Acceleration of complex high-performance computing ensemble simulations with super-resolution-based subfilter models

Mathis Bode[a,*], Jens Henrik Göbbert[a]

[a] *Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH, 52425 Jülich, Germany*

Abstract

Direct numerical simulation (DNS) of fluid flow problems has been one of the most important applications of high-performance computing (HPC) in the last decades. For example, turbulent flows require the simultaneous resolution of multiple spatial and temporal scales as all scales are coupled, resulting in very large simulations with enormous degrees of freedom. Another example is reactive flows, which typically result in a large system of coupled differential equations and multiple transport equations that must be solved simultaneously. In addition, many flows exhibit chaotic behavior, meaning that only statistical ensembles of results can be compared, further increasing the computational time. In this work, a combined HPC/deep learning (DL) workflow is presented that drastically reduces the overall computational time required while still providing acceptable accuracy.

Traditionally, all the simulations required to compute ensemble statistics are performed using expensive DNS. The idea behind the combined HPC/DL workflow is to reduce the number of expensive DNSs by developing a DL-assisted large-eddy simulation (LES) approach that uses a sophisticated DL network, called PIESRGAN, as a subfilter model for all unclosed terms and is accurate enough to substitute DNSs. The remaining DNSs are thus used in two ways: first, as data contributing to the ensemble statistics, and second, as data used to train the DL network. It was found that in many cases two remaining DNSs are sufficient for training the LES approach. The cost of the DL-supported LES is usually more than one order of magnitude cheaper than the DNS, which drastically speeds up the workflow, even considering the overhead for training the DL network.

Keywords: Generative Adversarial Network, High-Performance Computing, Machine Learning, Super-Resolution, Reactive Flows

1. Introduction

Solving the Navier-Stokes or derived equations for fluid flows, possibly coupled with multi-physics phenomena, is one of the most established high performance computing (HPC) applications. However, even with today's fastest supercomputers, some problems would be prohibitively expensive to solve. Typical engineering flow applications have in common that a simulation must properly discretize the large-scale flow behavior, such as that enforced by geometry features. This is particularly challenging for turbulent flows, which present another challenge: All scales must be simultaneously fully resolved down to the smallest scale of fluid motion to accurately predict its behavior because all length scales are coupled. The more turbulent a flow is, as measured by the Reynolds number, the ratio of inertial to viscous forces, the more the large and small scales are separated [1]. Consequently, direct numerical simulations (DNS) that aim to resolve all scales of turbulent flows can become very expensive - especially since many common engineering problems are highly turbulent. Therefore, many flow simulations require modeling of smaller scales to become computationally affordable, such as in large-eddy simulation (LES) [1, 2]. This can be done by decoupling the flow scales through a filter operation. Then, only the larger scales are solved, and the effect of the smaller scales below the filter on the larger scales is modeled. In LES, these so-called subfilter models, such as the Smagorinsky model [3], are therefore crucial.

A particularly challenging fluid flow problem is turbulent combustion. The chemistry that must be accurately predicted adds another layer of complexity. This is often done by solving a set of coupled partial differential equations (PDEs) for individual species along with the flow equations. This makes the simulations much more expensive than pure turbulent cases, and the resulting data are very large, since the data of all scalar fields must be stored. Furthermore, in order to sufficiently discretize e. g. a reaction zone [2], the smallest chemically relevant scales can be even smaller than the flow scales and thus require an even finer mesh.All in all, simulating turbulent flows with combustion quickly becomes prohibitively expensive and is often done on the largest supercomputers available. For this reason, they are an excellent target case for the present work.

This paper presents the application of recently developed AI super-resolution-based subfilter models to LES[4, 5, 6, 7, 8, 9, 10, 11, 12], focusing on their use to accelerate large-scale supercomputing workflows. Much attention has been given in the computer science community to AI super-resolution or single image super-resolution (SISR) problems. In such SISR problems, machine learning (ML) or deep learning (DL) techniques

*Corresponding author
Email address: m.bode@fz-juelich.de (Mathis Bode)

are used to add information to images in order to increase image resolution (i. e., to super-resolve the image). Typically, a network is trained on a large number of images to extract and learn features that are successively added to the super-resolved target image based on local information. In this way, they outperform classical techniques such as bicubic interpolation. To directly learn the end-to-end mapping between low- and high-resolution images, Dong et al. [13] introduced a super-resolution convolutional neural network (SRCNN), a deep convolutional neural network (CNN). Their approach has been continuously improved [14, 15, 16, 17, 18, 19, 20, 21] to correct several shortcomings, such as oversmoothed results, and to improve prediction accuracy, e. g., by introducing the concept of perceptual loss to better predict high-frequency details. Ledig et al. [22] suggested the use of generative adversarial networks (GANs) [23] instead of CNNs, which was further updated to the enhanced super-resolution GAN (ESRGAN) [24].

The ambition to add information based on the local state makes AI super-resolution a promising tool for any kind of underresolved simulation and experimental data, such as from satellites for weather prediction or many HPC problems ranging from climate research [25] to cosmology. [26]. However, this work focuses on the application of AI super-resolution to LES subfilter modeling to efficiently advance flow simulations in time, focusing on the case of turbulent premixed flame kernels. Turbulent premixed flame kernels have already been studied with super-resolution LES by Bode et al. [7]. Their work is substantially extended by the analysis and discussion in this paper. This includes a novel discussion of flame morphology, a first evaluation of the resulting scale-by-scale budget, and a quantification of the closure error with super-resolved LES data. Furthermore, a detailed discussion of training and evaluation costs is provided, emphasizing the speed-up potential of super-resolution LES. Finally, super-resolution LES results are used to quantify statistical fluctuations for an application case.

This paper is organized as follows: The next section explains the subfilter modeling approach, including the architecture, algorithm, and implementation details. Subsequently, additional methods are described. Then, the physical results of the flame kernel case are presented, the prediction quality of the model is highlighted, and computational aspects are discussed. The paper finishes with conclusions.

2. Super-Resolution Subfilter Modeling

This section summarizes the essential aspects of the so-called physics-informed enhanced super-resolution GAN (PIESRGAN), which will be applied as a super-resolution subfilter approach in the following. First, a model equation is introduced to explain the necessity of subfilter models in the LES context. Then, the architecture of the network and the closure algorithm are described. Finally, training and HPC implementation aspects are discussed.

2.1. Model Problem

A model transport equation for a scalar ϕ is

$$\partial_t (\rho\phi) + \nabla \cdot (\rho\phi\mathbf{u}) = \nabla \cdot (\rho D \nabla (\phi)) + \dot{\omega}, \tag{1}$$

where ∂_t is the time derivative, ∇ is the del operator corresponding to the spatial coordinates, ρ is the density, $\mathbf{u}$ is the velocity vector, D is the molecular diffusivity, and $\dot{\omega}$ is the source term. The direct solution of Eq. (1) together with other equations, such as those for density, momentum, and energy, would be called DNS if the simulation sufficiently resolves all relevant scales. A filter kernel $G(\mathbf{r})$ is used to decouple the large and small scales in the LES. This operation can be defined as

$$\overline{\{\cdot\}}(\mathbf{x}) = \iiint G(\mathbf{r})\{\cdot\}(\mathbf{x}-\mathbf{r})\,\mathrm{d}\mathbf{r}, \tag{2}$$

where an overbar denotes filtered quantities. Examples for G are the symmetric Gaussian filter kernel given by

$$G(\kappa) = \exp\left(-\frac{\kappa^2\Delta^2}{24}\right), \tag{3}$$

where κ is the magnitude of the wavenumber vector $\boldsymbol{\kappa}$, and wavenumber cutoff filter kernels.

