at higher Reynolds numbers in both forced HIT and turbulent premixed reacting flow configurations has been demonstrated in previous works [16,24]. These a priori predictive capabilities are successful when the ratio between the filter size and the Kolmogorov length scale is consistent across training and testing conditions [16, 24]. However, further research is needed to explore the framework's extrapolation capabilities to different geometric configurations.

Data availability

Data will be made available on request.

References

[1] Durbin PA. Some recent developments in turbulence closure modeling. Annu Rev Fluid Mech 2018;50(Volume 50, 2018):77–103. http://dx.doi.org/10.1146/annurev-fluid-122316-045020.

[2] Sagaut P, Meneveau C. Large eddy simulation for incompressible flows: An introduction. In: Scientific computation, (7). Springer; 2006, http://dx.doi.org/10.1007/b137536.

[3] Smagorinsky J. General circulation experiments with the primitive equations: I. The basic experiment. Mon Weather Rev 1963;91(3):99–164. http://dx.doi.org/10.1175/1520-0493(1963)091<0099:GCEWTP>2.3.CO;2.

[4] Germano M, Piomelli U, Moin P, Cabot WH. A dynamic subgrid-scale eddy viscosity model. Phys Fluids A Fluid Dyn 1991;3(7):1760–5. http://dx.doi.org/10.1063/1.857955.

[5] Bardina J. Improved turbulence models based on large eddy simulation of homogeneous, incompressible, turbulent flows. Stanford University; 1983.

[6] Pitsch H. Large-eddy simulation of turbulent combustion. Annu Rev Fluid Mech 2006;38:453–82. http://dx.doi.org/10.1146/annurev.fluid.38.050304.092133.

[7] Kutz JN. Deep learning in fluid dynamics. J Fluid Mech 2017;814:1–4. http://dx.doi.org/10.1017/jfm.2016.803.

[8] Duraisamy K, Iaccarino G, Xiao H. Turbulence modeling in the age of data. Annu Rev Fluid Mech 2019;51:357–77. http://dx.doi.org/10.1146/annurev-fluid-010518-040547.

[9] Brunton SL, Noack BR, Koumoutsakos P. Machine learning for fluid mechanics. Annu Rev Fluid Mech 2020;52:477–508. http://dx.doi.org/10.1146/annurev-fluid-010719-060214.

[10] Ihme M, Chung WT, Mishra AA. Combustion machine learning: Principles, progress and prospects. Prog Energy Combust Sci 2022;91:101010. http://dx.doi.org/10.1016/j.pecs.2022.101010.

[11] Fukami K, Fukagata K, Taira K. Assessment of supervised machine learning methods for fluid flows. Theor Comput Fluid Dyn 2020;34(4):497–519. http://dx.doi.org/10.1007/s00162-020-00518-y.

[12] Fukami K, Fukagata K, Taira K. Super-resolution reconstruction of turbulent flows with machine learning. J Fluid Mech 2019;870:106–20. http://dx.doi.org/10.1017/jfm.2019.238.

[13] Zhou Z, Li B, Yang X, Yang Z. A robust super-resolution reconstruction model of turbulent flow data based on deep learning. Comput Fluids 2022;239:105382. http://dx.doi.org/10.1016/j.compfluid.2022.105382.

[14] Pant P, Farimani AB. Deep learning for efficient reconstruction of high-resolution turbulent DNS data. 2020, http://dx.doi.org/10.48550/arXiv.2010.11348, arXiv preprint arXiv:2010.11348.

[15] Zhao Q, Jin G, Zhou Z. Deep learning method for the super-resolution reconstruction of small-scale motions in large-eddy simulation. AIP Adv 2022;12(12):125304. http://dx.doi.org/10.1063/5.0127808.

[16] Nista L, Pitsch H, Schumann CDK, Bode M, Grenga T, MacArt JF, et al. Influence of adversarial training on super-resolution turbulence reconstruction. Phys Rev Fluids 2024;9:064601. http://dx.doi.org/10.1103/PhysRevFluids.9.064601.

