Energy and AI 12 (2023) 100216

Contents lists available at ScienceDirect

Energy and AI

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

Conditional Generative Adversarial Networks for modelling fuel sprays

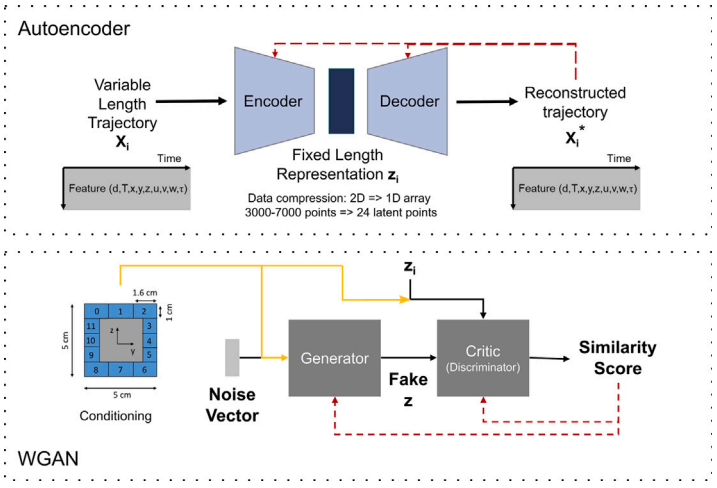
Cihan Ates [a,b,*], Farhad Karwan [a], Max Okraschevski [a], Rainer Koch [a], Hans-Jörg Bauer [a]

[a] Karlsruhe Institute of Technology (KIT), Institute of Thermal Turbomachinery, Germany
[b] Karlsruhe Institute of Technology (KIT), Machine Intelligence in Energy Systems, Germany

HIGHLIGHTS

- GANs were combined with autoencoders (AE) to conduct virtual spray simulations.
- AE converts variable length droplet trajectories into fixed length representations.
- Conditioned GANs mimic the latent representations of the evaporating droplets.
- Training data was provided from highly resolved Eulerian-Lagrangian LES simulations.
- Predictive accuracy highly depends on encoding methodology.

GRAPHICAL ABSTRACT



ARTICLE INFO

Keywords:
Generative Adversarial Networks
Generative learning
Fuel injection
Aero engines
Multivariate time series

ABSTRACT

In this study, the probabilistic, data driven nature of the generative adversarial neural networks (GANs) was utilized to conduct virtual spray simulations for conditions relevant to aero engine combustors. The model consists of two sub-modules: (i) an autoencoder converting the variable length droplet trajectories into fixed length, lower dimensional representations and (ii) a Wasserstein GAN that learns to mimic the latent representations of the evaporating droplets along their lifetime. The GAN module was also conditioned with the injection location and the diameters of the droplets to increase the generalizability of the whole framework. The training data was provided from highly resolved 3D, transient Eulerian–Lagrangian, large eddy simulations conducted with OpenFOAM. Neural network models were created and trained within the open source machine learning framework of PyTorch. Predictive capabilities of the proposed method was discussed with respect to spray statistics and evaporation dynamics. Results show that conditioned GAN models offer a great potential as low order model approximations with high computational efficiency. Nonetheless, the capabilities of the autoencoder module to preserve local dependencies should be improved to realize this potential. For the current case study, the custom model architecture was capable of conducting the simulation in the order of seconds after a day of training, which had taken one week on HPC with the conventional CFD approach for the same number of droplets (200,000 trajectories).

* Corresponding author at: Karlsruhe Institute of Technology (KIT), Institute of Thermal Turbomachinery, Germany.
E-mail address: cihan.ates@kit.edu (C. Ates).
URL: https://www.its.kit.edu (C. Ates).

https://doi.org/10.1016/j.egyai.2022.100216
Received 28 July 2022; Received in revised form 16 November 2022; Accepted 19 November 2022
Available online 21 December 2022

1. Introduction

In recent years, the impact of the aviation industry on the environment has become a major concern, where the legal constraints push the industry to further improve the design of each component with an environmentally friendly target. One key aspect here is the fuel injection into the pressurized combustion chamber, as it determines the rate of heat transfer, evaporation and the chemical reactions, which in turn dictate how clean and efficient the energy conversion process is. Therefore, spray analysis and its accurate representation is of critical importance for new engine design studies. Accessing this information, however is a difficult and expensive process. Conducting high fidelity simulations is a common strategy in the community to capture spray dynamics, typically in the Lagrangian frame by tracing every droplet bundle in the simulation domain. The liquid phase physics are then coupled with the Eulerian model of the gas phase for every conservation law being solved. Herein, one critical computational bottleneck is the Lagrangian phase calculations and its connection to the continuum, which has to be re-computed for every case being modelled in a design study. In a typical workflow, the costly know-how generated for droplet dynamics is utilized only once and has to be re-learned and utilized at every design iteration. Therefore, there is a need for low order but accurate representations of the spray dynamics.

One of the advancing frontiers of machine learning (ML) is generative modelling, which relies on representation learning. Such models have a probabilistic nature and rely on the discovered patterns in the data that can later be used to create new examples indistinguishable from the original dataset. In particular, Generative Adversarial Neural Networks (GANs) enable an implicit representation of the probability distributions, connecting the features present in the training dataset. Since its introduction by Goodfellow in 2014 [1], it has become a popular method in both academia and industry for synthetic data generation. Common applications are image/video processing [2,3], style transfer [4], synthetic scenery/music generation [5–7], speech enhancement [8], reinforcement learning [9], fault detection in gear systems [10], heterogeneous catalyst design [11], film cooling effectiveness in turbine blades [12], subfilter turbulence modelling [13] and its application in a planar turbulent jet flow [14]. A detailed comparative review of deep generative models including GANs could be found in [15,16].

While the capabilities of GANs have been extensively investigated, relatively few studies exist on time series modelling. TimeGAN [17] is one of the early examples of sequential data generator, in which an autoencoder module and a GAN module are combined. Herein, the autoencoder learns to create a lower-dimensional representation of the original fixed length sequence and the GAN model learns to create fake examples in latent (lower dimensional representation) space from white noise of the same sequence length. In TimeGAN, both modules are trained jointly, so that the overall model learns to encode features and generate samples at the same time. The workframe was later extended to handle more challenging time series problems including missing data and variable sequence lengths [18]. Recent GAN implementations for generating sequential data include medical time series (ICU) [19], chaotic systems [20], data generation for smart grids [21], spot price (S&P 500) [22], air processing system in test trucks [23], energy consumption at state level [24], energy use in buildings [25] and load scenario generation [26]. A detailed survey and taxonomy of GAN variants designed for time series related applications can be found in [27,28].

In this work, we deployed the probabilistic, data driven nature of the GANs to mimic the statistics of fuel droplets in aero engine combustors, by learning the underlying laws that lead to the observed droplet trajectories. The overall objective is to create a probabilistic model that can generate synthetic fuel sprays, while conserving the complex transient relations between the features such as velocity, temperature and diameter. The custom GAN architecture is based on two particular models: (i) RTSGAN [18] that is capable of learning temporal relationships and (ii) cGAN [29] which imposes conditional probability distributions on the learning process, in an attempt to increase the generalizability of the developed model. This modification allows to generate artificial droplet trajectories with different initial droplet size distributions at different initial positions, so that it can be deployed to generate alternative spray scenarios. The training data was provided from highly resolved Eulerian–Lagrangian large eddy simulations for evaporating spray clouds conducted within the scope of this work. Data preparation was done by post-processing the voluminous simulation data with in-house developed scripts to filter out droplet trajectories with several temporally resolved features such as position, velocity components, diameter and temperature of individual droplet bundles as function of the residence time within the combustion chamber. Learnt representations of variable length droplet trajectories were investigated for a variety of pattern recognition model architectures (i.e., autoencoders) and compared via principle component analysis. The spray statistics of the real and generated trajectories were analysed and the success of conditioning strategy is discussed.

2. Methods

2.1. Generating the training data for GANs

The droplet trajectories for the training of the ML models were generated by conducting Euler–Lagrangian simulations, which mimic the conditions of an aero engine combustor. For this study, the combustion chamber geometry was approximated by a rectangular domain (Fig. 1), where the fuel injector is represented by a bluff body near the gas inlet. The dimensions of the chamber were set according to the annular combustor designs utilized in modern engines. The shape and the size of the bluff body were determined based on an iterative procedure to create a central recircular zone (CRZ), which on average is similar to those generated with the real fuel injector geometry (Fig. 1).

The investigated case is non-reactive and physics of the case study, as well as the boundary conditions for the gas and the liquid phases are selected to reflect late relight conditions. This operating point is selected for two reasons. (i) The main goal of the study is to investigate how to create synthetic multivariate time series data of different lengths (i.e. droplet trajectories) efficiently and accurately, which can be still tested in a non-reactive problem. In accordance, through the selected operating point, the combustor could be approximated as a pseudo-inert system, where the majority of the injected fuel reaches to combustor exit unreacted. (ii) At the selected late relight conditions, the gas phase temperature is high enough to increase the complexity of feature space; that is, some of the liquid fuel can still evaporate, particularly the fine fraction of the droplet size distribution, such that there is still a complex physical coupling between the droplet size, temperature and the trajectory of the droplets (particularly changing drag behaviour with changing diameter).

The gas phase is modelled as air, while the liquid fuel is represented as n-Dodecan ($C_{12}H_{26}$), which is considered as a surrogate for the kerosene. The inlet gas temperature is set to 515 K, entering the annular domain at 10 m/s. Transient, two-phase flow simulations were conducted via open-source CFD code OpenFOAM by using the *sprayFoam* solver, which is capable of solving the governing equations required for transient, compressible and turbulent flows in the presence of a Lagrangian particle cloud (time scheme: Eulerian, gradient: Gauss linear, divergence: Gauss upwind, Laplacian: Gauss linear orthogonal, cell to face interpolations: linear. Solvers; density: PCG, pressure: GAMG, species: PBiCGStab.).

