Reconstruction of turbulent data with deep generative models for semantic inpainting from TURB-Rot database

M. Buzzicotti[1], F. Bonaccorso[1,2], P. Clark Di Leoni[3], L. Biferale[1]

[1] *Dept. Physics and INFN, University of Rome "Tor Vergata", Italy.*
[2] *Center for Life Nano Science@La Sapienza, Istituto Italiano di Tecnologia and INFN, University of Rome "Tor Vergata"*
[3] *Department of Mechanical Engineering, Johns Hopkins University, Baltimore, Maryland 21218, USA.*

(Dated: June 17, 2020)

We study the applicability of tools developed by the computer vision community for *feature learning* and *semantic image inpainting* to perform data reconstruction of fluid turbulence configurations. The aim is twofold. First, we explore on a quantitative basis, the capability of Convolutional Neural Networks embedded in a Deep Generative Adversarial Model (Deep-GAN) to generate missing data in turbulence, a paradigmatic high dimensional chaotic system. In particular, we investigate their use in reconstructing two-dimensional *damaged* snapshots extracted from a large database of numerical configurations of 3d turbulence in the presence of rotation, a case with multi-scale random features where both large-scale organised structures and small-scale highly intermittent and non-Gaussian fluctuations are present. Second, following a reverse engineering approach, we aim to rank the input flow properties (features) in terms of their qualitative and quantitative importance to obtain a better set of reconstructed fields. We present two approaches both based on Context Encoders. The first one infers the missing data via a minimization of the L_2 pixel-wise reconstruction loss, plus a small adversarial penalisation. The second, searches for the closest encoding of the corrupted flow configuration from a previously trained generator. Finally, we present a comparison with a different data assimilation tool, based on Nudging, an *equation-informed* unbiased protocol, well known in the numerical weather prediction community. The TURB-Rot database, `http://smart-turb.roma2.infn.it`, of roughly 300K 2d turbulent images is released and details on how to download it are given.

PACS numbers:

I. INTRODUCTION

Data assimilation (DA) is the art of interpolating, de-noising, super-resolving or filling missing information from a single realization, a time or oriented series, or a statistical sample of some random or deterministic dataset [1, 2]. It is important in many areas, such as classification and inpainting in computer vision (CV) [3–5], developing natural language processing [6, 7], inverse problems such as source localization in fluid dynamics [8, 9], or to prepare more and more refined initial conditions in numerical weather predictions [10–13], just to cite a few examples from apparently completely different fields. The fact is that our visual and/or data samples are very diverse, but also highly structured. Sometimes, the naked eye is better than sophisticated algorithms in interpolating, classifying, and guessing the meaning and contents of blurry and gappy images, if trained enough. It can be a picture taken from a dataset of digitalized images [14–19] or a visualization of real physical fields contents, e.g. the flow configuration at a given instant of our atmosphere. The problems are still the same, we often need to quickly understand the context, to control the quantitative details and contents, to classify it, to react or to predict the future evolution. For physical applications the challenges, as well as their importance, are clear. The problem can roughly be split into two sub-domains. The case where the missing or blurred information is scattered everywhere but locally affecting only a small cluster of pixels, or where big chunks of data are affected and one cannot use local interpolation techniques. Consider the three damaged images of a velocity amplitude on a 2d slice taken from a 3d direct numerical simulations of a rotating turbulent flow shown in Fig. 1. From this example, it is clear that while filling case (a) when the *noise* is affecting only small-scale properties might seem to be relatively simple (and indeed it is, if the gap is smaller than, or comparable with, the smallest active chaotic scale in the flow [20]), for cases (b) and (c) where the gap is bigger and bigger, it is not at all obvious to guess the ground truth (panel d). For big holes, we need first to learn the *context*, i.e. the multi-scale coherent structures and their statistical significance, and then inferring what the missing information should be, on the basis of our *semantic* understanding and with little possibility to get clues from nearby data.