It is convenient to work with Favre-filtering, denoted by a tilde and defined as $\overline{\rho}\widetilde{\{\cdot\}} = \overline{\rho\{\cdot\}}$., for compressible and variable density flows. A quantity can then be split into a filtered part and the subfilter contribution as $\{\cdot\} = \widetilde{\{\cdot\}} + \{\cdot\}''$ with $\widetilde{\{\cdot\}''} = 0$. Applied to Eq. (1) and assuming constant diffusivity and Reynolds operators, the Favre-filtered equation becomes

$$\partial_t\left(\overline{\rho}\widetilde{\phi}\right) + \nabla\cdot\left(\overline{\rho}\widetilde{\phi}\widetilde{\mathbf{u}}\right) = \nabla^2\left(\overline{D\rho\phi}\right) + \overline{\dot{\omega}} - \nabla\cdot\left(\overline{\rho}\widetilde{\phi''\mathbf{u}''}\right), \tag{4}$$

which is very similar to the unfiltered equation, Eq. (1), but has an additional term, which is shown here as the last term. All but this last term contain only filtered quantities, which can be fully resolved because the very small, possibly unresolvable scales are removed by the filter. However, the additional term in Eq. (4) contains subfilter contributions that are unknown in the LES, so this term is not closed and must be modeled.

2.2. Architecture

PIESRGAN is based on the idea of GANs, which are characterized by a generator network and a discriminator network. GAN models are generative models that aim at estimating the unknown probability density of observed data without an explicit data likelihood function, i. e. with unsupervised learning. The generator network is used for modeling and creates new modeled data. The discriminator tries to distinguish between generator-generated and real data and provides feedback to the generator network. Thus, as the learning process progresses, the generator gets better at creating data that is as close to real data as possible, and the discriminator learns to better identify fake data. This process can be thought of as two players playing a minimax zero-sum game to estimate the probability distribution of the unknown data.

The network architecture and training process is sketched in Fig. 1. Fully resolved 3-dimensional (3-D) data ("H") is filtered to obtain filtered data ("F"). The filtered data is used as input to the generator to produce the reconstructed data ("R"). The accuracy of the reconstructed data is evaluated using the fully resolved data. The discriminator attempts to discriminate between reconstructed and fully resolved data. The accuracy of the reconstruction is measured by the loss function which is

$$\mathcal{L} = \beta_1 L_{\text{adversarial}} + \beta_2 L_{\text{pixel}} + \beta_3 L_{\text{gradient}} + \beta_4 L_{\text{physics}}, \quad (5)$$

where $\beta_1, \beta_2, \beta_3$, and β_4 are coefficients that weight the different contributions of the loss terms, with $\sum_i \beta_i = 1$. In Eq. (5), the adversarial loss is the relativistic adversarial loss of the discriminator/generator, which measures both how well the generator is able to produce accurate reconstructed data relative to the fully resolved data, and how well the discriminator is able to detect false data. The pixel loss and the gradient loss are defined using the mean-squared error (MSE) of the feature itself and the gradient of the feature, respectively. The physics loss enforces physically motivated conditions such as conservation of mass, species, and elements, depending on the underlying physics of the problem. It reads

$$L_{\text{physics}} = \beta_{41} L_{\text{mass}} + \beta_{42} L_{\text{species}} + \beta_{43} L_{\text{elements}}, \quad (6)$$

where β_{41}, β_{42}, and β_{43} are coefficients that weight the various physical contributions to the loss term, with $\sum_i \beta_{4i} = 1$. The physically motivated loss term is very important for the application of PIESRGAN to flow problems. If the conservation laws are not very well fulfilled, the simulations tend to explode rapidly, which is an important difference to super-resolution in the context of images. Errors that may be acceptable there can easily become too large for use as a subfilter model [5].

The 3-D CNN layers (Conv3D) [27] are the backbone of the generator. They use a kernel size of 3 and stride 1 combined with Leaky Rectified Linear Unit (LeakyReLU) layers for activation [28]. The residual in residual dense block (RRDB) introduced for ESRGAN is essential for the performance of state-of-the-art super-resolution. It replaces the residual block (RB) used in previous architectures and includes fundamental architectural elements such as residual dense blocks (RDBs) with skip connections. A residual scaling factor β_{RSF} helps to avoid instabilities in forward and backward propagation. RDBs use internal dense connections. The output of each layer within the dense block (DB) is sent to all subsequent layers. The discriminator network is simpler. It inherits basic CNN layers (Conv3D) combined with LeakyReLU layers for activation with and without batch normalization (BN). The final layers include a fully connected layer with LeakyReLU and dropout with dropout factor β_{dropout}. For all cases in this work, 80 layers were used for the generator network and 28 layers were used for the discriminator network. A summary of all hyperparameters is given in Tab. 1. In general, the network hyperparameters were found to be robust with respect to the different test cases, i. e., a working combination for one case usually results in sufficiently accurate results for other cases. However, the complex combustion case presented in this work requires more physically motivated terms as part of the loss function than the cases discussed by Bode et al. [5]. Since the additional terms for L_{physics} implicitly reduce the weight for the original mass conversation enforcement term, the range for β_4 has been increased. Subsequently, the lower bound for β_2 was reduced to satisfy the summation condition. In general, the β coefficients cannot be estimated theoretically and are the result of extensive testing. This can be done either manually or using AutoML frameworks.

Table 1: Overview of the PIESRGAN hyperparameters. The given ranges represent the sensitivity intervals with acceptable network results. The central values were used for the case in this work.

β_1	$[0.2 \times 10^{-5}, 0.6 \times 10^{-4}, 0.8 \times 10^{-4}]$
β_2	$[0.721, 0.890, 0.918]$
β_3	$[0.04, 0.06, 0.15]$
β_4	$[0.01, 0.05, 0.18]$
β_{RSF}	$[0.1, 0.2, 0.3]$
β_{dropout}	$[0.2, 0.4, 0.5]$
$l_{\text{generator}}$	$[1.2 \times 10^{-6}, 4.5 \times 10^{-6}, 5.0 \times 10^{-6}]$
$l_{\text{discriminator}}$	$[4.4 \times 10^{-6}, 4.5 \times 10^{-6}, 8.5 \times 10^{-6}]$

2.3. Algorithm

PIESRGAN-LES is advanced in time using the established LES equations. As a consequence of the filter operation on the equations, unclosed terms appear that require information below the filter width to be evaluated. The LES subfilter algorithm aims to reconstruct this necessary information to close the LES equations. This is done at each time step. Chemistry can be included in the PIESRGAN during the training process [6].

The species splitting introduced by Bode [6, 8] was used for the production runs in this work. Consequently, the set of species was split into primary and secondary species. While the secondary species were updated at the same time as the velocities, the update of the primary species is shifted by half a time step. Furthermore, the prediction accuracy for the primary species is improved by solving additional transport equations on the reconstructed mesh. Solving these additional transport equations at high resolution also helps to deal with small reconstruction errors and dissipates small Gibbs-type errors.