The turbulence model was selected after conducting preliminary single phase simulations with the same setup by comparing the URANS and LES models available in OpenFOAM environment. As the URANS does not bring much computational efficiency compared to Wall Adapting Local Eddy-viscosity (WALE) LES while losing information on instantaneous flow fluctuations, LES was utilized in the extended two

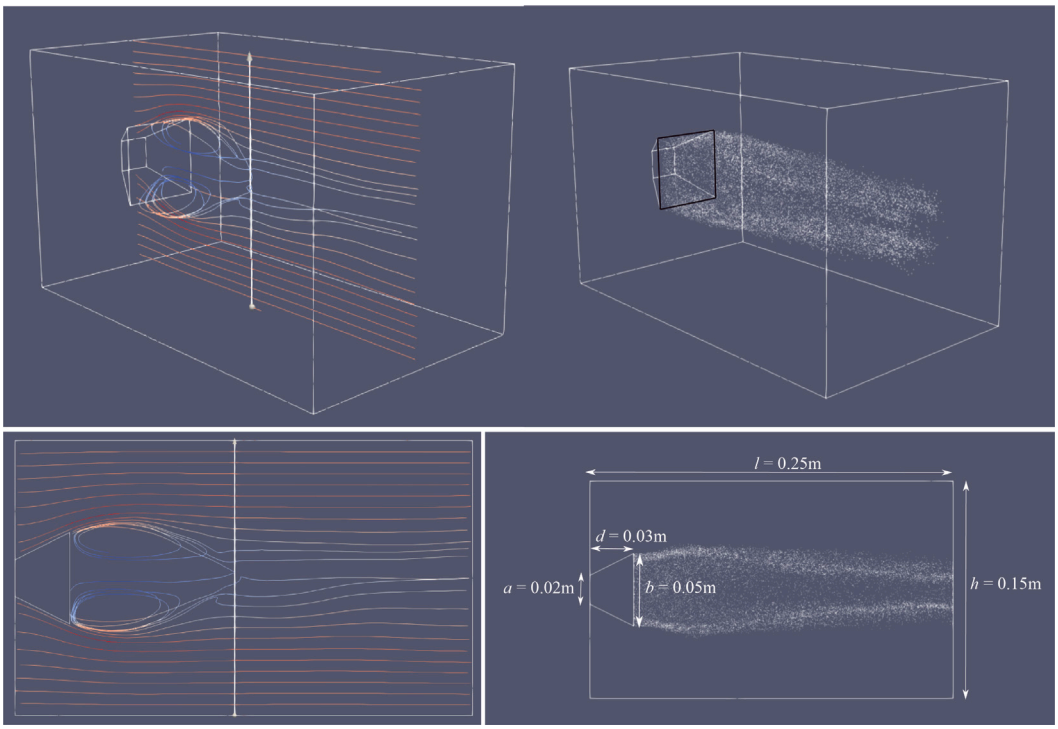


Fig. 1. Rectangular solution domain with the evolved coherent structures and the droplet trajectories.

Table 1
Spray injection parameters. Initial droplet temperature corresponds to the temperature after the secondary atomization. Fuel injection temperature is considered as 353 K.

Fuel name	$C_{12}H_{26}$
Droplet temperature	415 K
Injection duration	0.7 s
Fuel mass flow rate	0.06 kg s^{-1}
Initial droplet velocity	5 m s^{-1}
Parcels per second	400,000

Table 2
Rosin–Rammler distribution for the initial droplet sizes. Spray data was calculated via smoothed particle hydrodynamics simulations with in-house developed code (turboSPH) [30].

Reference Diameter	100×10^{-6} m
Width parameter	2.5
Lower diameter value	80×10^{-6} m
Upper diameter value	200×10^{-6} m

Table 3
LES parameters.

Subgrid scale model	WALE model
Mesh size (unstructured)	3 mm
Courant Number	0.1
Time step size	1×10^{-6} s
Solution writing interval	1×10^{-4} s

phase flow simulations. All forces acting on the fuel droplets except the drag forces are neglected. To account for the fuel evaporation, the Stefan–Maxwell model is applied to calculate instantaneous mass transfer rate.

The liquid fuel droplets are injected from the edges of the square injection patch (black lines) shown in Fig. 1. Tables 1–3 summarize the spray injection settings, initial droplet size distributions and LES model parameters, respectively. The droplet size distribution is based on the SPH simulations of the atomization process (see Table 2) and it is sampled homogeneously at the injection patch while creating the droplet parcels. It should also be noted that these initial conditions represent the distribution after the secondary atomization process.

Once the CFD simulations were completed, the relevant Lagrangian phase variables (droplet ID, diameter, temperature, velocity and position) were filtered for each time step. The raw data in the binary format was then converted into the VTK format, which can be processed conveniently in Python. In the next step, min–max-scaling was applied on the data, a common strategy in data driven, distance-based learning methods. Finally, a numpy array was created that stored the trajectory information of each droplet for its entire life time in the combustion chamber by using the droplet ID, which is a unique feature for each droplet. The size of the time dimension of the droplet trajectory arrays typically changes in between 300–700 (i.e., time-wise depth of the 2D array), depending on its evaporation and trajectory dynamics. In total, 200,000 trajectories were extracted, which can be downloaded via the link given in *Data Availability*.

2.2. Conditional time series GANs

The objective of the current study is to learn the statistics of fuel droplets in aero engine combustors by learning the underlying laws that lead to the observed droplet trajectories, while conserving the transient dependencies between the features (position, diameter, temperature, velocity). This was achieved by combining two neural network models: an autoencoder and a Wasserstein-GAN (WGAN). The general architecture of the model is illustrated in Fig. 2. Droplet trajectories were used only to train the autoencoder model, independently from the GAN model. The autoencoder consists of two sub-neural networks: an encoder and a decoder. During the training, droplet trajectories are fed to the encoder model. It consists of a N-layer GRU network with hidden dimensions d_{AE}. Because of their inherent properties, GRU cells can learn and keep the sequential dependencies of the time series in their state function, overcoming the common memory problem of vanilla RNN cells. The objective of the encoder is to simplify the generative

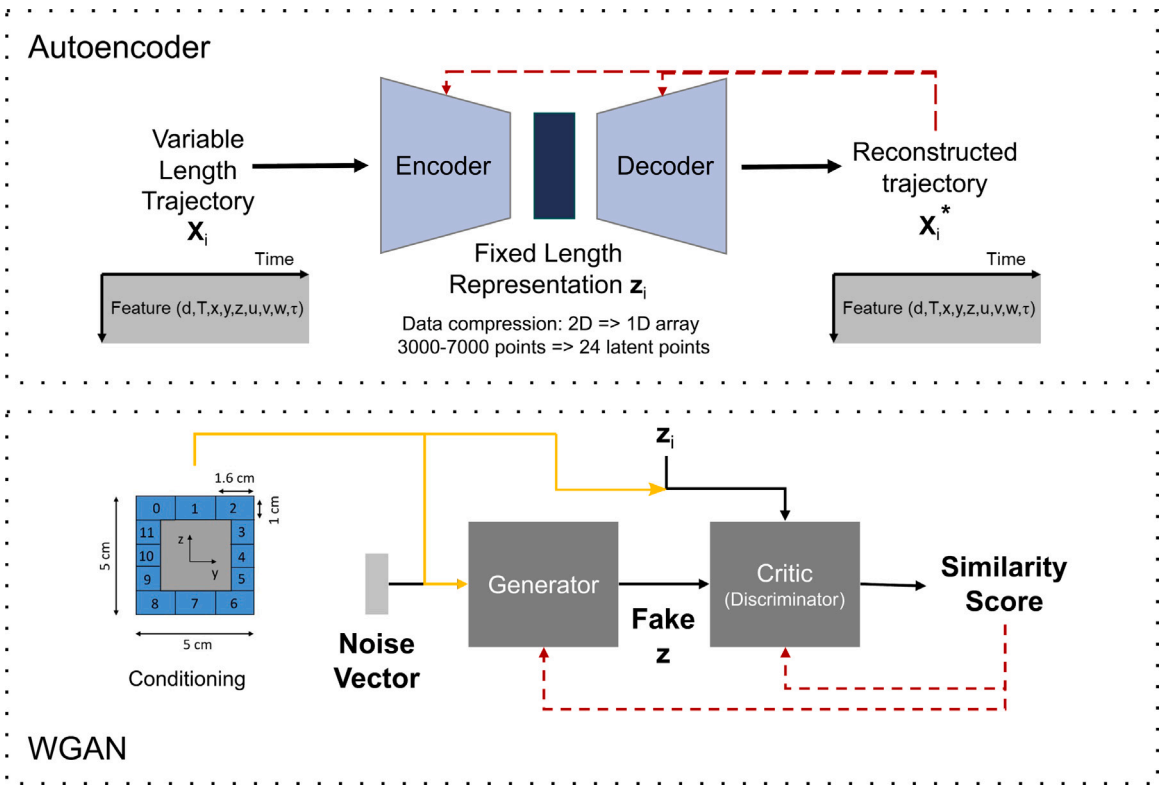


Fig. 2. Implemented model architecture. Top: autoencoder, bottom: WGAN model.

learning process, by compressing variable length droplet trajectories into fixed-length latent representation $\mathbf{z_i}$. In other words, the dimension of the latent vector is independent from the sequence length. The role of the decoder, on the other hand, is to learn how to unroll a compressed data into a variable length sequence. The training of the autoencoder is done via reconstruction loss of the droplet trajectories ($\mathbf{X_i} - \mathbf{X_i^*}$). It should be noted that each time series instance that is fed to the model contains two types of features: global features and dynamics features ($\mathbf{X_i}$). Global features do not change over time and are thus constant for a sequence sample (they are extracted from the dynamic features). Examples for global features are class labels or properties of the sequence itself, such as the sequence length. On the other hand, dynamic features of the sequence (diameter, position, velocity, temperature) change over time. A training instance is the concatenation of the data from both feature types. The global feature of sequence length is of particular importance, as it makes decoding the compressed fixed length data back to its original shape possible. This is reflected into the loss function as a linear combination of reconstruction loss for global features and dynamics features. Cross-entropy and mean squared error losses are used for categorical features and continuous features, respectively. The detailed procedure of creating the latent vector $\mathbf{z}$ can be found in [18].