In this paper, we aim at investigating these kind of questions using TURB-Rot, a huge dataset made of hundred thousands of 2d configurations extracted from a direct numerical simulation of 3d turbulent flow in the presence of rotation, see appendix C and [21]. Rotating turbulence is a paradigmatic case with very rich physics, where energy injected from the external forcing produces both large-scale cyclonic and anti-cyclonic structures as well as small-scale intermittent homogeneous and isotropic highly non-Gaussian fluctuations, resulting on chaotic flow excitation

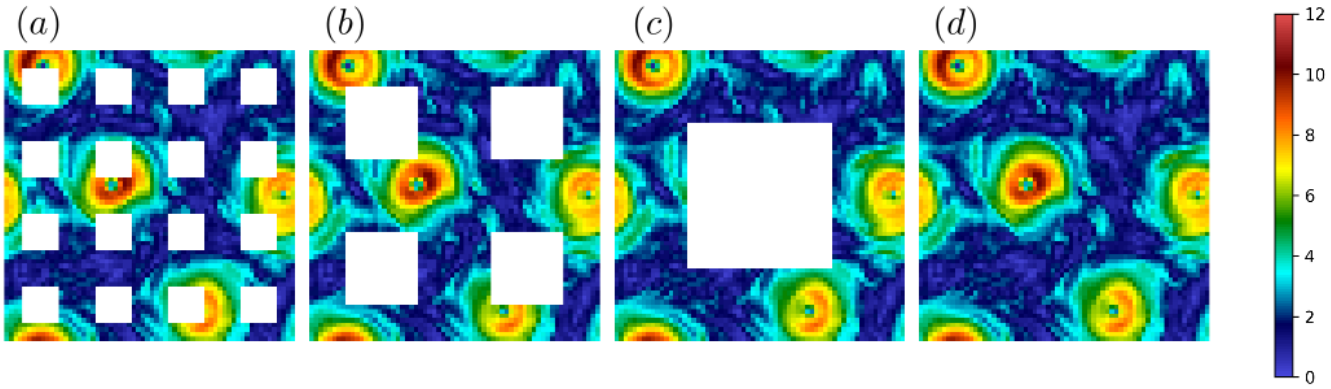

FIG. 1: Example of a typical inpainting exercise on a 2d slice of a 3d turbulent rotating flows (here the color code is proportional to the local velocity intensity). We imagine to have either many small-scale damages (panel a) or larger and larger gaps (panels b and c) leading to the need to guess global semantic information of the image, e.g. the internal structure of a whole vortex and the shear flow in the intermediate regions. The final task is to reconstruct the ground truth (panel d).

spanning many decades in scales and frequencies (see, e.g. the Fourier spectrum in Fig. 3 and [22] for a recent review). Rotating (and stratified) turbulence is also a key set-up in many geophysical applications, both for atmosphere and ocean dynamics. To achieve the task of reconstructing the missing information in these snapshots we will make use of two different Machine Learning (ML) algorithms both based on *Context Encoders* (CE). The first one was proposed in [3] and consists of a Generator (G) made of an encoder that stores the context of the input data in a compact (low-dimensional) latent representation by doing a down-sampling of the original corrupted image, and a decoder that builds on this representation to generate the missing information. The second one is given by a variation of the same idea [4], where first we train a Deep-GAN conditioned on a noise vector, able to produce an entire realistic sample for the flow fields in the whole 2d domain, then we freeze the network and perform a second optimization with *back propagation* on the input noise vector, meant to find the optimal entry data which generate the known context. We will also compare these methods with a non-ML, equation informed, DA algorithm based on Nudging [23–25], a method that uses the Navier-Stokes equations (NSE) with a Newton relaxation feedback term, imposing a linear reward/penalisation depending if the evolved fields are close/far from the data to be assimilated.