Since chemistry is often only locally active, this can also be used to save computational time by adaptively solving only in relevant regions. The algorithm starts with the LES solution Φ_{F}^{n} at time step n, which includes the entirety of all fields in the simulation except the primary species, and the LES solution of the primary species $\Xi_{\text{F}}^{n+1/2}$ at time step $n + 1/2$. It consists of repeating the following steps:

1. Use the PIESRGAN to reconstruct Φ_{R}^{n} from Φ_{LES}^{n} with $\Xi_{\text{F}}^{n+1/2}$ as additional information.
2. Use Φ_{R}^{n} to estimate the unclosed terms Ψ_{LES}^{n} in the LES equations for all Φ fields by applying a filter operator.
3. Use Ψ_{LES}^{n} and Φ_{LES}^{n} to advance the LES equations of Φ to Φ_{LES}^{n+1}.
4. Use the PIESRGAN to reconstruct $\Xi_{\text{R}}^{n+1/2}$ from $\Xi_{\text{LES}}^{n+1/2}$.
5. Use $\Xi_{\text{R}}^{n+1/2}$ to update the fields of Ξ to $\Xi_{\text{R}}^{n+1/2;\text{update}}$ by solving the unfiltered scalar equations on the mesh of $\Xi_{\text{R}}^{n+1/2}$ with Φ_{R}^{n} as additional information.
6. Use $\Xi_{\text{R}}^{n+1/2;\text{update}}$ to estimate the unclosed terms $\Gamma_{\text{LES}}^{n+1/2}$ in the LES equations of Ξ for all fields by evaluating the local terms with $\Xi_{\text{R}}^{n+1/2;\text{update}}$, Φ_{F}^{n}, Φ_{F}^{n+1}, and Φ_{R}^{n} as well as applying a filter operator.

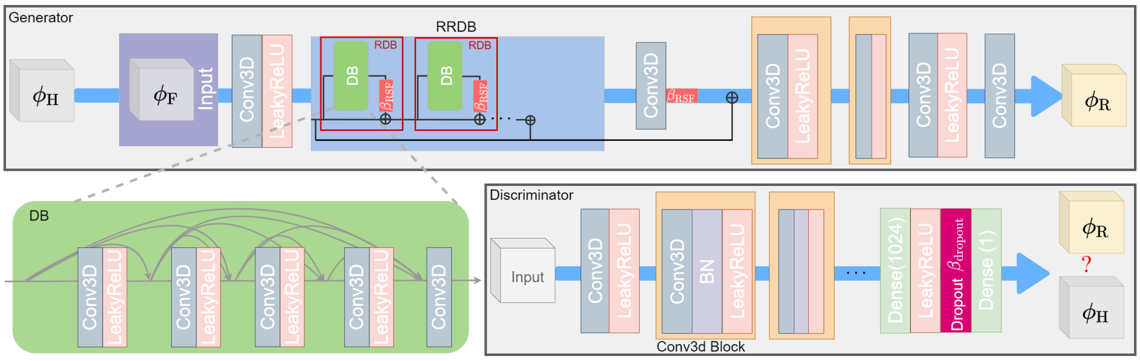

Figure 1: Sketch of PIESRGAN. "H" denotes high-fidelity data, such as DNS data, "F" are corresponding filtered data, and "R" are the reconstructed data. The components are: Conv3D - 3D Convolutional Layer, LeakyReLU - Activation Function, DB - Dense Block, RDB - Residual Dense Block, RRDB - Residual in Residual Dense Block, β_{RSF} - Residual Scaling Factor, BN - Batch Normalization, Dense - Fully Connected Layer, Dropout - Regularization Component, $\beta_{dropout}$ - Dropout Factor. Image from [5] which is under CC BY 4.0 license.

7. Use $\Gamma_{LES}^{n+1/2}$ and $\Xi_{LES}^{n+1/2}$ to advance the LES equations of Ξ to $\Xi_{LES}^{n+3/2}$.

2.4. *Training Details*

The data was split into training and test data to avoid reproduction of fully seen data. During the training and querying process, it was found that consistent normalization of quantities is very important for highly accurate results [5]. For example, in turbulence and combustion, some quantities have a logarithmic behavior. A normalization and transformation to a non-logarithmic scale supports the learning ability of the neural network. Furthermore, the training and reconstruction are done on the basis of subboxes, since the reconstruction of too large boxes at once can become very memory intensive. Typically, each subbox is chosen large enough to cover the relevant physical scales [5], and sizes from 16^3 to 32^3 gave good results for the test cases presented in this work. The filter width can become problematic when non-uniform meshes are used. In these cases, training with multiple filter widths is suggested to achieve good accuracy over the entire domain [8].

A common challenge during the training process is the initialization of the network weights. To simplify the training process, the trained weights of the turbulence-only case were used to initialize the weights of the networks for the combustion cases. The generator and discriminator can then be further updated with the case-specific data. However, it was found that further updating of the discriminator weights is often not desirable, and that generator-only updates lead to better overall prediction results.

The extrapolation capability of data-driven methods is always an issue. Many trained networks only work well in regions that were accessible during the training process. This can be very problematic in flow applications, where low Reynolds number data is often abundant, while high Reynolds number data is not computable at all, making transfer learning difficult. To deal with this problem, concepts such as two-step training approaches [5] can be used, relying on the improved predictive capabilities of GANs compared to single networks [6]. This open question is beyond the scope of the present work.

2.5. *High-Performance Computing Implementation Details*

The latest supercomputers derive most of their power from GPUs. For example, each JURECA-DC GPU cluster node at the Jülich Supercomputing Center, Forschungszentrum Jülich, has two AMD EPYC 7742 CPUs with a total of 128 cores and four Nvidia A100 GPUs, resulting in 2322 EFLOP. 94.4 % of this performance comes from the GPUs, while all CPU cores account for only 5.6 %. Therefore, any efficient HPC implementation must make maximum use of the GPUs, and in order to use the GPUs efficiently, PIESRGAN was implemented using a TensorFlor/Keras framework with OpenMP, MPI, and CUDA. The implementation details for training and simulation with PIESRGAN are described in the following two sections.

The implementation could be simplified if the computing power of the CPUs were neglected and everything was computed on the GPUs. Considering the FLOP performance, this would lead to only small losses on modern computing nodes, as discussed for the JURECA-DC GPU cluster nodes. However, there are two practical reasons against this: First, on many cluster nodes, the ratio of CPU cores to the number of GPUs is larger, shifting the available computing power toward the CPUs. Second, and more importantly, many complex simulation codes have not yet been ported to GPUs and may never be. Therefore, they would not be able to adopt a GPU-only approach, making a high degree of portability inevitable given the software frameworks currently in use. As a consequence, the approach taken in this work relies on a well-defined API. However, the use of CPUs and GPUs also means that an imbalance between CPU and GPU workloads can thwart the overall simulation performance, since, e. g., the GPUs often have to wait until the CPUs have completed their equations. As mentioned before, the size of the reconstructed subboxes can be used to balance this load. The larger the subboxes, the more expensive it is for the GPUs, but the smaller the number of cells on the LES mesh that the CPUs need to process. Note that the API can also be used with CPUs on both sides, allowing PIESRGAN-LES to run on CPU-only clusters.

To facilitate the reproducibility of this work and clarify more technical details, a basic version of PIESRGAN is available on GitLab (https://git.rwth-aachen.de/Mathis.Bode/PIESRGAN.git).

3. Further Methods

This section presents further methods used in the discussion of the results, such as the numerical methods of the simulation code framework and the tools used to analyze the physics of the premixed flame kernel case.

3.1. Simulation Code Framework

The cases were computed with the low-Mach solver of the CIAO code on a staggered mesh [29]. CIAO is an arbitrary order finite difference code that solves the Navier-Stokes equations together with multi-physics effects [30]. It is optimized to run efficiently on CPU-intensive supercomputers [31, 32]. The Poisson equation was solved using the HYPRE-AMG multigrid solver [33, 34], and the species and temperature equations were discretized using a fifth-order weighted essentially non-oscillatory (WENO) scheme [35]. Furthermore, the symmetric operator split of Strang [36] was applied to these equations. The resulting system of ordinary differential equations for the zero-dimensional homogeneous reactor in each grid cell was solved with a fully time-implicit backward difference method [37, 38]. The Hirschfelder and Curtiss approximation [39] was used with a velocity correction to ensure species mass conservation to simplify multi-species diffusion.