The generative part of the whole modelling framework is the GAN, which tries to create realistic compressed representations ($\mathbf{z_i}$) from random noise vectors. It is built from two competing neural networks, referred as the generator G and the discriminator D. The task of the generator is to create "real-like" samples from a noise vector z, in an attempt to fool the discriminator model. In contrast, the task of the discriminator is to judge whether the sample is real or fake (synthetically generated by the generator). This adversarial process can be regarded as a min–max game between the generator and discriminator. In the vanilla GAN, the objective function can be expressed as follows:

$$\min_G \max_D \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[1 - \log D(G(z))] \quad (1)$$

where $\mathbb{E}$ represents the expected value or the expectancy. The discriminator is trained such that it maximizes the probability of assigning the true label to an incoming sample $\log(D(x))$, whereas the generator is trained by minimizing $\log(1 - D(G(z)))$. The iterative updating of each network's weights (i.e., model parameters) is done via back propagation algorithm. A practical interpretation here is that the discriminator acts like a learned loss function for the generator. Since it is not possible to define a loss function a priori how a real sample looks like, GANs aim to learn it implicitly as a parameterized model through the discriminator (also referred as critic in the community depending on the definition of loss function). In the current work, the Wasserstein GAN (WGAN) was used as the generative model for a stable training [31]. The Wasserstein distance between the fake and the real examples in latent space is used as the feedback for the training, with the weight clipping implementation of [32].

In order to create droplet trajectories of variable lengths, the RTS-GAN implementation [18] was altered. The custom implementation can be found in Supplementary Material 1. There are two major updates in the code. First, the original model is tuned such that variable length sequences can be reconstructed to reflect different residence times of droplets in the combustion chamber. For instance, the residence time of a droplet can change drastically depending on its own velocity, initial diameter and the injection position, as well as the transient gas phase flow dynamics. It is important to mention that the authors of RTSGAN published two different versions of their code. One version is suited for complete and fixed-length sequences (C1), while the other (C2) is applied for incomplete and variable-length sequences. In this work, the C1 code version was modified to be able to work with variable sequence lengths. This was done by creating a hybrid version from C1 and C2. One key change was adding a global feature containing information about sequence length to each training instance of the training data set as a pre-processing step. This global feature is simply an integer representing sequence length. In the encoder model, this information gets embedded in the latent space representation $\mathbf{z_i}$. Hence, it can be decoded back to the correct sequence length via decoder model after

Table 4
Hyperparameters tested for the autoencoder module with the lowest reconstruction loss after their training.

RNN cell architecture	Number of hidden dimensions ($\mathbf{z_i}$)	Loss value
GRU	16	1e−4
GRU	24	5e−5
GRU	48	6e−4
LSTM	24	5e−5
LSTM	32	5e−5

the training. When the GAN is used to generate fake examples, it can still generate trajectories with varying lengths as it is interpreted as a part of the learnt latent space representation by the Generator.

In order to have a control on the initial state of the spray, a further modification of the RTSGAN was needed. As stated in the introduction section, the goal of this work is to create a spray model, where droplet trajectories can be conditioned on initial diameter and initial position. The existing GAN architecture was therefore transformed into a conditional GAN (CGAN). The CGAN is an extension of the vanilla approach [1] and was first introduced in 2014. The purpose of a CGAN is to control the output of the model by conditioning the generator and discriminator with additional information c, in order to generate samples from a prescribed class. The objective function of CGAN becomes:

$$\min_G \max_D \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x|c)] + \mathbb{E}_{z\sim p_z(z)}[1 - \log D(G(z|c))] \quad (2)$$

This additional information can either be a class label or be data from other modalities. It is integrated into both the discriminator and generator via an additional input layer [29]. For the droplet trajectory problem, we labelled each sequence with a class label indicating either the injection location (i.e., initial position) or the droplet size. The class label, which is implemented as an integer, serves as a conditional information during training and inference. In the model, the class label is fed into an embedding layer, which transforms the input into a 10-dimensional vector. The conditional vector $\mathbf{c}$ is then concatenated together with the noise vector. Finally, the resulting vector is fed into the generator. During the training of the discriminator, the conditional vector $\mathbf{c}$ is also concatenated together with the hidden representation of the real or fake samples. In summary, conditioning allows to ask the Generator model not only to create a realistic droplet trajectory, but one starts at a given position and/or a defined size.

2.3. Model experiments and training

For the training of the autoencoder and the WGAN, the trajectory dataset (200,000 instances) were down-sampled to 20,000 via two different policies. At first, the trajectories were randomly sampled irrespective of their initial conditions. The models are then tested with another randomly selected sub-set of 20,000 trajectories. Then, we first classify the trajectories based on their initial properties (injection patch location, diameter) into 32 classes and then down-sample them in a way that they have the same probability distribution with the large initial database. The models trained with the second policy was also tested on randomly selected test set of same size. In both cases, the predictive accuracy of the models stays the same, irrespective of how the training set is created. After this initial observation, a set of different hyperparameters was tested for both the autoencoder module and GAN module for a reduced training set to speedup the training times, particularly for the autoencoder. Herein, the RNN cell architecture (LSTM/GRU) and the number of hidden dimensions were varied as shown in Table 4. The number of training epochs was set as 1000 for each experiment and the best set of weights achieved during the training was saved. The best-performing Encoder variant was then used to create training examples in latent space ($\mathbf{z_i}$) for the GAN module.

The hyperparameter testing for the GAN module was limited to the learning rate. For the current case study, lower learning rates were found to perform poorly (<1e-4) and the best training performance was achieved with a learning rate of 1e-3. The other hyperparameters were kept as recommended in the RTSGAN study [18].

The training of both the autoencoder and GAN modules was conducted on bwUniCluster using one GPU (NVIDIA Tesla V100) and all 14 processor nodes (Intel Xeon Gold 6230). The training of the autoencoder and the GAN modules took 24 h and 12 h, respectively. Once the model is trained, it took about 10 s to generate a full spray simulation on a typical office PC.

3. Results

3.1. Autoencoder evaluation for reconstructing the droplet trajectories

The autoencoder module aims to minimize the difference between the input tensor $\mathbf{X_i}$ (time and feature dimensions of the trajectory) and the output tensor $\mathbf{X_i^*}$ (Fig. 2). This difference is typically referred as the reconstruction error. A small reconstruction error indicates that the autoencoder is capable of compressing the data into a lower-dimensional space by maintaining most of the information. However, this loss value is a global measure and only gives the average loss over the whole trajectory (tensor) and does not provide any information about the spatial and temporal dependencies of the features (or where the error is originated from). Therefore, an additional statistical analysis for individual properties (diameter, velocity, temperature, position) at different locations is needed to extract more specific information about the reconstruction of each single feature. In this work, as an additional metric, the global statistics at four different equidistant locations are evaluated by constructing the probability density functions (PDFs) of each physical feature. These functions are then compared with the original density functions at the same locations. The similarity between the learnt and the reconstructed PDFs shows the capability of the autoencoder model to learn and preserve the temporal and spatial correlations in the trajectories. In order to quantify whether the autoencoder is capable of capturing the evaporation physics properly or not, which is not directly visible in the features or the PDFs, a third metric is also introduced. Herein, coupled thermal effects (rate of evaporation and expansion) are estimated by comparing the incremental change in droplet volumes between two relatively close evaluation planes at preselected locations in the simulation domain (i.e, a differential volume element). As a final comparison, reconstructed trajectory examples are presented. This metric is for qualitative purposes and allows a more practical and intuitive view on the performance of the model.

The reconstruction loss for the trajectories was determined by applying Mean-Squared-Error for a batch of samples. After the training, the highest reconstruction error was in the order of 1e-4 in all cases in both training and test sets, indicating that the sequence reconstruction process is globally accurate. The worst performance was given by GRU (48) as 6e-4 (Table 4). As highlighted before, the reconstruction loss does not provide much insight about the performance of the autoencoder, as it only measures the overall success of the reconstructed 2D array. Hence, it cannot describe how much of the temporal or pairwise relations (e.g. diameter vs. temperature) are preserved. In other words, we need to deploy alternative metrics to quantify the predictive accuracy of the data conversion process from variable length, multivariate time series data to fixed-length, latent vector representations.

In an attempt to quantify how well the casual relations are learnt in the autoencoder module, we first compared the global spray statistics over 20,000 droplet trajectories for the original and reconstructed samples. The tested trajectories were sampled randomly from the main example pool (200,000 samples). The statistical consistency is checked by comparing the constructed probability density functions (PDFs) for each feature on four equidistant evaluation planes perpendicular to the main flow direction (0.033 m, 0.103 m, 0.173 m and 0.243 m).