Before moving to the results, let us make a few statements. The literature using equation-informed or equation-free (with or without some physics constraint) tools to model, reconstruct and assimilate fluid flow is huge, one can find some past and recent reviews and books here [1, 2, 10, 12, 26–29]. It is also important to point out a few recent developments in the field of turbulence reconstruction [25, 30–33] and in the efforts to merge ML and DA [34, 35]. The goal of this paper is not to develop new ML algorithms for DA neither to spend millions of computing hours to find the optimal combination of network hyper-parameters to improve the final reconstruction. These are important issues that would deserve a different work. Here we are motivated by exploring how much some of the most popular and recent CV algorithms can be used for a quantitative assessment of DA for fluid configuration. We do this by implementing a series of systematic quantitative benchmarks based on (i) spectral properties; (ii) point-wise L_2 errors and (iii) probability distribution functions for large-scale quantities (velocity components) and small-scale ones (vorticity). Similarly, our goal is not meant to explore other subtle and important points connected to the need of imposing soft or hard physical constraints coming from the original properties of the chaotic ODEs or PDEs [36–42], we limit in the first part to a CV approach, as if we do not know the equations of motion (as it happens in many real applications also in basic science). Here, we specified to 2d snapshots from rotating turbulent flows but there is nothing that prevents us (except eventually an issue connected with the capacity of the Deep-GAN structures) to apply the same techniques to other flow configurations and/or fully 3d or to (3+1)d data structures [43–46]. Work in this direction is underway and it will be reported elsewhere. In a final part of the paper, we also address one example from the other -opposite- point of attack: supposing we know and we use the equations of motion and/or the embedded time-evolution. There are many DA approaches which have been developed in the past for fluid flows, e.g. Kalman filters [47, 48], Kriging [49, 50], proper orthogonal decomposition [26, 27, 51, 52] to cite just a few. As already mentioned above, to compare with ML we chose an algorithm based on Nudging. Finally, we also made available the whole database used for the training and validation of the two CE1 and CE2, hoping to trigger the interest of the community to improve on our results.
The main results of our work can be summarized as follows.

- Both CE1 and CE2 are capable to refill well the missing regions in the corrupted flow configurations, with

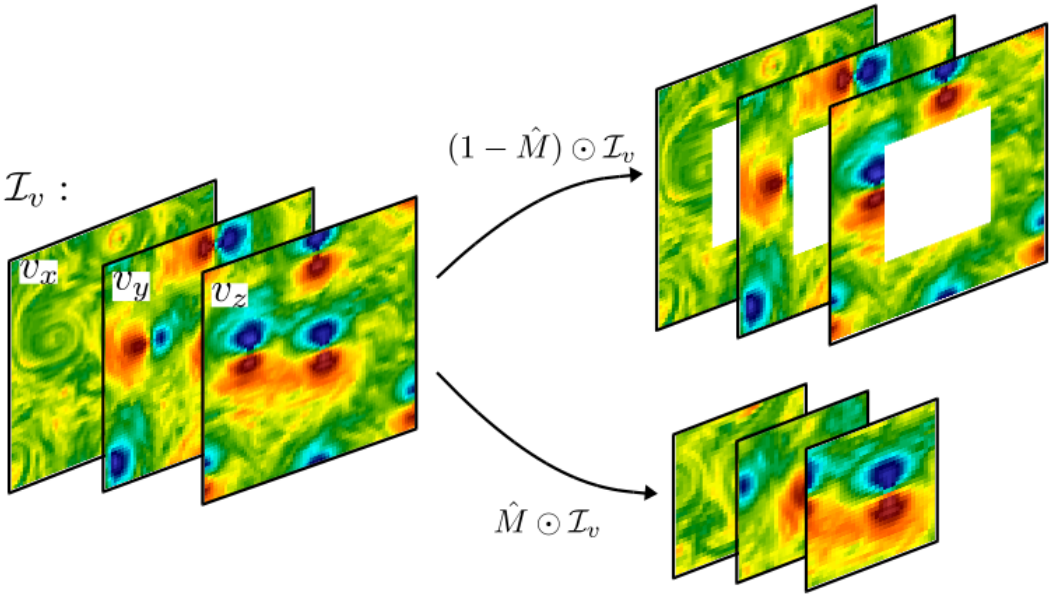

FIG. 2: Sketch of the typical image damaging protocol we implement, separating the set of the velocity components $\mathcal{I}_v = (v_1, v_2, v_3)$ in the 2d plane (x_1, x_2) in a corrupted image, $(1-\hat{M})\odot\mathcal{I}_v$, and in its removed hole, $\hat{M}\odot\mathcal{I}_v$, where $\hat{M}$ is a mask made of 0s (in the external frame) and 1s (in the central hole).

average local L_2 errors that can be as small as 15% for CE1 in the presence of large gaps and $3-4\%$ for smaller damaged spots. For CE2 we get roughly the double.

- Multi-scale spectral properties and probability distribution functions for both velocity and vorticity are extremely well reproduced, including extreme events.
- Concerning features ranking, CE works better with large-scale velocity in inputs than with small-scale vorticity field, probably due to the large intermittent and non-Gaussian properties of the latter.
- Nudging gives comparable results and the trade-off between the two approaches must be considered depending on the applications and on the quality of the information and numerical tools available (e.g. if we know the equations and we possess the numerical set-up to evolve in time a fully developed 3d turbulent flow, something that typically requires the evolution of billions of degrees of freedom [53]).