3.2. Flame Geometry

The geometric properties of a flame kernel are sufficient to accurately predict its macroscopic properties such as flame speed and heat release, assuming the laminar flamelet concept and unity Lewis numbers. A key is the flame surface density Σ, which determines the reaction rate [40]. Due to the interaction with the turbulent flow, the flame surface exhibits a wide range of different scales. These range from integral scales, which contain information about the initial and boundary conditions, to the Kolmogorov length scale, which is the dissipative limit scale for which a quasi-universal statistical theory exists [41].

Given a progress variable ζ that varies in space, a threshold ζ_0 can be used to define an interface that separates the regions where $\zeta(\boldsymbol{x}) > \zeta_0$ from the regions where $\zeta(\boldsymbol{x}) < \zeta_0$. Consequently, for a given threshold, a phase indicator function $\Gamma(\boldsymbol{x}, t)$ can be defined as $\Gamma(\boldsymbol{x}, t) = \mathcal{H}(\zeta(\boldsymbol{x}, t) - \zeta_0)$, where $\mathcal{H}$ is the Heaviside step function. To proceed, the structure function of the phase indicator function, i.e,

$$\langle(\delta\Gamma)^2\rangle(\boldsymbol{r}; t) = \langle(\Gamma(\boldsymbol{x} + \boldsymbol{r}, t) - \Gamma(\boldsymbol{x}, t))^2\rangle, \tag{7}$$

is introduced and finally computed by taking an angular average to allow a scale sensitive analysis of the geometry of the surface of the flame kernel.

The relationship between the indicator structure function and the morphology of the flame kernel surface can be demonstrated by taking the small and large scale boundaries. Kirste and Parod [42] showed that for class C^2, the small scale limit of Eq. (7) is given by

$$\lim_{r\to 0}\langle(\delta\Gamma)^2\rangle = \frac{\Sigma r}{2} + O(r^2), \tag{8}$$

where the surface density is given by

$$\Sigma = \langle|\nabla\Gamma|\rangle \tag{9}$$

and quantifies the area of the iso-scalar surface of the flame divided by the volume V. Note that the calculation of the surface density by Eq. (7) is computationally very efficient and does not require the tessellation of the iso-surface or the definition of a level set function. In the large-scale limit, $\langle(\delta\Gamma)^2\rangle$ tends to

$$\lim_{r\to\infty}\langle(\delta\Gamma)^2\rangle = 2\langle\Gamma\rangle(1 - \langle\Gamma\rangle) \tag{10}$$

and is therefore related to the volume $\langle\Gamma\rangle$ enclosed by the iso-surface. If there is sufficient scale separation, Eq. (7) also gives information about the fractal dimension of the flame kernel surface. Furthermore, it is common to express the surface density by a characteristic length scale

$$L_\Sigma = \frac{4\langle\Gamma\rangle(1 - \langle\Gamma\rangle)}{\Sigma}, \tag{11}$$

which is related to the wrinkle scale $L_\Sigma^* = (4\Sigma_{\max})^{-1}$ of premixed flames, which lies between the Taylor microscale and the Kolmogorov scale, as recently shown by Kulkarni et al. [43].

3.3. Flame Morphology

The morphology of the flame kernel surface plays an important role in the evolution of the flame. The flame speed depends on geometrical features, such as the local curvature, and the interaction of the flame kernel with the surrounding flow field, represented by the stretch rate. The modeling of flame speed and heat release rate requires an accurate prediction of the characteristics of the flame surface.

The modeling of turbulent premixed flames is often based on the so-called level-set approach, where the propagation of the flame surface is modeled by prescribing a burning velocity. This burning velocity depends on the local morphology of the flame surface and its interaction with the turbulence [44]. A precise knowledge of the morphology of turbulent flames is therefore essential to accurately predict flame propagation. In the next section, a scale sensitive approach [45] is used to test the ability of the PIERSGAN to provide this information.

A kinematic scale-sensitive framework has been developed to analyze the evolution of reacting turbulent flame fronts by Gauding et al. [45]. This framework is derived from first principles and is based on a two-point statistical equation for iso-scalar sets representing the flame front. It provides detailed information on turbulent strain and heat release on the flame structure and flame propagation at different scales and is therefore ideally suited to validate the PIERSGAN.

The statistical scale-sensitive budget for the indicator structure function in scale space $\boldsymbol{r}$ is given by

$$\begin{aligned}\underbrace{\partial_t\langle(\delta\Gamma)^2\rangle}_{\text{Unsteady}} = &\underbrace{-\langle\boldsymbol{\nabla}_{\boldsymbol{r}} \cdot \langle(\delta\boldsymbol{u})(\delta\Gamma)^2\rangle\rangle}_{\text{Transfer}-r}\\ &\underbrace{+2\langle S_d\rangle_s\Sigma - 4\langle\Gamma^{\pm}S_d^{\mp}|\boldsymbol{\nabla}_{\boldsymbol{x}}\Gamma|^{\mp}|\rangle}_{S_d-\text{Term}} \underbrace{+2\langle(\delta\Gamma)^2(\boldsymbol{\nabla}_{\boldsymbol{x}} \cdot \boldsymbol{u})^{\oplus}\rangle}_{\text{Dilatation}},\end{aligned} \tag{12}$$

where the superscript $\bullet^{\oplus} = (\bullet^{+} + \bullet^{-})/2$ is the arithmetic mean of the quantity $\bullet$ between the two points $\boldsymbol{x}^{+}$ and $\boldsymbol{x}^{-}$. Equation (12) describes the relevant physical effects acting on the flame surface defined by the indicator function $\Gamma(\boldsymbol{x}, t)$ and, in addition to the unsteady term, includes an inter-scale transfer term, a diffusion term resulting from the displacement speed S_d, and a dilatation term that accounts for the density change over the flame surface due to heat release. Knowledge of the displacement speed is essential for modeling turbulent premixed flames. Equation (12) is derived from first principles and testing whether the equation is satisfied over the full range of scales r is an extensive validation of the PIERSGAN approach.

4. Results and Discussion

This section presents modeling results for a premixed flame kernel with uniform Lewis numbers. The accuracy and computational cost are discussed.

4.1. Premixed Flame Kernel

A database of different iso-octane/air flame kernels was used as an application in this work. The cases were computed by Falkenstein et al. [46, 47, 48] and include fully turbulent flame kernels and planar cases, as well as cases with unity Lewis numbers and constant Lewis numbers. The turbulent flame kernels used a uniform mesh of 960 grid points per direction to discretize a periodic box resulting in five cells per Kolmogorov length at the start of the simulation and a reaction mechanism with 26 species. The box was always initialized with homogeneous isotropic turbulence (HIT), which decays over time and mimics conditions in SI engines. The unburnt temperature is $T_u = 600\,\mathrm{K}$, the initial pressure is $p^0 = 6\,\mathrm{bar}$, and the air-fuel equivalence ratio is $\phi = 1.0$ for all cases and the cases are in the thin reaction zone combustion regime. This work focuses on the unity Lewis number case. Note that five cells per Kolmogorov length resolves the turbulence very well. However, the flame thickness is significantly smaller, making this high resolution necessary [46].

Falkenstein et al. [46] introduced a simplified reaction progress variable ζ with the thermal diffusion coefficient D_{th} as the diffusion coefficient to simplify their analysis. It reads

$$\partial_t (\rho\zeta) + \nabla \cdot (\rho\mathbf{u}\zeta) = \nabla \cdot (\rho D_{\mathrm{th}} \nabla (\zeta)) + \dot{\omega}_{\zeta}, \qquad (13)$$

where $\dot{\omega}_{\zeta}$ is the chemical source term of the simplified reaction progress variable defined as

$$\dot{\omega}_{\zeta} = \dot{\omega}_{\mathrm{H_2}} + \dot{\omega}_{\mathrm{H_2O}} + \dot{\omega}_{\mathrm{CO}} + \dot{\omega}_{\mathrm{CO_2}}. \qquad (14)$$

The evolution of a flame kernel realization is visualized in Fig. 2, and the 3-D structure is shown in Fig. 3.