Fig. 3 illustrates several examples for droplet diameters, temperature, one velocity component and one position feature. PDFs for all

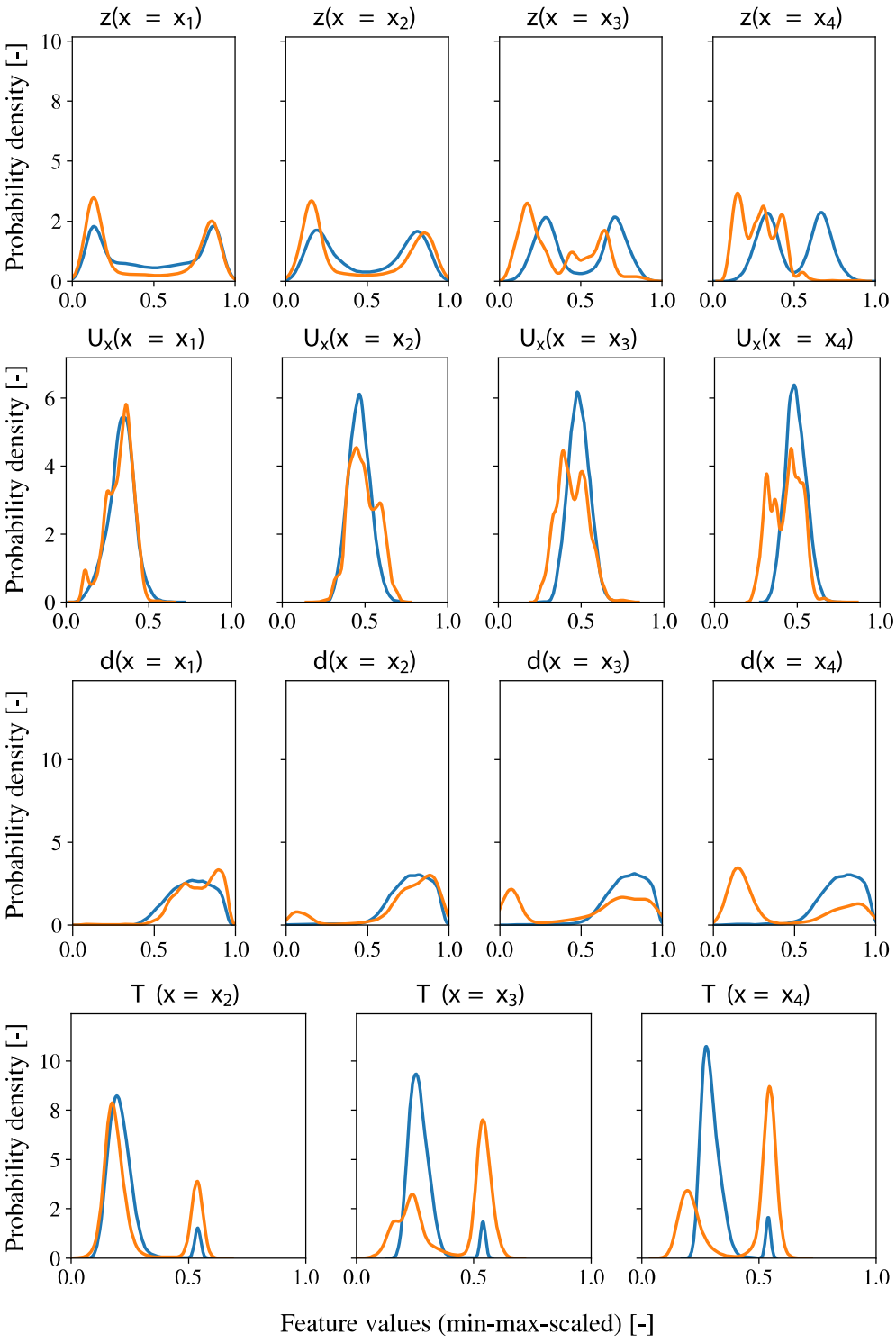


Fig. 3. Global spray statistics for the original (blue) and reconstructed data (orange) for droplet diameters, temperature, one velocity component and one position feature. PDFs are calculated from 20,000 trajectories at plane locations: $x_1 = 0.033$ m, $x_2 = 0.103$ m, $x_3 = 0.173$ m, $x_4 = 0.243$ m. The rest can be found in Supplementary Material 2.

tested autoencoder models at all four planes can be found in Supplementary Material 2. Irrespective of the deployed RNN cell type (GRU/LSTM) or the number of neurons of the hidden state, PDFs of the original and the reconstructed data showed noticeable deviations and the PDFs of the reconstructed data could only match the original data in few planes. The reconstruction errors were typically smaller at planes closer to the injection location and deviates more from the original sequence at further downstream locations, corresponds to later residence times.

In order to assess the similarity between the original PDFs and the reconstructed PDFs, we used a simple, binary classification approach. If the reconstructed and original PDFs are similar for a given feature (e.g., diameter) at a given plane (e.g x_1), the method get a score of +1. For instance, a value of 4 indicates that the model was capable of reproducing accurate PDFs for all four planes for that particular feature (see Supplementary Material 2 for visual PDF comparisons for all features and models). If the distribution is not captured qualitatively, we give a score of zero. Herein, similarity means a Hellinger distance smaller than

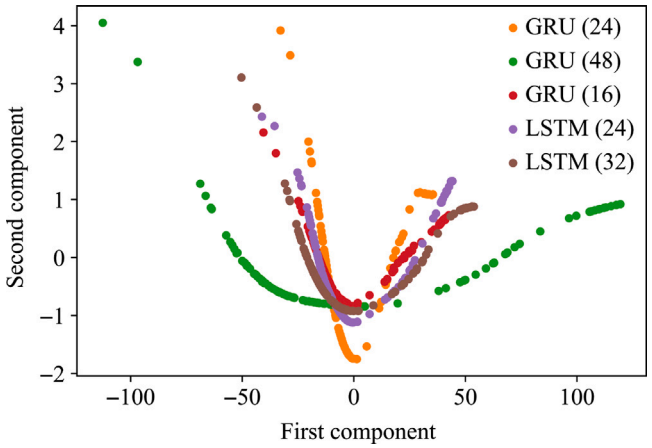


Fig. 4. Comparison of first two principal components for latent space vectors calculated via different autoencoder architectures.

0.3. This grading is then repeated for all features, for each autoencoder model. The overall score is then summed, as an indicator of the model performance. Table 5 summarizes the accuracy of the reconstructed spray PDFs for all features as a cumulative score. The columns and rows represent the tested autoencoder models and trajectory features, respectively.From Table 5, it seems that the data encoding and the following reconstruction process does cannot preserve the temporal evolution of the features along the combustor properly, despite the very low global reconstruction losses (Table 4). It is seen that some are relatively more successful, such as the LSTM (24) and GRU (24), with respect to reconstruction of global statistics. Nonetheless, the RNN-based autoencoder fails to capture the spatiotemporal dependencies in the multivariate time series (how individual features change in time and how they are correlated in time).

In an attempt to analyse the differences in alternative decoding strategies and why we observe such deviation in local feature probability distributions despite low reconstruction errors, Principal Component Analysis (PCA) is used on the latent representations ($\mathbf{z_i}$) generated by each autoencoder model. In principle, the usefulness of the encoding strategy comes from its ability to represent a sparse, high dimensional data with a much lower dimensional projection ($\mathbf{z_i}$ instead of $\mathbf{X_i}$), while maintaining as much information of the original feature space as possible. Herein, the question of interest is, whether RNN-based autoencoder learns the overall data representation, or it is capable enough to extract the casual dependencies within features, as well as with respect to time. In accordance, we would like to analyse the complexity of the latent data space as proxy, similar to the work of [25]. With linear PCA, it was seen that the first two principal components constitute >98% of the total variance in latent space, indicating that the compressed data in the latent space is projected into a simple manifold. It should be noted that PCA is performed on the latent space representation z (Fig. 2), hence includes the whole lifetime of a droplet trajectory implicitly. Interestingly, the latent representations of droplet trajectories of all autoencoder modules (except GRU (48) due to its relatively higher dimensionality) have similar distributions in the latent space (Fig. 4). The feature embedding in latent space did not change by increasing the latent space dimension, or the memory functions of the RNN cells. When these encoded features are constructed back with the decoder, the feature statistics show significant local differences with respect to original trajectories, as seen in Fig. 3 and Table 5 and do not match well with the original trajectory distributions.

In order to further investigate the discrepancy between the low training errors and the loss of accuracy in capturing local feature statistics, the changes in total droplet volumes within a differential combustor volume were evaluated as an additional criteria. This integrated value is considered as an indicator of how well the autoencoder captures global variables of interest such as volume change in the liquid phase at a given plane. For that purpose, differential combustor volumes are defined by placing two plane couples with an inter-distance of 1 mm at four axial locations. The resulting relative percentage volume changes are given in Fig. 5 for the relatively better performing autoencoders at the four planes PDFs are derived. It is seen that the models yields a very high accuracy in the plane-wise integrated quantities (maximum axis value is 1.25%). The reconstruction accuracy of GRU (24) is relatively better than the LSTM (24) architecture, for which the difference between the original and the reconstructed plane-averaged volume change is less than 1%. The LSTM (24) model tends to overestimate the volume change, which can be linked to an artificially larger expansion with increased temperature. In particular, when the temperature PDFs for GRU (24) and LSTM (24) are compared, it is seen that the peak temperature is shifted towards higher temperatures for GRU (24), while the opposite is true for LSTM (24) (see Supplementary Material 2 for all temperature PDFs). It is also worth nothing here that at the closest plane (x_1), net differential volume change was found to be slightly positive in the original data, which is due to the greater impact of volume expansion upon contact with the hot gas compared to the volumetric loss due to evaporation. Such an effect could not be preserved with either GRU or LSTM based autoencoders. The plane-wise volume change analysis empirically shows that the RNN-based autoencoders tend to minimize the differences in global representations during the training, rather than extracting and learning the spatiotemporal variations in the time series data, which is needed for an accurate low order model substitution for detailed Lagrangian trajectory models.