The paper is organized as follows. In Sec. II we give a brief introduction of the physics of turbulence under rotation in Sec. III we describe the detail of the CE1 network and we show the results of this approach, in Sec. IV we describe the CE2 network and we discuss the results, in Sec. V we introduce and discuss the details of Nudging technique. In Sec. VI we conclude and summarize all main results. In appendix A-B we show the detailed structure of the two networks used. In appendix C we discuss the original data-base used for the CE1 and CE2 and the protocol to download all data.

II. ROTATING TURBULENCE

Let us first make a small digression introducing the most important features of turbulence under rotation. In a rotating frame of reference, the Navier-Stokes equations take the form:

$$\frac{\partial \boldsymbol{v}}{\partial t} + \boldsymbol{v}\cdot\boldsymbol{\nabla}\boldsymbol{v} + 2\Omega\hat{x}_3\times\boldsymbol{v} = -\boldsymbol{\nabla}p + \nu\nabla^2\boldsymbol{v} + \boldsymbol{f} \tag{1}$$

where ν is the kinematic viscosity, $\boldsymbol{f}$ is an external forcing mechanism, $2\Omega\hat{x}_3\times\boldsymbol{v}$ is the Coriolis force with Ω being the rotation frequency and with the rotation axis here chosen to be parallel to $\hat{x}_3$. The incompressibilty condition $\boldsymbol{\nabla}\cdot\boldsymbol{v} = 0$ closes the equations. The most important feature of rotating flows is the tendency to accumulate energy

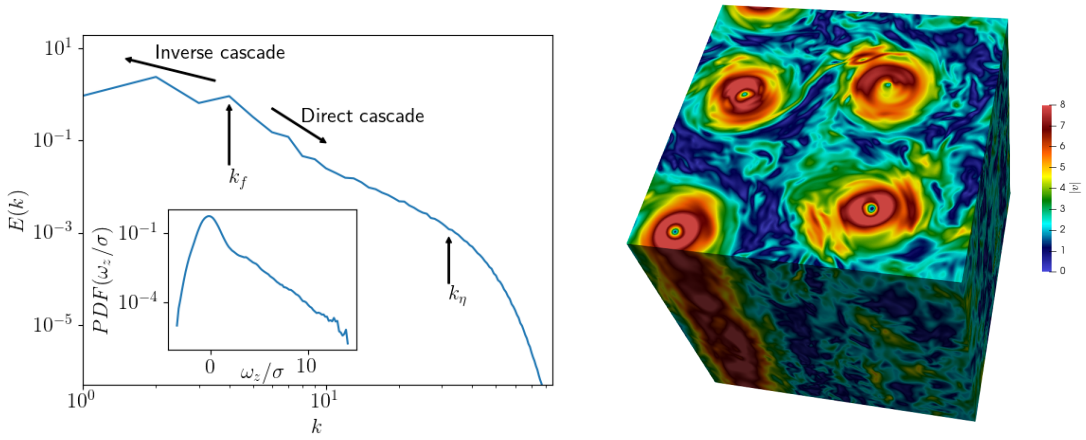

FIG. 3: Right: 3d rendering of the velocity amplitude $|\boldsymbol{v}|$, in a rotating turbulent flow. Left: 3d averaged energy spectrum, $E(k) = \sum_{\mathbf{k}=k}^{k+1} \langle |\hat{\boldsymbol{v}}(\mathbf{k})|^2 \rangle$, where with $\hat{\boldsymbol{v}}(\mathbf{k})$ we denote the Fourier coefficient at wavenumber $\boldsymbol{k}$. The forcing and the Kolmogorov wavenumbers, k_f, k_η are indicated with arrows, notice the presence of power-law energy fluctuations on almost two decades and for wavenumbers smaller and larger than k_f, the indication of the split-energy cascade scenario. Inset: standardized probability distribution function of the vorticity in the direction of the rotation axis, ω_3. Notice the presence of strong non-Gaussian, intermittent and skewed fluctuations up to $O(10)$ (and higher) standard deviations, σ.

to larger and larger scale, leading to the formation of columnar quasi-2d vortical structures orientated along the rotation axis (see Fig. 3 and [54–61]). The phenomenon is accompanied by a simultaneous transfer of energy to smaller and smaller scales too, producing strong non-Gaussian tails in the vorticity probability distribution function (Fig. 3, inset of left panel). We are in the presence of a split-cascade scenario, where the energy injected by the external stirring mechanism is redistributed to both small and large wavenumbers [22].