Quantitatively, the evolution of the flame kernel with time is shown in Figs. 4-6 for the turbulent kinetic energy in the unburnt mixture, the surface density Σ, and the characteristic length scale of the surface L_{Σ}. Although the HIT decays, the decay is very slow due to the short time span considered. The surface density and the surface density length scale are more

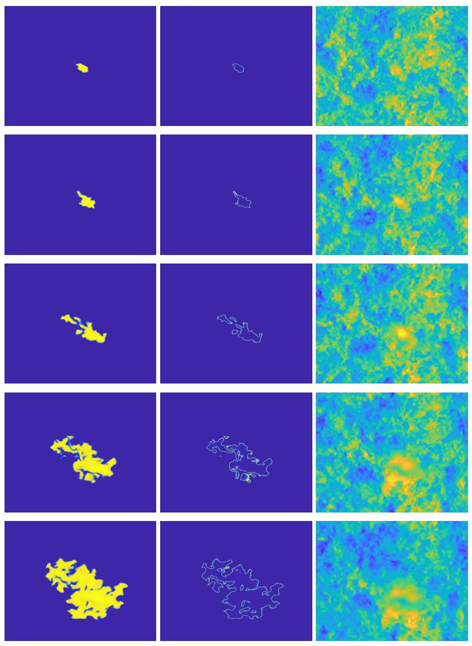
Figure 2: Visualization of 2-D slices of ζ, $\dot{\omega}_{\zeta}$, and U (left to right) for five different increasing time steps (top to bottom) for the fully turbulent flame kernel with unity Lewis number. The first time shows 6.0×10^{-5} s, and the time increment is 7.5×10^{-5} s. The final time is 3.6×10^{-4} s, which is also used for the a priori analysis. Colormaps span from blue (minimum) over green to yellow (maximum). Note that the flame kernel does not break into parts at the latest time shown. A coherent flame kernel topology was maintained at all times.

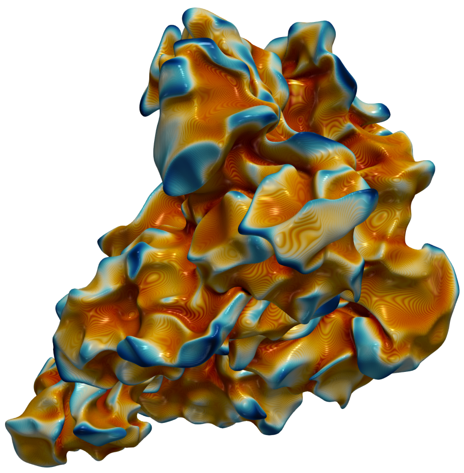
Figure 3: Flame surface of one flame kernel realization colored with the source term of a progress variable.

interesting. The surface density is an important quantity for the global heat release [46, 47, 48]. It increases with time as the wrinkling increases. The characteristic length scale grows slightly with time.

4.2. Flame Morphology

The budget given by Eq. (12) is shown in Fig. 7 for the DNS data. The budget reflects the notion that the sum of the dilation and transfer terms is positive because turbulent motion increases the morphological content of the flame surface. On

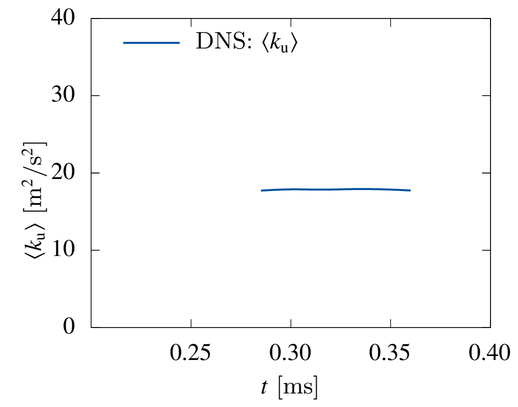

Figure 4: Temporal evolution of the averaged turbulent kinetic energy in the unburnt mixture $\langle k_u \rangle$.

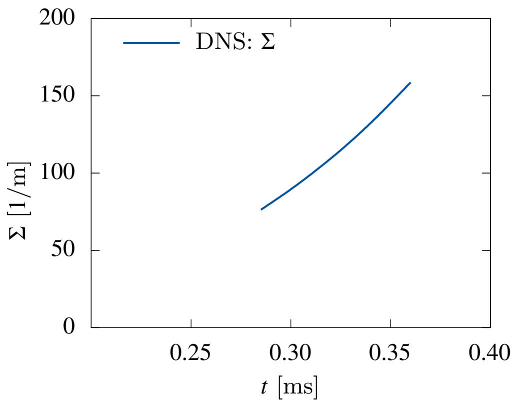

Figure 5: Temporal evolution of the surface density Σ.

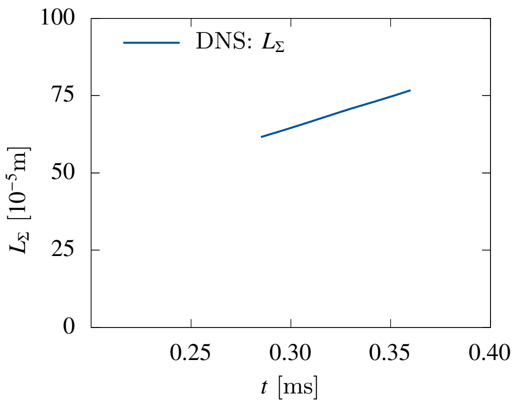

Figure 6: Temporal evolution of the characteristic length scale L_Σ.

the other hand, the diffusion term is negative, reflecting the smoothing effect of molecular diffusion on the flame surface. The unsteady term is strictly positive, indicating the growth of the flame core volume by chemical reaction. The budget is well closed on all scales.

4.3. Modeling Quality

Figure 8 presents reconstruction results for the simplified reaction progress variable, two species mass fractions, and a velocity component. The agreement between the fully resolved and reconstructed data is good. The filtered data are less sharp as they have less small-scale structure due to the 15 cells filtering with an explicit Gaussian filter. Note that due to the periodic boundary conditions, the same filter can be applied everywhere.

A first quantitative but still a priori test can be done using the scale-by-scale budget introduced in Sec. 4.2 and shown in Fig. 7. Figure 9 shows the difference of the budget between the DNS data and the reconstructed data obtained by PIERSGAN. This difference is close to zero on all scales except the very largest scales beyond the radius of the flame kernel. This proves the validity of the accelerated HPC/DL-workflow over all scales. Since the largest scales do not contribute significantly to the calculation of the displacement speed S_d, the PIERSGAN is well suited for modeling turbulent flames.

Falkenstein et al. [47, 48] indicate that accurate prediction of surface growth is a key to understanding cycle-to-cycle variations (CCVs) in terms of global heat release in such flame kernel setups. The surface growth is a highly coupled quantity and very difficult to predict with reduced order models. The a posteriori prediction of the surface density is therefore a perfect metric to evaluate the prediction quality of PIESRGAN-LES. The evolution of the flame surface density Σ is shown in Fig. 10,as well as the ensemble averaged turbulent kinetic energy in the unburnt mixture in Fig. 11. To further demonstrate the training and execution processes of PIESRGAN, the flame surface densities of three different flame kernels are shown, two flame kernels used for training and one flame kernel used for testing. Variations between the different runs can be seen, making the accurate prediction of the target flame kernel remarkable. Overall, the agreement between DNS data and PIESRGAN-LES data is good, highlighting the potential of the introduced PIESRGAN-subfilter modeling approach.