The above discussed observations reveal that alternative approaches are needed to fully capture the temporal and spatial interactions in the multivariate feature space. Convolutional neural networks, for instance, could help to build a hierarchical data representations and provide an efficient way to extract those local relationships via sets of kernels. One practical problem here is the variable size of the input data, due to the differences in the life times of the droplets depending on their initial size and trajectories. Even in the simulated non-reactive engine conditions, the life time of a droplets exhibits complex distributions (Fig. 7(a)). In order to alleviate the variable length issue, we considered that the distribution can be discretized into bins of finite intervals and an ensemble of convolutional autoencoders (cAEs) can be trained by padding the instances within the bins to the maximum size (Fig. 7(b,c)). In order to test the above hypothesis, we trained an cAE for the trajectory interval of 100–120 time frames (i.e., arrays of [100,10] to [120,10]). The samples with missing values were padded with zeros. In the preliminary training experiments, the original feature size of 120 × 10 was found be too skewed for the CNN layers to work. Therefore, the arrays were reorganized as into a shape of 40 × 30, while keeping the original feature space of droplet properties together (i.e, patches of 10). Several model structures with changing number of layers, filters and kernel sizes have been tested. Fig. 7(d) shows an example trajectory of a droplet with the best model hyperparameters. In the first encoding layer, 16 filters were used with a kernel size of (10 × 1) with maximum pooling (2,1), in an attempt to learn the inter-feature dependencies at a given time. The following encoding layer used 8 kernels (2 × 2). For decoding, four CNN layers were used and upsampling is done only in the third layer. The same error function is used to train the overall model. The comparison between the original and the reconstructed trajectory features shows that with a proper encoding policy, it is possible to learn the temporal evolution of the multidimensional feature space of the trajectories. Nonetheless, it should be noted that in these isolated CNN experiments, the learning space is much simpler than that of the RNN-based encoder which includes different physical regimes with a much broader droplet diameter and residence time distributions. Furthermore, the training of the whole ensemble cAEs would require a significant amount of hyperparameter testing to maximize the local reconstruction accuracy,

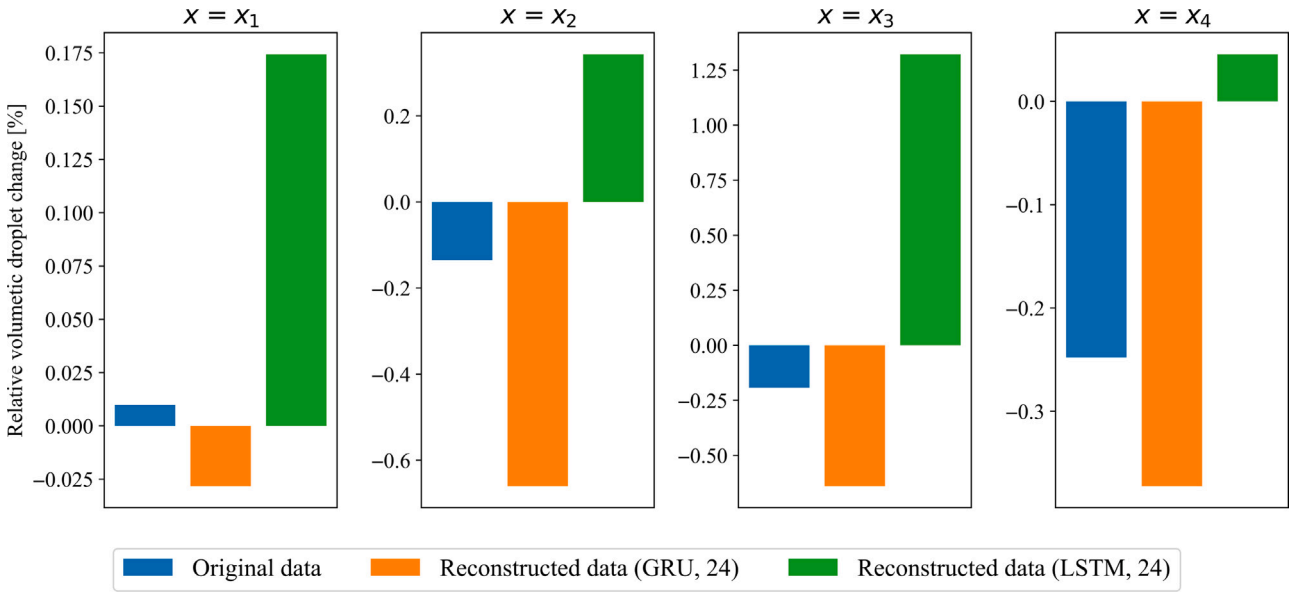


Fig. 5. Relative droplet volumetric changes reconstructed by GRU (24) and LSTM (24) autoencoders.

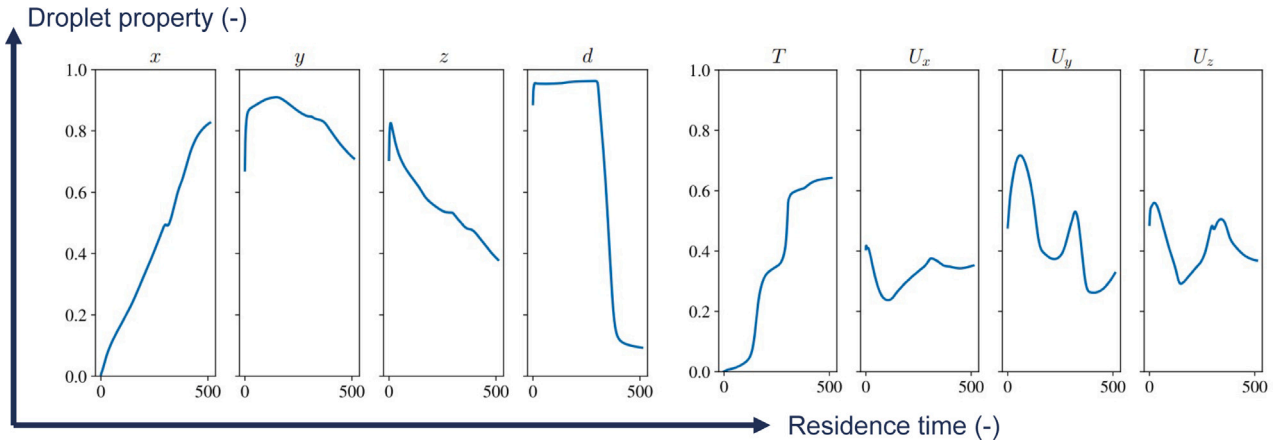


Fig. 6. Sampled droplet trajectory with the CGAN model (diameter-conditioned).

while ensuring that the overall approach does not overfit. The difficulty in the encoding of spatially and temporally correlated data illustrated here indicates that there is a need for further research in how to handle variable length, multivariate time series data, while preserving the local relationships and causality, as the high level objectives, such as creating generative, low order model representations (i.e., digital twins) of physical assets like fuel injectors in aeroengine combustors, require high precision and accuracy at both local and global levels. Herein, different binning and upsampling policies can be tested beyond simple padding to improve the robustness of the model in the future work.

Table 5
Global evaluation matrix for reconstructed spray statistics.

Feature	GRU (16)	GRU (24)	LSTM (24)	LSTM (32)
d	0	2	1	0
T	3	0	2	1
U_x	3	2	3	3
U_y	0	3	3	1
U_z	0	0	1	0
y	0	3	3	2
z	0	0	3	3
Final score ($\sum$)	6	10	16	9

3.2. Generating droplet sprays with CGAN

In the previous section, it has been shown that the RNN-based encoding of the original droplet trajectory data is capable of capturing integrated quantities such as droplet volume changes at given planes. Nonetheless, the local probability density distributions of the features were found to be difficult to learn irrespective of the autoencoder complexity. Even the best models failed to capture the whole of local statistics, particularly towards the combustor exit. In order to assess whether the encoding methodology could be improved by alternative approaches, an ensemble of convolutional autoencoders (cAE) is hypothesized, which is to be trained on different fractions of the variable length trajectory data. We tested the idea for one cAE and demonstrated that the spatiotemporal relationships can be learnt. Nonetheless, the model did require hyperparameter tuning even for a single cAE unit, around 40 of which is needed to create the ensemble model. In this section, we will focus on the second part of the problem, that is, exploring the capabilities of implicit generative models given the simpler latent representation of a complex input data. In particular, the performance of the custom CGAN model will be discussed as a generative fuel spray model, assuming that the latent space created by