[17] Kim H, Kim J, Won S, Lee C. Unsupervised deep learning for super-resolution reconstruction of turbulence. J Fluid Mech 2021;910:A29. http://dx.doi.org/10.1017/jfm.2020.1028.

[18] Fukami K, Fukagata K, Taira K. Super-resolution analysis via machine learning: a survey for fluid flows. Theor Comput Fluid Dyn 2023.

[19] Yang L, Zhang Z, Song Y, Hong S, Xu R, Zhao Y, et al. Diffusion models: A comprehensive survey of methods and applications. ACM Comput Surv 2023;56(4):1–39. http://dx.doi.org/10.1145/3626235.

[20] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative adversarial networks. Commun ACM 2020;63(11):139–44. http://dx.doi.org/10.1145/3422622.

[21] Shu D, Li Z, Barati Farimani A. A physics-informed diffusion model for high-fidelity flow field reconstruction. J Comput Phys 2023;478:111972. http://dx.doi.org/10.1016/j.jcp.2023.111972.

[22] Hassanaly M, Glaws A, Stengel K, King RN. Adversarial sampling of unknown and high-dimensional conditional distributions. J Comput Phys 2022;450:110853. http://dx.doi.org/10.1016/j.jcp.2021.110853.

[23] Bode M, Gauding M, Lian Z, Denker D, Davidovic M, Kleinheinz K, et al. Using physics-informed enhanced super-resolution generative adversarial networks for subfilter modeling in turbulent reactive flows. Proc Combust Inst 2021;38(2):2617–25. http://dx.doi.org/10.1016/j.proci.2020.06.022.

[24] Nista L, Schumann CDK, Grenga T, Attili A, Pitsch H. Investigation of the generalization capability of a generative adversarial network for large eddy simulation of turbulent premixed reacting flows. Proc Combust Inst 2023;39(4):5279–88. http://dx.doi.org/10.1016/j.proci.2022.07.244.