The data used in the CV experiments comes from solving Eq. (1) on triple periodic cubic box of size $2\pi \times 2\pi \times 2\pi$ with 256 grid points in each direction. The forcing mechanism is a Gaussian process, delta-correlated in time, with support in wavenumber space around $k_f = 4$. We fixed $\Omega = 8$, resulting in a Rossby number $Ro = E_{tot}^{1/2}/k_f\Omega \sim 0.1$, where E_{tot} is the flow kinetic energy. The dissipation is modeled by an hyperviscous term $\nu\nabla^4\boldsymbol{u}$, which replaces the laplacian in (1), with $\nu = 1.6 \times 10^{-6}$. A large scale friction term, $-\beta\boldsymbol{v}$ is also added to the right hand side of the equations in order to reach stationarity, with $\beta = 0.1$. In Fig. 3 (left) we show the energy spectrum $E(k)$ of the simulation, with an arrow denoting the scale where forcing is acting. It is easy to see how both ends of the spectrum get populated with active modes on almost two decades in wave-numbers. The inset of the figure shows the probability density function (PDF) of ω_3, the vorticity component parallel to the rotation axis. The distribution is highly skewed and non-Gaussian, favoring the alignment of vorticity with the rotation axis. Lastly, in the right panel of Fig. 3 we show a visualization the velocity amplitude $|\boldsymbol{v}|$ from where we can see the formation of large columnar vortices that cover the whole domain.

As shown in Fig. 1, for our inpainting experiments we do not use the whole three dimensional fields, but only two dimensional horizontal cuts in the plane (x_1, x_2). Furthermore, in order to decrease the computational cost of training the two CEs, the velocity images are downsized to $L \times L$ with $L = 64$, by applying a pass-band filter for $k < k_\eta$, where with $k_\eta \sim 32$ we indicate the Kolmogorov dissipative wavenumber, defined as the scale such that for $k > k_\eta$ the velocity field becomes smooth [53]. The above phenomenology show the whole complexity of our task: we want to reconstruct/assimilate data from 2d snapshots of 3d highly multi-scale and chaotic fields, characterized by strong intermittent and non-differentiable small-scale fluctuations superposed to big large-scale cyclonic and anti-cyclonic vortical structures. These are complex features strongly different from the ones characterising other inpainting tasks based on popular images data-base [14–19].

III. CONTEXT ENCODER FOR IMAGE GENERATION (CE1)

Let us first present the context encoder as suggested in [3] with reference to Figs. 2 and 4 and to appendix A for specific details. The idea is to have a Deep-GAN that reconstructs the missing part of the three velocity configurations. To that end we define the masking operator $\hat{M}$, which takes the value 1 on the missing pixels and 0 on the unaltered locations in our original image $\mathcal{I}_v$. We will then denote with $\hat{M} \odot \mathcal{I}_v$ the damaged part of $\mathcal{I}_v$ and with $(1 - \hat{M}) \odot \mathcal{I}_v$

the region of the flow where we have the full information, see Fig. 2 for a graphical representation applied to a set of three velocity components, $\mathcal{I}_v = (v_1, v_2, v_3)$. The overall architecture is based on a pipeline with first the generator (G) made of an *encoder* which takes as input the damaged image, $(1 - \hat{M}) \odot \mathcal{I}_v$, produces a *latent* representation of the relevant features with a down-sampling of the dataset, then in the second part of the network a decoder learns how to generalise the features and proposing a candidates for the three velocity components in the missing gap, $G[(1 - \hat{M}) \odot \mathcal{I}_v]$. The Generator part of the context encoder, panel (a) of Fig. 4, is trained to minimize the L_2 norm between the generated image and the ground truth inside the hole:

$$\mathcal{L}_{rec} = \mathbb{E}_{\mathcal{I}_v}\{||\hat{M} \odot \mathcal{I}_v - G[(1 - \hat{M}) \odot \mathcal{I}_v]||_2\}, \tag{2}$$