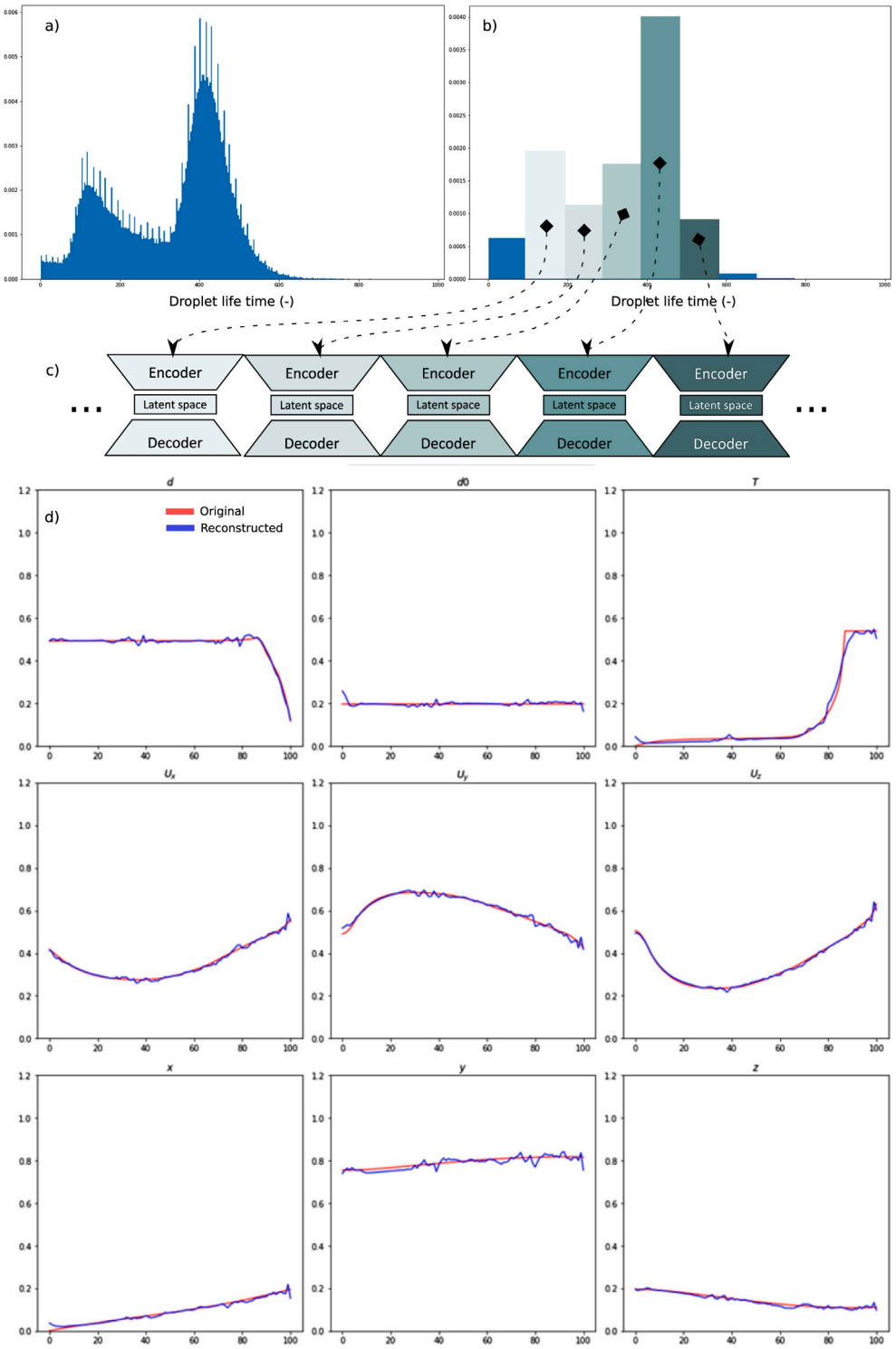


Fig. 7. Trajectory life time distributions (a), grouping idea with respect to time dimension of trajectories (b) and its integration with the ensemble convolutional autoencoders (cAE) with paddings (c). Sample reconstructed trajectories with a single cAE for trajectory interval [100,10]–[120,10] (d).

the encoding–decoding model is meaningful enough to perform such an assessment. This assumption is made to elaborate on the bottlenecks whole modelling framework for the future work. For that purpose, the trained autoencoder module GRU (24) is used in tandem with the GAN module, since it showed the best performance among all other models (see Section 3.1) and yields low error (<1%) in plane-wise integrated quantities of interest. Herein, the task of the autoencoder is to compress the variable length sequence into a fixed-length, simple representation. The CGAN model then creates similar hidden representations with the generator, which is trained by using the feedback from the Critic (Fig. 2). The synthetic (or fake) latent representations will be then decoded to create the artificial droplet trajectories. It should be reminded that the reason behind following this strategy is relax the difficulty of learning implicit coupled probability distributions by reducing the dimensionality of the problem, from around 5000 dimensions to 24.

Fig. 6 illustrates temporal evolution of a single droplet trajectory generated by the CGAN model conditioned on initial droplet diameter. In general, the generator seems to learn the trend of the features,

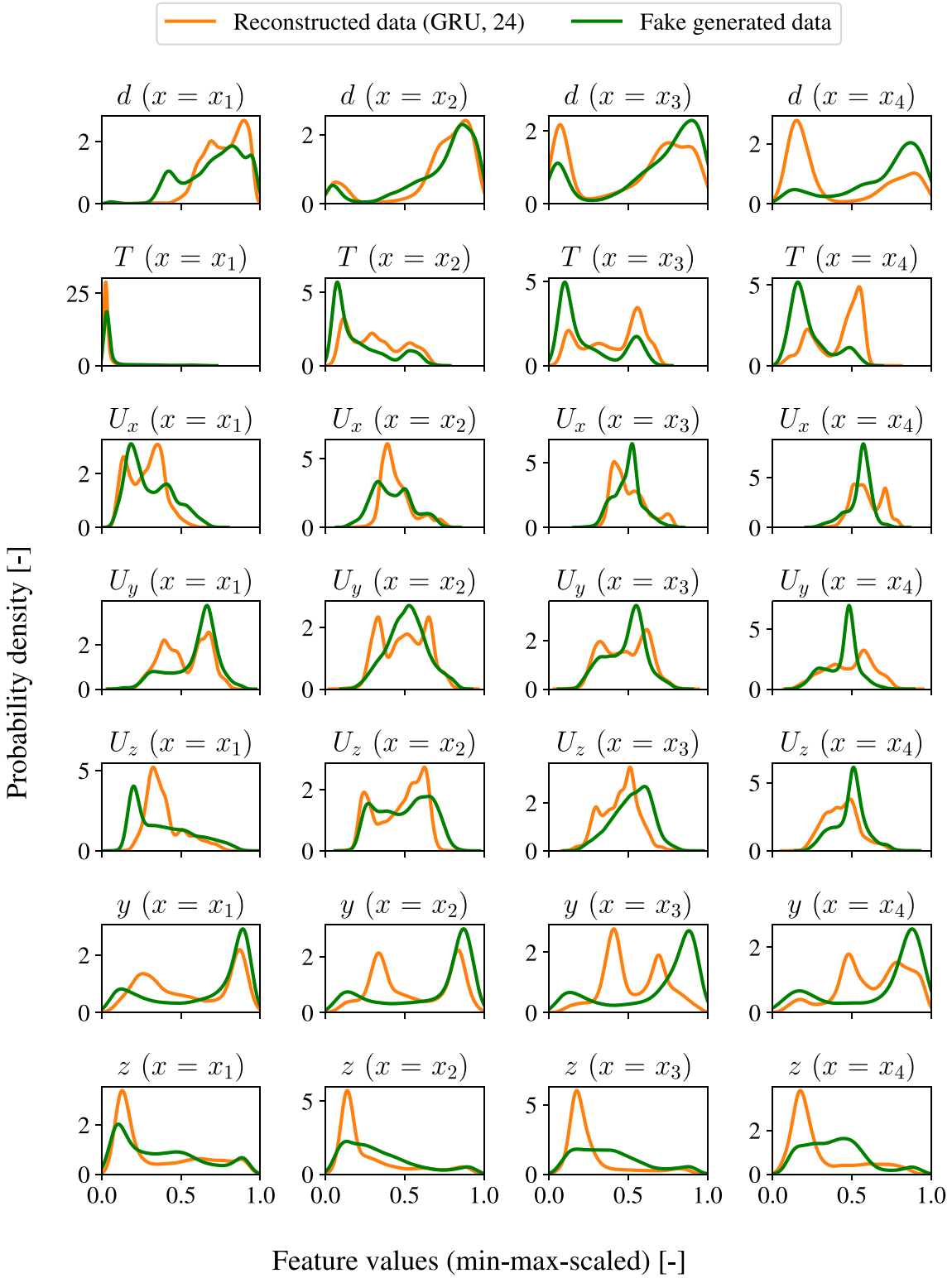


Fig. 8. Global spray statistics for the reconstructed (orange) and generated (green) trajectories with the CGAN model (diameter-conditioned).

although the curves may look noisier for some trajectories. The x-position is usually predicted realistically, since this is almost a linear function with respect to sequence step. In some cases, the correlation between droplet size and temperature is represented quite well. In other cases, however, the predicted temperatures show unnatural oscillations (See Supplementary Material 2 for more examples). For the lateral and axial velocity components, synthetic trajectories were found to mimic the observed fluctuations in the flow field.

The global spray statistics of the GAN module conditioned on the initial diameter are shown in Fig. 8. PDFs are constructed by using 20,000 real and fake trajectories. It should be noted that the PDFs of the generated spray (fake data) are compared with the reconstructed data