[25] Grenga T, Nista L, Schumann CKD, Karimi A, Scialabba G, Attili A, et al. Predictive data-driven model based on generative adversarial network for premixed turbulence-combustion regimes. Combust Sci Technol 2023;195(15):3923–46. http://dx.doi.org/10.1080/00102202.2022.2041624.
[26] Nista L, Schumann C, Grenga T, Karimi AN, Scialabba G, Bode M, et al. Turbulent mixing predictive model with physics-based generative adversarial network. In: 10th European combustion meeting. 2021, p. 460–5. http://dx.doi.org/10.18154/RWTH-2021-07028.
[27] Zhang K, Zuo W, Gu S, Zhang L. Learning deep CNN denoiser prior for image restoration. In: 2017 IEEE conference on computer vision and pattern recognition. 2017, p. 2808–17. http://dx.doi.org/10.1109/CVPR.2017.300.
[28] Lee S, You D. Data-driven prediction of unsteady flow over a circular cylinder using deep learning. J Fluid Mech 2019;879:217–54. http://dx.doi.org/10.1017/jfm.2019.700.
[29] Subramaniam A, Wong ML, Borker RD, Nimmagadda S, Lele SK. Turbulence enrichment using physics-informed generative adversarial networks. 2020, arXiv e-prints, arXiv–2003 https://arxiv.org/pdf/2003.01907.
[30] Goyal P, Dollár P, Girshick R, Noordhuis P, Wesolowski L, Kyrola A, et al. Accurate, large minibatch SGD: Training ImageNet in 1 hour. 2017, arXiv preprint arXiv:1706.02677 https://arxiv.org/pdf/1706.02677.
[31] Jia X, Song S, He W, Wang Y, Rong H, Zhou F, et al. Highly scalable deep learning training system with mixed-precision: Training ImageNet in four minutes. 2018, arXiv preprint arXiv:1807.11205 https://arxiv.org/pdf/1807.11205.
[32] Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. In: Pereira F, Burges C, Bottou L, Weinberger K, editors. Advances in neural information processing systems, vol. 25. Curran Associates, Inc.; 2012, p. 1097–105. http://dx.doi.org/10.1145/3065386.
[33] Sirignano J, MacArt JF, Freund JB. DPM: A deep learning PDE augmentation method with application to large-eddy simulation. J Comput Phys 2020;423:109811. http://dx.doi.org/10.1016/j.jcp.2020.109811.
[34] Duraisamy K. Perspectives on machine learning-augmented Reynolds-averaged and large eddy simulation models of turbulence. Phys Rev Fluids 2021;6(5):050504. http://dx.doi.org/10.1103/PhysRevFluids.6.050504.
[35] MacArt JF, Sirignano J, Freund JB. Embedded training of neural-network subgrid-scale turbulence models. Phys Rev Fluids 2021;6(5):050502. http://dx.doi.org/10.1103/PhysRevFluids.6.050502.
[36] Sirignano J, MacArt JF. Deep learning closure models for large-eddy simulation of flows around bluff bodies. J Fluid Mech 2023;966:A26. http://dx.doi.org/10.1017/jfm.2023.446.
[37] Kurz M, Offenhäuser P, Beck A. Deep reinforcement learning for turbulence modeling in large eddy simulations. Int J Heat Fluid Flow 2023;99:109094. http://dx.doi.org/10.1016/j.ijheatfluidflow.2022.109094.
[38] Fischer P, Kerkemeier S, Min M, Lan Y-H, Phillips M, Rathnayake T, et al. NekRS, a GPU-accelerated spectral element Navier–Stokes solver. Parallel Comput 2022;114:102982. http://dx.doi.org/10.1016/j.parco.2022.102982.
[39] Owen LD, Ge W, Rieth M, Arienti M, Esclapez L, Soriano BS, et al. PeleMP: The multiphysics solver for the combustion Pele adaptive mesh refinement code suite. J Fluids Eng 2024;146(4):041103. http://dx.doi.org/10.1115/1.4064494.
[40] Dupuy D, Odier N, Lapeyre C. Data-driven wall modeling for turbulent separated flows. J Comput Phys 2023;487:112173. http://dx.doi.org/10.1016/j.jcp.2023.112173.
[41] Serhani A, Xing V, Dupuy D, Lapeyre C, Staffelbach G. Graph and convolutional neural network coupling with a high-performance large-eddy simulation solver. Comput Fluids 2024;278:106306. http://dx.doi.org/10.1016/j.compfluid.2024.106306.
[42] Desjardins O, Blanquart G, Balarac G, Pitsch H. High order conservative finite difference scheme for variable density low mach number turbulent flows. J Comput Phys 2008;227(15):7125–59. http://dx.doi.org/10.1016/j.jcp.2008.03.027.
[43] Attili A, Bisetti F, Mueller ME, Pitsch H. Formation, growth, and transport of soot in a three-dimensional turbulent non-premixed jet flame. Combust Flame 2014;161(7):1849–65. http://dx.doi.org/10.1016/j.combustflame.2014.01.008.