where $G[\bullet]$ indicates the snapshots of three velocity components provided by the network as output and with $\mathbb{E}_{\mathcal{I}_v}$ we intend average over the set of training configurations. The final network is supplemented with an adversarial structure, see panel (b) of Fig. 4, made of a discriminator (D) to provide loss gradients to the context generative encoder G. The learning protocol is a two-player game where the discriminator D is inputted alternatively with ground truth images in the hole or the ones generated by the context encoder G and tries to distinguish among them while the generator G tries to confuse D by proposing samples that appear as close as possible to real turbulent configuration in the missing region. The adversarial task for the discriminator component of the encoder is to *maximize* the binary cross-entropy function:

$$\mathcal{L}_{adv} = \mathbb{E}_{\mathcal{I}_v}\{\log(D[\hat{M} \odot \mathcal{I}_v]) + \log(1 - D[G[(1 - \hat{M}) \odot \mathcal{I}_v])\} \tag{3}$$

where $D[\bullet]$ is the output of the discriminator with values $\in (0, 1)$. The above expression is maximized when the discriminator is able to assign 1 to the real images and 0 to the ones proposed by G. We notice that in the second term of the above expression we only use data proposed by the generator G in the gap of size $\ell \times \ell$ and do not supply the whole image of size $L \times L$ to not help the discriminator to exploit potential discontinuities when passing from the hole filling to the external frame.
Finally, the total loss that must be *minimized* to train the Generator part of the CE1 is given by a linear superposition of the two components:

$$\mathcal{L} = \lambda_{rec}\mathcal{L}_{rec} + \lambda_{adv}\mathcal{L}_{adv}, \tag{4}$$

with $\lambda_{rec}+\lambda_{adv} = 1$, and where the adversarial loss is here evaluated only with the images produced by the Generator: $\log(1 - D[G[(1 - \hat{M}) \odot \mathcal{I}_v])$. In the following section we will present the results by varying the typical length scale, ℓ of the spatial hole where data are missing but keeping the overall damaged area the same (see section III A) or by varying the structure of the input data, using a multi-channel information made of three images of the three velocity components, v_1, v_2, v_3 or a single channel given by the vertical vorticity ω_3 (see section III B).

A. Image inpainting: large vs small scales information

We start by analysing the training performance and the final validation for the CE1 network by removing from the input a squared hole of size $\ell \times \ell$ with $\ell = L/2$ located at the centre of the image. Here we present data when we used in input three channels with the three velocity components in the plane and we later compare with the case when the total missing area is the same but each single missing spot is smaller.
In Fig. 5 we show the evolution of the reconstruction loss $\mathcal{L}_{rec}$, of the adversarial loss $\mathcal{L}_{adv}$ and of their weighted sum (4) obtained with $\lambda_{adv} = 0.001$, measured on the validation dataset, as a function of the epochs during training, for details see appendix A. For training we have used a total number of configurations $N_{tra} = 81920$ while for validation we used a set of $N_{val} = 20480$ configurations never seen during training. As one can see from Fig. 5, the training converges rapidly without any over-fitting. In the inset of the same Fig. 5 we show the quality of the gap filling at different epochs, to give a visual demonstration of the ability of CE1 to progressively learn the correct flow structures.
We now proceed with a more quantitative analysis of the CE1 performances using the network frozen at its final epoch configuration. In Fig. 6 we show a series of 3 different gap filling experiments using configuration never showed during training as well as a point-wise analysis of the normalized L_2 norm calculated on vertical lines for each fixed horizontal position, x_1:

$$\Delta_v(x_1) = \frac{\ell^{-1}\sum_{x_2}\sum_{i=1}^{3}[v_i^{truth}(x_1, x_2) - G_i(x_1, x_2)]^2}{E_{tot}} \tag{5}$$

where $G_i(x_1, x_2)$ for $i = 1, 2, 3$ are the three components of the field generated by CE1 inside the missing region. The above expression is normalized dividing by the averaged flow energy over a subset of $N = 2048$ of the validation dataset