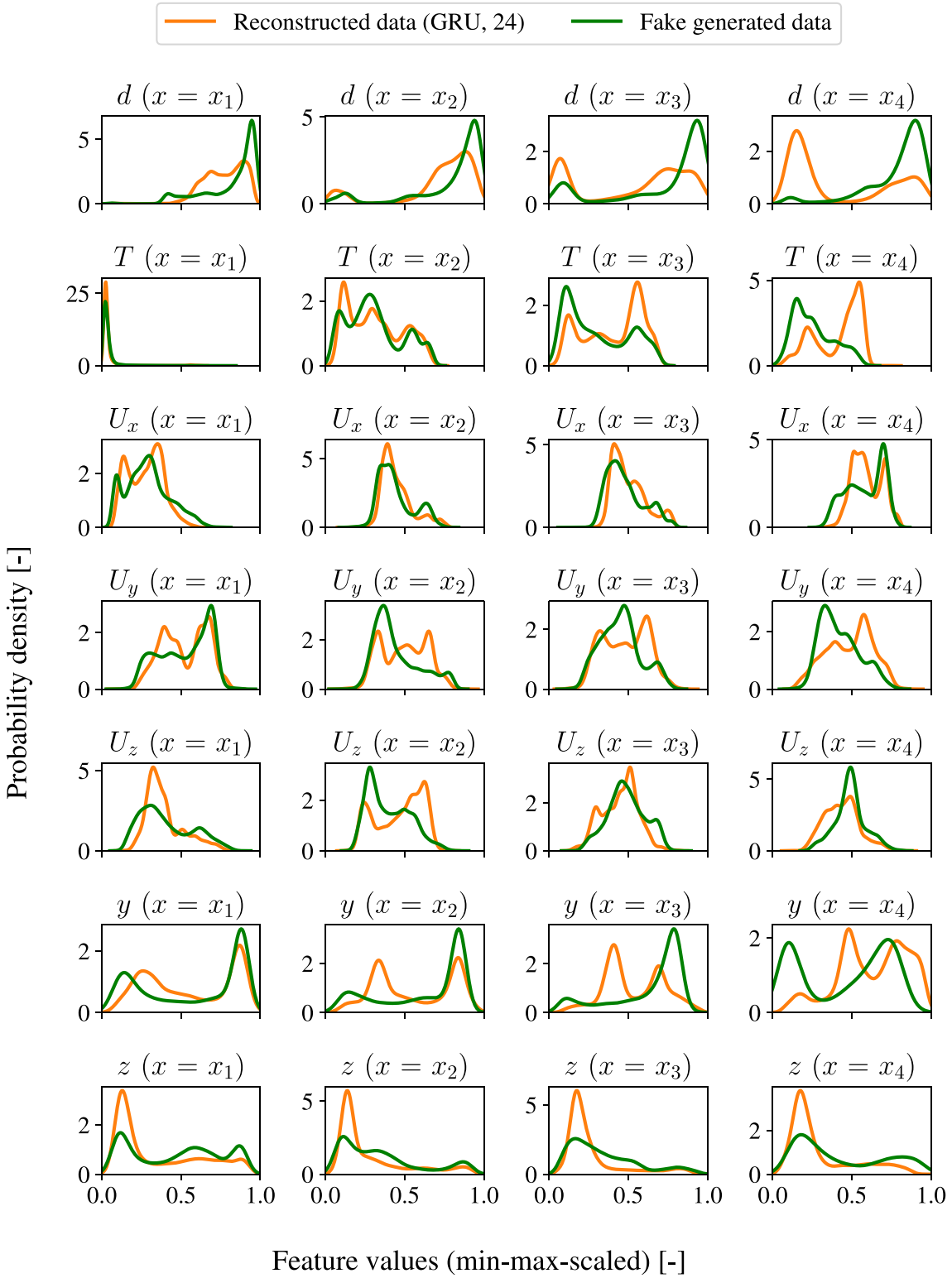


Fig. 9. Global spray statistics for the reconstructed (orange) and generated (green) trajectories with the CGAN model (position-conditioned).

of real spray trajectories in order to quantify the errors originating from the GAN module, as the AE module already has errors in local statistics. Comparisons with the real spray data can be found in Supplementary Material 2. It is seen that at some planes, the PDFs of the reconstructed trajectory features were captured quite well, while on others the statistics were learnt not properly. The generative model yields a lower number larger droplets and a higher number of fine droplets at the first sampling plane. The best match between both curves is found for the second sampling plane. The overlap decreases with increasing x-coordinate (axial direction), where the generative model tends to underestimate the number of fine droplets. The temperature distribution is captured only reasonably well up to the second plane. Going

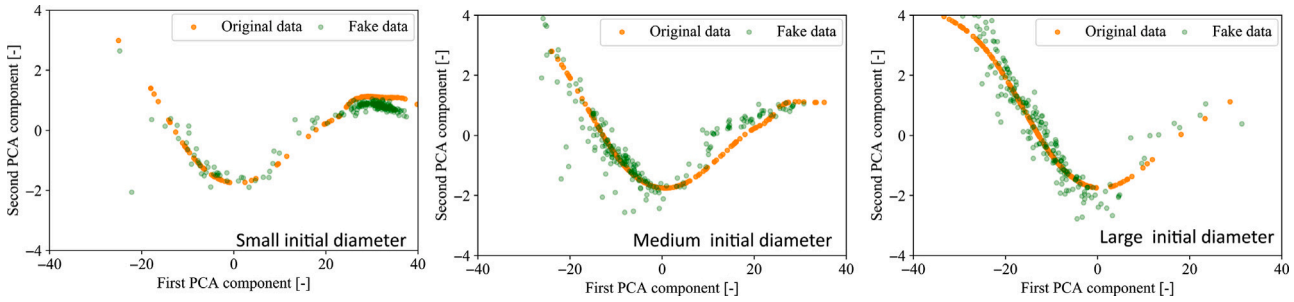


Fig. 10. Comparison between the 2D latent representations of original samples and fake samples for different initial diameters. Dimensionality reduction was done via PCA.

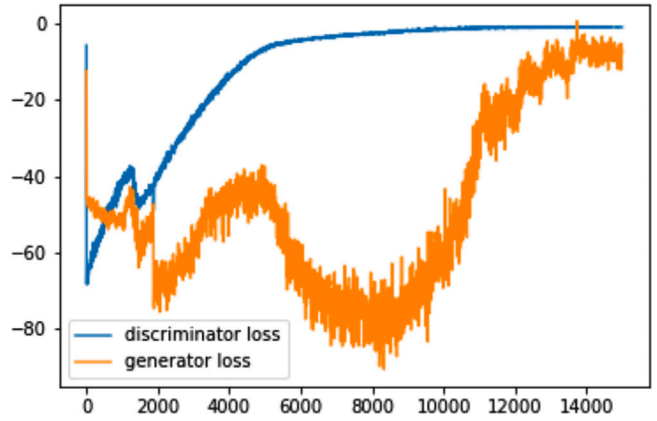


Fig. 11. Evolution of training loss functions of the GAN module (diameter conditioned).

downstream, the lower temperature peak is over-estimated, whereas the higher temperature peak is under-estimated, compared to the reconstructed trajectories. The performance regarding the three velocity components varies at different locations. The distribution of the initial positions is captured quite well, however at larger x, hence at later times, the overlap between the PDFs was found to be reduced. In general, it was found that the discrepancy between the PDFs of the generated trajectory and the reconstructed data decreases towards the exit of the combustor.

When the model is conditioned on the initial position (Fig. 9), there is not much improvement compared to the diameter-conditioned GAN (Fig. 8). The position-conditioning did not affected the model performance with respect to position, temperature or velocity statistics. Interestingly, the PDFs for the droplet diameter are found to be worse than the ones generated by the diameter-conditioned GAN in all four planes. In other words, conditioning on diameter was a relatively more successful strategy compared to the conditioning on the initial position. It may be due to the fact that the fuel spray was symmetric with respect to the azimuthal angle, therefore conditioning the position on yz plane did not bring additional information regarding to the trajectory dynamics, compared to the diameter information.

Similar to the autoencoder model, we looked into the lower-dimensional representations of the generated trajectories via PCA analysis and compared it with the real samples. Fig. 10 illustrates how the 2D linear projections are distributed for different droplet size ranges (i.e., first two PCs that carries more than 98% of the total variance of the original latent feature space). Herein, three classes of droplet sizes are selected for visualization based on their initial diameters: small (<116 μm), medium (140 μm < d_0 < 152 μm) and large (188 μm < d_0 < 200 μm) droplets. It is seen that the diameter-conditioned GAN can generate samples that are close to the original ones in the latent space with respect to the requested diameter labels. Due to its probabilistic nature, the fake samples are spread around the CFD-generated data (orange dots in Fig. 10), yet with acceptable variance. An important observation here is the fact that the GAN model does not suffer from the mode collapse problem. The generator could provide diverse examples following the original distribution and do not cluster into a small region of the latent space (i.e., generate samples that are very similar/identical which are acceptable for the Critic). Comparisons in the latent space show that conditioning a GAN with respect to initial droplet properties can control the creativity of the model successfully, even for a multivariate time series problem. It should be noted that there are alternative generative learning methods beyond the GANs that can be conditioned, such as Gaussian Mixture Models [33] or Hidden Markov Models [34,35]. The capabilities of such alternative approaches could be further explored in the follow-up studies. It should be also noted that it is very difficult to interpret the outcome of the generative models such GAN directly. At best, we can measure the similarities between the generated and the original data, such as the Wasserstein distance which converges towards to the end of the training 11. Even in that case, however, the "distance" itself is an abstract quantity. Therefore, particularly for scientific or engineering oriented applications, the evaluation of the generative models should be coupled with the physics of the problem for a better interpretation. In the current work, we used alternative metrics such as the analysis of the generated trajectories with respect to their statistics via PDFs and integrated quantities such as plane-averaged volume changes. In the future work, it may be worth to investigate the added value of modifying error definitions to train the generative module to inject more physics from the known statistics of the physical problem as a regularization term.

It should be also noted that direct learning of the underlying mutimodal probability distributions in the high dimensional feature space (5000 dimensions even if we solve the variable sequence length problem) is an extremely challenging task, if we are particularly interested in preserving the local inter-dependencies of the transient droplet trajectories. This is why simplification of the data representation was considered a crucial prerequisite before training a generative model. Our analysis showed that this pre-processing step is still very difficult, even if we decouple it from the generative learning process. RNN-based models with complex memory functions were at best capable of keeping track of integrated variables of interest such as the droplet volume change in a differential combustor volume. When we further simplify the preprocessing of data representation with the ensemble cAE idea 7, the encoding–decoding performance was much found to be much better. These comparisons indicate that the bottleneck of overall the learning process for such high dimensional data in the autoencoding strategy and there is a need for further research to establish robust methodologies that can preserve the local, multimodal dependencies.

4. Conclusions

In this work, a custom conditional GAN model was tested as a statistical spray model to generate artificial droplet trajectories for aero engine combustors. It includes two key features. First, it is capable

of handling sequences with variable lengths. Second, the model can be conditioned on different initial states by either defining an initial droplet diameter or initial droplet position. The data set used for the training was extracted from the transient Eulerian–Lagrangian CFD simulations. Due to the high dimensionality of the input feature space (3000–7000), generative learning task is conducted in two stages: (i) encoding of the variable size droplet trajectory data into fixed length, low dimensional representations via autoencoders and (ii) learning the multimodal dependencies of transient fuel droplet features (initial diameter, diameter, temperature, position, velocity, residence time) in the low dimensional representations via conditional GANs.