[44] Davidovic M, Pitsch H. Scalar mass conservation in turbulent mixture fraction-based combustion models through consistent local flow parameters. Combust Flame 2024;262:113329. http://dx.doi.org/10.1016/j.combustflame.2024.113329.
[45] Tomboulides A, Lee J, Orszag S. Numerical simulation of low mach number reactive flows. J Sci Comput 1997;12:139–67. http://dx.doi.org/10.1023/A:1025669715376.
[46] Kim J, Moin P. Application of a fractional-step method to incompressible Navier-Stokes equations. J Comput Phys 1985;59(2):308–23. http://dx.doi.org/10.1016/0021-9991(85)90148-2.
[47] Palmore JA, Desjardins O. Technique for forcing high Reynolds number isotropic turbulence in physical space. Phys Rev Fluids 2018;034605. http://dx.doi.org/10.1103/PhysRevFluids.3.034605.
[48] Bailly C, Juve D. A stochastic approach to compute subsonic noise using linearized Euler's equations. In: 5th AIAA/CEAS aeroacoustics conference and exhibit. 1999, p. 1872. http://dx.doi.org/10.2514/6.1999-1872.
[49] Wang X, Yu K, Wu S, Gu J, Liu Y, Dong C, et al. ESRGAN: Enhanced super-resolution generative adversarial networks. In: Leal-Taixé L, Roth S, editors. Proceedings of the European conference on computer vision. Cham: Springer International Publishing; 2019, p. 63–79. http://dx.doi.org/10.1007/978-3-030-11021-5_5.
[50] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: 2016 IEEE conference on computer vision and pattern recognition. 2016, p. 770–8. http://dx.doi.org/10.1109/CVPR.2016.90.
[51] Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, et al. TensorFlow: Large-scale machine learning on heterogeneous systems. 2015, URL https://www.tensorflow.org/.
[52] Li K, Yang S, Dong R, Wang X, Huang J. Survey of single image super-resolution reconstruction. IET Image Process 2020;14(11):2273–90. http://dx.doi.org/10.1049/iet-ipr.2019.1438.
[53] Jolicoeur-Martineau A. The relativistic discriminator: A key element missing from standard GAN. 2018, http://dx.doi.org/10.48550/arXiv.1807.00734, arXiv preprint arXiv:1807.00734.
[54] Ben-Nun T, Hoefler T. Demystifying parallel and distributed deep learning: An in-depth concurrency analysis. ACM Comput Surv 2019;52(4):1–43. http://dx.doi.org/10.1145/3320060.
[55] Sergeev A, Del Balso M. Horovod: fast and easy distributed deep learning in TensorFlow. 2018, http://dx.doi.org/10.48550/arXiv.1802.05799, arXiv preprint arXiv:1802.05799.
[56] Smith SL, Kindermans P-J, Ying C, Le QV. Don't decay the learning rate, increase the batch size. 2017, http://dx.doi.org/10.48550/arXiv.1711.00489, arXiv preprint arXiv:1711.00489.
[57] Keskar NS, Mudigere D, Nocedal J, Smelyanskiy M, Tang PTP. On large-batch training for deep learning: Generalization gap and sharp minima. 2016, http://dx.doi.org/10.48550/arXiv.1609.04836, arXiv preprint arXiv:1609.04836.
[58] Hrycej T, Bermeitinger B, Handschuh S. Training neural networks in single vs. double precision. In: Proceedings of the 14th international joint conference on knowledge discovery, knowledge engineering and knowledge management. SciTePress, INSTICC; 2022, p. 307–14. http://dx.doi.org/10.5220/0011577900003335.
[59] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. Pytorch: An imperative style, high-performance deep learning library. In: Proceedings of the 33rd international conference on neural information processing systems, vol. 32. Red Hook, NY, USA: Curran Associates, Inc.; 2019, p. 1–12. http://dx.doi.org/10.48550/arXiv.1912.01703.
[60] Bai J, Lu F, Zhang K, et al. ONNX: Open neural network exchange. 2019, GitHub repository, GitHub, https://github.com/onnx/onnx.
[61] Cardoso R, Golubovic D, Lozada IP, Rocha R, Fernandes J, Vallecorsa S. Accelerating GAN training using highly parallel hardware on public cloud. In: EPJ web of conferences, vol. 251, EDP Sciences; 2021, p. 02073. http://dx.doi.org/10.1051/epjconf/202125102073.
[62] Hoffer E, Hubara I, Soudry D. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R, editors. Proceedings of the 31st international conference on neural information processing systems, vol. 30. Red Hook, NY, USA: Curran Associates, Inc.; 2017, p. 1729–39. http://dx.doi.org/10.48550/arXiv.1705.08741.
[63] Nista L, Schumann CKD, Scialabba G, Grenga T, Attili A, Pitsch H. The influence of adversarial training on turbulence closure modeling. In: AIAA SCITECH 2022 forum. 2022, p. 1–9. http://dx.doi.org/10.2514/6.2022-0185.