4.4. Speedup

Two important factors must be taken into account when evaluating the simple speedup of PIESRGAN-LES over DNS without training costs: First, PIESRGAN shifts and often reduces the number of operations from advancing quantities on a fine mesh with resulting smaller time step to advancing quantities on a coarser mesh with resulting larger time step plus doing the reconstruction. Depending on the case, this can result in a speedup for PIESRGAN-LES even on a CPU-only cluster that can be used for both PIESRGAN-LES and DNS. Second, the ability of PIESRGAN-LES to efficiently use HPC clusters with GPUs must be considered. However, a fair consideration is difficult if DNS is not able to use GPUs at all. For example, the supercomputing environment at the Jülich Supercomputing Centre (JSC) at Forschungszentrum Jülich (FZJ), provides nodes with and without GPUs. However, the billing for both types of nodes is based only on the number of CPU cores (”core-h”) used, even though the nodes with GPUs are much more powerful, as discussed in Sec. 2.5, and therefore a core-hour on a node equipped with GPUs should be more expensive. On the other hand, using FLOPs alone to evaluate computational cost is also inappropriate because CPU units are typically capable of more complex operations than GPU units.

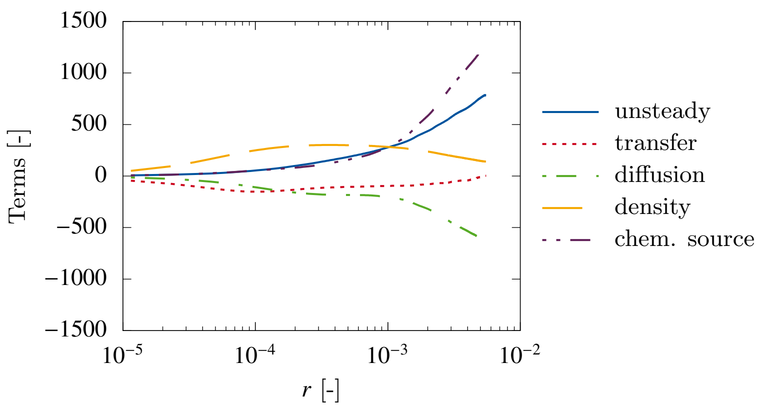

Figure 7: Scale-by-scale budget computed from DNS data.

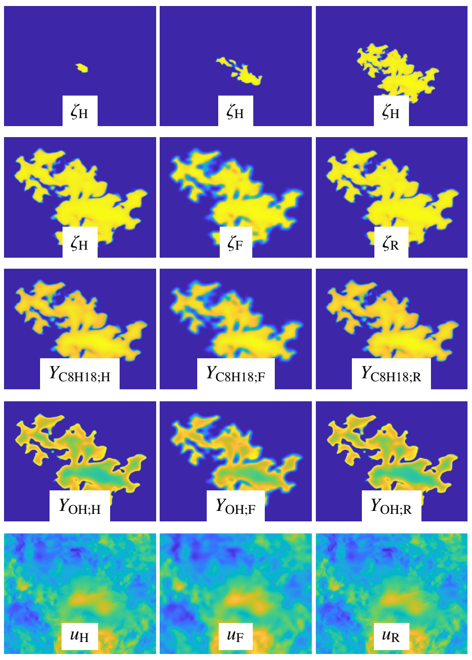

Figure 8: Visualization of one turbulent premixed flame kernel realization. The first row shows the temporal evaluation of the simplified progress variable ζ at 0.06 ms, 0.21 ms, and 0.36 ms. All other figures show a zoomed view of the fully resolved data, filtered data, and reconstructed data for the simplified reaction progress variable ζ, the C_8H_{18} mass fraction, Y_{C8H18}, the OH mass fraction, Y_{OH}, and a velocity component u employing PIESRGAN at 0.36 ms. Colormaps span from blue (minimum) to yellow (maximum).

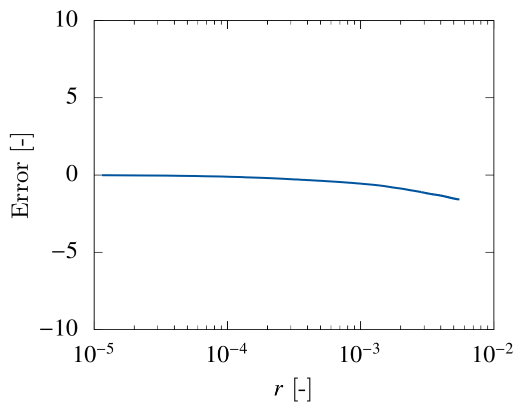

Figure 9: Closing error of scale-by-scale budget of reconstructed data compared to DNS data.

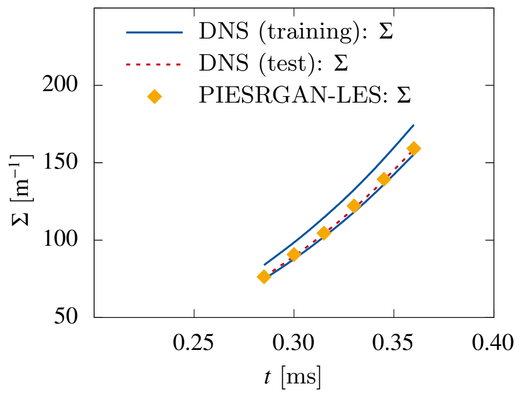

Figure 10: Evolution of the flame surface density Σ over time t.

4.5. Training Cost

As a data-driven method, PIESRGAN requires a training cost that is determined by the cost of computing the training data and the cost of the training itself. As described above, the training is done based on subsets of the whole domain. Therefore, a measure of training cost is given by the size of the data used for training and the number of subboxes that can be trained per GPU per minute, assuming nearly linear scaling for training with multiple GPUs. Bode [9] evaluated that more than 60 subboxes with 4096 cells per minute can be used for training on nodes equipped with two Tesla V100 GPUs. This means that with current GPUs, the cost of training itself is typically not the main cost factor for flow problems. In addition to the good training performance of modern GPUs, this is due to the often very expensive nature of turbulent flow simulations, in contrast to many other domains where training is relatively expensive. The training cost for the premixed case was 273×10^3 core-h (compared to 22×10^6 core-h for data generation). Note that the cost of training depends strongly on factors such as the time resolution of the simulation data and the total number of data samples used per time step. Furthermore, for practical applications, the computational time budget needs to account for divergent training trials, which are likely and can greatly increase

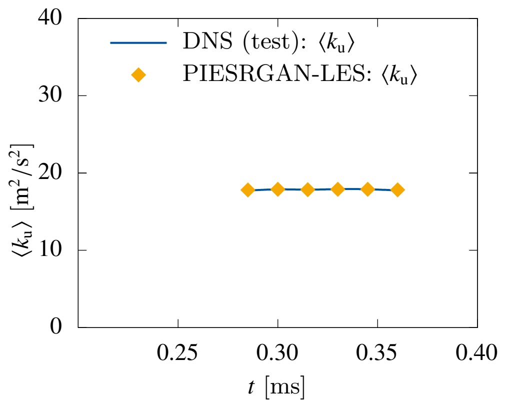

Figure 11: Evolution of the ensemble-averaged turbulent kinetic energy in the unburnt $\langle k_u \rangle$ over time t.

the cost.

The question of the cost of training data is more difficult to answer and highly case-dependent. There are many scenarios where usable data is already available. This could be from previous simulations, but also from experiments. However, it is important to note that not all existing data is useful for training, e. g., the time resolution of the stored data must be fine enough.