During the training phase of the autoencoders, the reconstruction errors were found to be very small for the whole training data, yet the global statistics of the fuel droplets along the combustor were not conserved and disagreed noticeably from the original distributions. The reconstruction error was found to be typically highest for the late stages of the droplet trajectories, indicating that a generic recurrent autoencoder architecture may not be sufficient to capture the sequential dependencies of the multivariate auto-regression task, even with GRU/LSTM cells. Hyperparameter tuning also failed to improve the reconstructed local probability density functions of the fuel sprays. The best model yielded acceptable match between the original and reconstructed feature probability density distributions in only 16/28 of the sampled local distributions. Analysis of the integrated (differential volumes perpendicular to main flow direction) quantities such as total change in the droplet volumes, on the other hand, showed that the trained RNN-based autoencoders are good at capturing the global behaviour. CNN-based autoencoders seems to be better at capturing spatiotemporal dependencies in droplet trajectories, yet variable array sizes originating from the physics of the evaporating spray with a droplet size distribution makes direct utilization of a single Convolutional autoencoder impractical. Herein, the potential of an ensemble approach with a robust upsampling/downsampling policies should be further investigated. Training of the GAN model in the latent space was much more straightforward and successful. The model was capable to mimic some high level features such as the initial droplet diameter of the trajectory and mode collapse was not observed. The diameter based conditioning was also found to improve the spray statistics generated by the GAN model further.

Comparisons showed that how the high dimensional, variable length feature space of the multivariate time series data is projected into a low dimensional, fixed length representation is the key step for creating accurate low order models of complex physical phenomena such as fuel injection in aero engines and there is a need for further research to establish robust methodologies that can preserve local, multimodal dependencies.

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.

Data availability

Droplet trajectory data and the custom GAN code can be found in Supplementary Material 1.

Acknowledgements

This work was performed on the bwUniCluster supercomputer. The authors acknowledge support by the state of Baden-Württemberg, Germany, through bwHPC.

We acknowledge support by the KIT Publication Fund of the Karlsruhe Institute of Technology, Germany.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.egyai.2022.100216.

References

[1] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. Generative adversarial nets. In: Ghahramani Z, Welling M, Cortes C, Lawrence N, Weinberger K, editors. Advances in neural information processing systems, vol. 27. Curran Associates, Inc.; 2014, URL https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

[2] Liu MY, Huang X, Yu J, Wang TC, Mallya A. Generative Adversarial Networks for Image and Video Synthesis: Algorithms and Applications. Proc IEEE 2021;109(5):839–62. http://dx.doi.org/10.1109/JPROC.2021.3049196, arXiv:2008.02793.

[3] Tzaban R, Mokady R, Gal R, Bermano AH, Cohen-Or D. Stitch it in Time: GAN-Based Facial Editing of Real Videos. 2022, arXiv:2201.08361.

[4] Palsson S, Agustsson E, Timofte R, Van Gool L. Generative adversarial style transfer networks for face aging. In: 2018 IEEE/CVF Conference on computer vision and pattern recognition workshops. CVPRW, 2018, p. 2165–21658. http://dx.doi.org/10.1109/CVPRW.2018.00282.

[5] Zhang H, Xu T, Li H, Zhang S, Huang X, Wang X, Metaxas DN. StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks. 2016, CoRR arXiv:1612.03242.

[6] Mogren O. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. 2016, http://dx.doi.org/10.48550/ARXIV.1611.09904, URL https://arxiv.org/abs/1611.09904.

[7] Donahue C, McAuley J, Puckette M. Adversarial audio synthesis. 2018, http://dx.doi.org/10.48550/ARXIV.1802.04208, URL https://arxiv.org/abs/1802.04208.

[8] Pascual S, Bonafonte A, Serrà J. SEGAN: speech enhancement generative adversarial network. 2017, CoRR abs/1703.09452 arXiv:1703.09452.

[9] Razavi-Far R, Ruiz-Garcia A, Palade V, Schmidhuber J. Generative adversarial learning: architectures and applications. Springer; 2022.

[10] Zhou K, Diehl E, Tang J. Deep convolutional generative adversarial network with semi-supervised learning enabled physics elucidation for extended gear fault diagnosis under data limitations. Mech Syst Signal Process 2023;185:109772. http://dx.doi.org/10.1016/j.ymssp.2022.109772, URL https://www.sciencedirect.com/science/article/pii/S0888327022008408.

[11] Ishikawa A. Heterogeneous catalyst design by generative adversarial network and first-principles based microkinetics. Sci Rep 2022;12(1):1–9. http://dx.doi.org/10.1038/s41598-022-15586-9.

[12] Wang Y, Wang Z, Wang W, Tao G, Shen W, Cui J. Two-dimensional prediction of the superposition film cooling with trench based on conditional generative adversarial network. Int J Therm Sci 2023;184:107976. http://dx.doi.org/10.1016/j.ijthermalsci.2022.107976, URL https://www.sciencedirect.com/science/article/pii/S129007292200504X.

[13] Bode M, Gauding M, Lian Z, Denker D, Davidovic M, Kleinheinz K, Jitsev J, Pitsch H. 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, URL https://www.sciencedirect.com/science/article/pii/S1540748920300481.

[14] Gauding M, Bode M. Using physics-informed enhanced super-resolution generative adversarial networks to reconstruct mixture fraction statistics of turbulent jet flows. In: Jagode H, Anzt H, Ltaief H, Luszczek P, editors. High performance computing. Cham: Springer International Publishing; 2021, p. 138–53.

[15] Jabbar A, Li X, Omar B. A survey on generative adversarial networks: Variants, applications, and training. ACM Comput Surv 2021;54(8). http://dx.doi.org/10.1145/3463475.

[16] Bond-Taylor S, Leach A, Long Y, Willcocks CG. Deep generative modelling: A comparative review of VAEs, GANs, normalizing flows, energy-based and autoregressive models. IEEE Trans Pattern Anal Mach Intell 2022;44(11):7327–47. http://dx.doi.org/10.1109/TPAMI.2021.3116668.

[17] Yoon J, Jarrett D, van der Schaar M. Time-series generative adversarial networks. Adv Neural Inf Process Syst 2019;32(NeurIPS):1–11.

[18] Pei H, Ren K, Yang Y, Liu C, Qin T, Li D. Towards Generating Real-World Time Series Data. 2021, arXiv:2111.08386v1.

[19] Esteban C, Hyland SL, Rätsch G. Real-valued (medical) time series generation with recurrent conditional GANs. 2017, http://dx.doi.org/10.48550/ARXIV.1706.02633, URL https://arxiv.org/abs/1706.02633.

[20] Naruse M, Matsubara T, Chauvet N, Kanno K, Yang T, Uchida A. Generative adversarial network based on chaotic time series. Sci Rep 2019;9(1):12963. http://dx.doi.org/10.1038/s41598-019-49397-2.

[21] Zhang C, Kuppannagari SR, Kannan R, Prasanna VK. Generative Adversarial Network for Synthetic Time Series Data Generation in Smart Grids. In: 2018 IEEE International conference on communications, control, and computing technologies for smart grids, SmartGridComm 2018. IEEE; 2018, p. 1–6. http://dx.doi.org/10.1109/SmartGridComm.2018.8587464.

[22] Wiese M, Knobloch R, Korn R, Kretschmer P. Quant GANs: deep generation of financial time series. Quant Finance 2020;20(9):1419–40. http://dx.doi.org/10.1080/14697688.2020.1730426, arXiv:1907.06673.

[23] Nord S. Multivariate Time Series Data Generation using Generative Adversarial Networks : Generating Realistic Sensor Time Series Data of Vehicles with an Abnormal Behaviour using TimeGAN. 2021, URL http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-302644.

[24] Asre S, Anwar A. Synthetic Energy Data Generation Using Time Variant Generative Adversarial Network. Electronics (Switzerland) 2022;11(3). http://dx.doi.org/10.3390/electronics11030355.

[25] Baasch G, Rousseau G, Evins R. A Conditional Generative adversarial Network for energy use in multiple buildings using scarce data. Energy AI 2021;5:100087. http://dx.doi.org/10.1016/j.egyai.2021.100087.

[26] Yilmaz B, Korn R. Synthetic demand data generation for individual electricity consumers: Generative Adversarial Networks (GANs). Energy AI 2022;9(February):100161. http://dx.doi.org/10.1016/j.egyai.2022.100161.

[27] Brophy E, Wang Z, She Q, Ward T. Generative adversarial networks in time series: A survey and taxonomy. 2021, arXiv:2107.11098.

[28] Festag S, Denzler J, Spreckelsen C. Generative adversarial networks for biomedical time series forecasting and imputation. J Biomed Inform 2022;129:104058. http://dx.doi.org/10.1016/j.jbi.2022.104058, URL https://www.sciencedirect.com/science/article/pii/S1532046422000740.

[29] Mirza M, Osindero S. Conditional generative adversarial nets. 2014, arXiv preprint arXiv:1411.1784.

[30] Chaussonnet G, Dauch T, Keller M, Okraschevski M, Ates C, Schwitzke C, Koch R, Bauer HJ. Progress in the Smoothed Particle Hydrodynamics Method to Simulate and Post-process Numerical Simulations of Annular Airblast Atomizers. Flow Turbul Combust 2020;105(4):1119–47. http://dx.doi.org/10.1007/s10494-020-00174-6.

[31] Arjovsky M, Chintala S, Bottou L. Wasserstein generative adversarial networks. In: International conference on machine learning. PMLR; 2017, p. 214–23.

[32] Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville A. Improved training of wasserstein GANs. In: Proceedings of the 31st International conference on neural information processing systems. Red Hook, NY, USA: Curran Associates Inc.; 2017, p. 5769–79.

[33] Manduchi L, Chin-Cheong K, Michel H, Wellmann S, Vogt JE. Deep conditional Gaussian mixture model for constrained clustering. 2021, http://dx.doi.org/10.48550/ARXIV.2106.06385, URL https://arxiv.org/abs/2106.06385.

[34] Li Y, Song L, Zhang C. Sparse conditional hidden Markov model for weakly supervised named entity recognition. In: Proceedings of the 28th ACM SIGKDD Conference on knowledge discovery and data mining. ACM; 2022, http://dx.doi.org/10.1145/3534678.3539247.

[35] Panousis KP, Chatzis S, Theodoridis S. Variational conditional dependence hidden Markov models for skeleton-based action recognition. 2020, http://dx.doi.org/10.48550/ARXIV.2002.05809, URL https://arxiv.org/abs/2002.05809.