4.6. Computing Cost

Various metrics can be defined to access the computational cost, as different simulation workflows have different requirements. For time-critical simulations, e. g., simulations coupled to an experiment, the time-to-solution may be the most important factor, which is typically reduced by using more parallel cores, usually resulting in less efficient simulations due to an overhead caused by parallelization. The overhead can be caused by additional or slow communication between cores, nodes, or even clusters. If too many cores are used for a particular task, the actual computation time is relatively reduced compared to the communication time, which has some fixed timings that are independent of the number of cores involved. On the other hand, efficiency may be the most important metric for workflows that want to make the most of their limited computation time. Often, a trade-off between time-to-solution and efficiency is chosen. Another level of complexity is added when chaotic processes, such as turbulence, are involved in the simulations. Since the results are only statistically meaningful, enough data or ensembles must be computed to achieve convergent statistics. In this context, a simplified cost model for complex simulation workflows including ML/DL can be defined as

$$\begin{aligned} C_{\text{total}} =& n_{\text{ensemble}} \times C_{\text{simulation}}(n_{\text{timesteps}}, n_{\text{cells}}) \\ &+ C_{\text{trainingdata}} + C_{\text{trainingprocess}} \quad (15) \\ \approx& \mathcal{F}_{\text{C}} \times n_{\text{ensemble}} \times n_{\text{timesteps}} \times n_{\text{cells}} \\ &+ C_{\text{trainingdata}} + C_{\text{trainingprocess}} \quad (16) \end{aligned}$$

with n_{ensemble} as the number of ensemble simulations, $n_{\text{timesteps}}$ as the number of computed time steps per simulation, n_{cells} as the number of cells per simulation, $C_{\text{simulation}}$ as the cost per simulation, $C_{\text{trainingdata}}$ as the cost of computing the training data, and $C_{\text{trainingprocess}}$ as the cost of performing the PIESRGAN training. The Eq. (16) assumes a linear influence of the number of computed time steps and the number of cells on the simulation cost and introduces the cost factor $\mathcal{F}_{\text{C}}$. The linear dependence between the number of time steps and the cost is trivial. The linear effect of the number of cells is only valid as long as each core has enough work to do, i. e., if the ratio of cells per core is large enough, as for the runs in this work. The cost factor is an estimate of the relative computational cost of a setup and is, e. g., higher for a simulation with the PIESRGAN-subfilter model, since it requires more operations, than for the corresponding DNS without subfilter model.

For many turbulent problems, significant deviations between individual flow realizations can be found. An example of this is the turbulent premixed flame kernel simulations discussed above, which were performed explicitly to study variations in the context of CCVs. CCVs are important in engines, e. g., as they can have a significant impact on pollutant formation and unwanted pre-ignition or knocking. One cause of CCVs is the initial turbulent field at the beginning of the flame kernel. Therefore, multiple flame kernel simulations must be performed to analyze this effect. Thus, the task is to efficiently use a given amount of computational time to enable as many (accurate) realizations as possible. For this purpose, 40×10^6 core-h were available on JURECA-DC (Booster). As already mentioned, the cost per DNS realization turned out to be about 11×10^6 core-h (including on-the-fly training) and thus about four complete realizations would have been possible (without additional costs for on-the-fly training). To improve this, PIESRGAN-LESs were used with 7×10^6 cells (compared to 884×10^6 for the DNS) and a 1.25 times larger time step compared to the DNS. The cost factor was found to be 16, and two DNS realizations were computed for training to avoid lock-in effects on the initial state. Thus, about 17 realizations were possible given the 40×10^6 core-h, compared to less than 4 without the subfilter modeling approach. Also note that each additional realization also comes at the cost of 1.15×10^6 core-h compared to the DNS cost of 11×10^6 core-h.

5. Application

The discussed combined HPC/DL-workflow can be applied to quantitatively evaluate the flame kernel variations. Figure 12 shows the evolution of the flame surface density for several realizations. The effect of the different HIT conditions is evident. Interestingly, the PIESRGAN-LESs give results outside the training range, highlighting the advantage of PIESRGAN-LESs acting on the smallest scales and thus allowing variations on the larger scales. Finally, the variations of the flame surface density at 0.36 ms can be quantified as $155.0\,\text{m}^{-1} \pm 12.9\,\text{m}^{-1}$.

6. Conclusions

This paper demonstrates how a combined HPC/DL-workflow can drastically reduce the computational cost of obtaining statistical ensemble results. It is able to efficiently predict complex

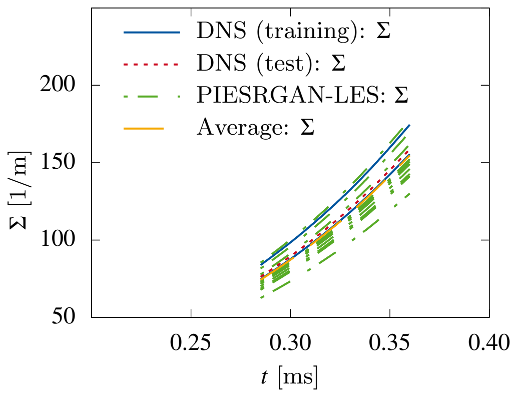

Figure 12: Evolution of the flame surface density Σ over time t for CCVs.

statistics of complex flows, which is usually not possible with classical subfilter models on current supercomputers. However, although the GAN architecture used gives good results, further research is needed to further optimize it and to better understand the impact of certain features on the prediction accuracy. In addition, the effect of hyperparameters, such as learning rate or model parameters, needs to be further investigated. All of these small choices can make a big difference in whether the learning process converges or diverges. There is still a lot of engineering work to be done to achieve results as accurate as those presented here. While the computer science community has developed a lot of practical experience in this area, the lack of such practical experience may be the most likely reason for failure for many domain scientists. Finally, this approach can only be as good as the underlying data. Therefore, it requires communities to be more honest about the accuracy and shortcomings of existing cases. Principles such as the use of FAIR (Findability, Accessibility, Interoperability, and Reuse) are an important step forward, but need to become more relevant in everyday science.

The turbulent combustion application case presented in this work was a carefully chosen example to demonstrate both the modeling power of AI-based super-resolution in fluid dynamics and the corresponding computational benefits. However, the presented modeling framework is not limited to such cases, and the range of possible applications, from weather prediction to drug development, seems endless.

Acknowledgements

The authors acknowledge computing time grants for the projects JHPC55 and TurbulenceSL by the JARA-HPC Vergabegremium provided on the JARA-HPC Partition part of the supercomputer JURECA at Jülich Supercomputing Centre, Forschungszentrum Jülich, the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for funding this project by providing computing time on the GCS Supercomputer JUWELS at Jülich Supercomputing Centre (JSC), and 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.

References

[1] S. B. Pope, Turbulent flows, Cambridge University Press, Cambridge, UK, 2000.
[2] H. Pitsch, Large-eddy simulation of turbulent combustion, Annual Review of Fluid Mechanics 38 (2006) 453–482.
[3] J. Smagorinsky, General circulation experiments with the primitive equations: I. The basic experiment, Monthly Weather Review 91 (3) (1963) 99–164.
[4] M. Bode, M. Gauding, K. Kleinheinz, H. Pitsch, Deep learning at scale for subgrid modeling in turbulent flows: regression and reconstruction, Lecture Notes in Computer Science 11887 (2019) 541–560.
[5] M. Bode, M. Gauding, Z. Lian, D. Denker, M. Davidovic, K. Kleinheinz, et al., Using physics-informed enhanced super-resolution generative adversarial networks for subfilter modeling in turbulent reactive flows, Proceedings of the Combustion Institute 38 (2021) 2617–2625.
[6] M. Bode, Applying physics-informed enhanced super-resolution generative adversarial networks to finite-rate-chemistry flows and predicting lean premixed gas turbine combustors, arXiv preprint arXiv:2210.16219 (2022).
[7] M. Bode, M. Gauding, D. Goeb, T. Falkenstein, H. Pitsch, Applying physics-informed enhanced super-resolution generative adversarial networks to turbulent premixed combustion and engine-like flame kernel direct numerical simulation data, Proceedings of the Combustion Institute (2023).
[8] M. Bode, Applying physics-informed enhanced super-resolution generative adversarial networks to turbulent non-premixed combustion on non-uniform meshes and demonstration of an accelerated simulation workflow, arXiv preprint arXiv:2210.16248 (2022).
[9] M. Bode, Applying physics-informed enhanced super-resolution generative adversarial networks to large-eddy simulations of ECN Spray C, SAE International Journal of Advances and Current Practices in Mobility 4 (2022) 2211–2219.
[10] M. Bode, AI super-resolution-based subfilter modeling for finite-rate-chemistry flows: A jet flow case study, SAE Technical Paper 2023-01-0200 (2023).
[11] M. Bode, AI super-resolution: Application to turbulence and combustion, in: N. Swaminathan, A. Parente (Eds.), Machine learning and its application to reacting flows, Lecture Notes in Energy 44, Springer, 2023.
[12] M. Bode, AI super-resolution subfilter modeling for multi-physics flows, Platform for Advanced Scientific Computing Conference (PASC '23).
[13] C. Dong, C. C. Loy, K. He, X. Tang, Learning a deep convolutional network for image super-resolution, in: European Conference on Computer Vision, Springer, 2014, pp. 184–199.
[14] C. Dong, C. C. Loy, K. He, X. Tang, Image super-resolution using deep convolutional networks, IEEE Transactions on Pattern Analysis and Machine Intelligence 38 (2) (2015) 295–307.
[15] J. Kim, J. Kwon Lee, K. Mu Lee, Accurate image super-resolution using very deep convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1646–1654.
[16] J. Kim, J. Kwon Lee, K. Mu Lee, Deeply-recursive convolutional network for image super-resolution, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1637–1645.
[17] W. Lai, J. Huang, N. Ahuja, b. p. y. Yang, M.-H., Deep Laplacian pyramid networks for fast and accurate super-resolution.
[18] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556 (2014).
[19] J. Johnson, A. Alahi, L. Fei-Fei, Perceptual losses for real-time style transfer and super-resolution, in: European Conference on Computer Vision, Springer, 2016, pp. 694–711.
[20] Y. Tai, J. Yang, X. Liu, C. Xu, MemNet: A persistent memory network for image restoration, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 4539–4547.
[21] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, Y. Fu, Image super-resolution using very deep residual channel attention networks, in: Proceedings of the European Conference on Computer Vision, 2018, pp. 286–301.
[22] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al., Photo-realistic single image super-resolution using a generative adversarial network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4681–4690.

[23] I. Goodfellow, J. Pouget-Agadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative Adversarial Networks, arXiv preprint arXiv:1406.2661 (2014).

[24] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, C. Change Loy, ESRGAN: Enhanced super-resolution generative adversarial networks, in: Proceedings of the European Conference on Computer Vision, 2018.

[25] K. Stengel, A. Glaws, D. Hettinger, R. N. King, Adversarial super-resolution of climatological wind and solar data, Proceedings of the National Academy of Sciences 117 (2020) 16805–16815.

[26] Y. Li, Y. Ni, R. A. C. Croft, T. Di Matteo, S. Bird, Y. Feng, AI-assisted superresolution cosmological simulations, Proceedings of the National Academy of Sciences 118 (2021) e2022038118.

[27] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.

[28] A. L. Maas, A. Y. Hannun, A. Y. Ng, Rectifier nonlinearities improve neural network acoustic models, Proceedings of the 30th International Conference on Machine Learning 30 (2013).

[29] M. Bode, A. Deshmukh, T. Falkenstein, S. Kang, H. Pitsch, Hybrid scheme for complex flows on staggered grids and application to multiphase flows, Journal of Computational Physics 474 (2023) 108478.

[30] O. Desjardins, G. Blanquart, G. Balarac, H. Pitsch, High order conservative finite difference scheme for variable density low Mach number turbulent flows, Journal of Computational Physics 227 (15) (2008) 7125–7159.

[31] M. Bode, M. Davidovic, H. Pitsch, Towards clean propulsion with synthetic fuels: Computational aspects and analysis, in: High-Performance Scientific Computing, Springer Nature, 2019, pp. 185–207.

[32] M. Bode, T. Falkenstein, S. Kang, H. Pitsch, High-Q Club, http://www.fz-juelich.de/ias/jsc/EN/Expertise/High-Q-Club/CIAO/_node.html (2015).

[33] R. D. Falgout, U. M. Yang, hypre: A library of high performance preconditioners, in: P. M. A. Sloot, A. G. Hoekstra, C. J. K. Tan, J. J. Dongarra (Eds.), Computational Science — ICCS 2002, Springer Berlin Heidelberg, 2002, pp. 632–641.

[34] V. E. Henson, U. M. Yang, BoomerAMG: A parallel algebraic multigrid solver and preconditioner, Applied Numerical Mathematics 41 (1) (2002) 155–177.

[35] G.-S. Jiang, C.-W. Shu, Efficient implementation of weighted ENO schemes, Journal of Computational Physics 126 (1) (1996) 202–228.

[36] G. Strang, On the construction and comparison of difference schemes, SIAM Journal on Numerical Analysis 5 (3) (1968) 506–517.

[37] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, C. S. Woodward, SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers, ACM Transactions on Mathematical Software 31 (3) (2005) 363–396.

[38] P. N. Brown, G. D. Byrne, A. C. Hindmarsh, VODE: A variable-coefficient ODE solver, SIAM Journal on Scientific and Statistical Computing 10 (5) (1989) 1038–1051.

[39] J. O. Hirschfelder, C. F. Curtiss, R. B. Bird, Molecular theory of gases and liquids, John Wiley and Sons, New York, 1954.

[40] K. N. C. Bray, P. A. Libby, J. B. Moss, Unified modeling approach for premixed turbulent combustion—Part I: General formulation, Combustion and Flame 61 (1) (1985) 87–102.

[41] U. Frisch, Turbulence - The legacy of A. N. Kolmogorov, Cambridge University Press, Cambridge, UK, 1995.

[42] R. Kirste, G. Porod, Röntgenkleinwinkelstreuung an kolloiden Systemen Asymptotisches Verhalten der Streukurven, Kolloid-Zeitschrift und Zeitschrift für Polymere 184 (1) (1962) 1–7.

[43] T. Kulkarni, R. Buttay, M. H. Kasbaoui, A. Attili, F. Bisetti, Reynolds number scaling of burning rates in spherical turbulent premixed flames, Journal of Fluid Mechanics 906 (2021).

[44] N. Peters, The turbulent burning velocity for large-scale and small-scale turbulence, Journal of Fluid Mechanics 384 (1999) 107–132.

[45] M. Gauding, F. Thiesset, H. Pitsch, M. Bode, Morphology and kinematics of turbulent reactive fronts and application to premixed combustion, submitted (2023).

[46] T. Falkenstein, S. Kang, L. Cai, M. Bode, H. Pitsch, DNS study of the global heat release rate during early flame kernel development under engine conditions, Combustion and Flame 213 (2020) 455–466.

[47] T. Falkenstein, A. Rezchikova, R. Langer, M. Bode, S. Kang, H. Pitsch, The role of differential diffusion during early flame kernel development under engine conditions - Part I: Analysis of the heat-release-rate response, Combustion and Flame 221 (2020) 502–515.

[48] T. Falkenstein, H. Chu, M. Bode, S. Kang, H. Pitsch, The role of differential diffusion during early flame kernel development under engine conditions - Part II: Effect of flame structure and geometry, Combustion and Flame 221 (2020) 516–529.