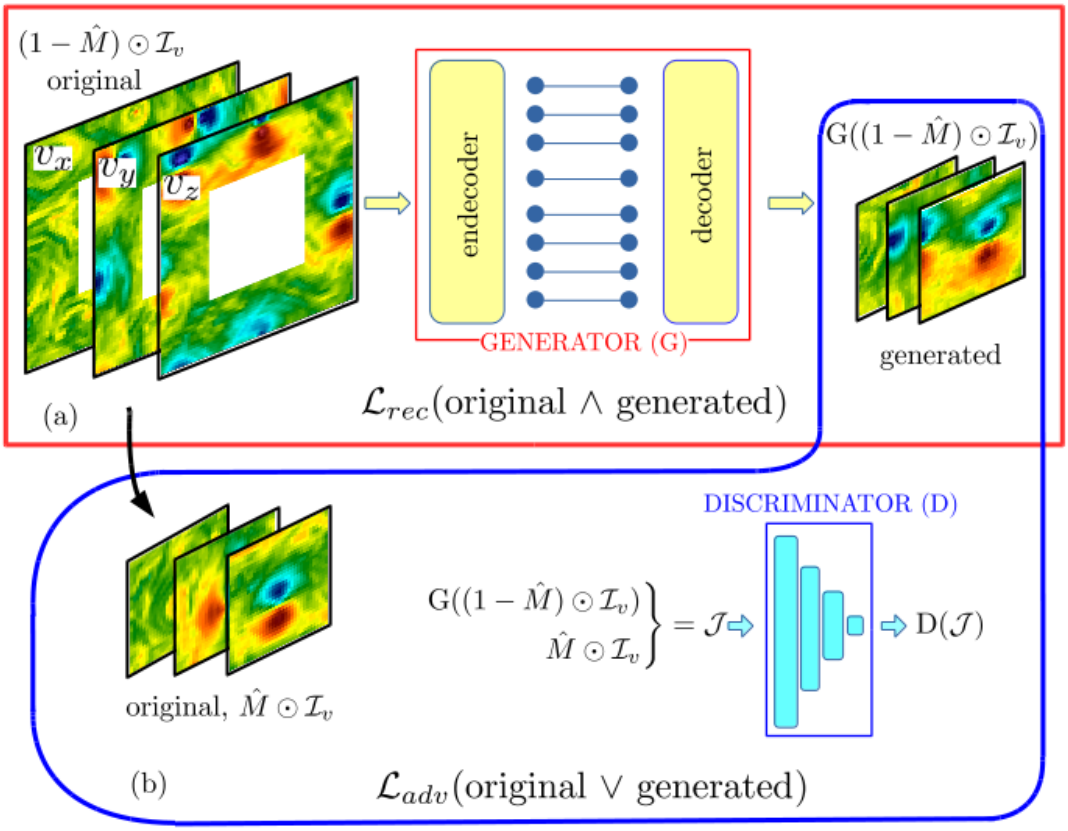

FIG. 4: Sketch of the Context Encoder (CE1) configuration. The machine is made of two blocks. A Generator (G), depicted in panel (a), where we give in input the three channels made of the three corrupted configurations of size $L \times L$ with $L = 64$ for the three velocity components in the plane, $v_1(x_1, x_2), v_2(x_1, x_2), v_3(x_1, x_2)$ and obtain as output the fields proposed to fill the gap of size $\ell \times \ell$. The second block is given by a Discriminator (D), see panel (b), which is trained to distinguish the original fields from the ones proposed by the generator for the hole region. See appendices A and B for details about the internal structure of G and D.

that are the configurations used in our reconstruction experiment: $E_{tot} = L^{-2} \sum_{x_1,x_2} \sum_{i=1}^{3} \langle [v_i^{truth}(x_1, x_2)]^2 \rangle_N$, where we used $\langle \bullet \rangle_N$ for a shorthand notation of the average. It is interesting to notice (5th column, red line) that the local normalized L_2 error for each snapshot is of the order of $10 - 20\%$, when we are in the middle of the missing region, showing a very good reconstruction property of our CE. In the same panels we plot the averaged errors, $\langle \Delta_v(x_1) \rangle_N$. As on can see, the max averaged error is $O(15)\%$ (black lines). Notice also that the error slightly improve for horizontal position close to the gap edge, indicating that the network is able to detect local correlations also. In Fig. 7 we test the CE1 at changing the spatial distribution of the missing data, by distributing the same amount of total gaps but on squares with smaller and smaller size $\ell/L = 1/2, 1/4, 1/8$. As one can see from the black curves in the last column of the same figure, the averaged errors, $\langle \Delta_v(x_1) \rangle_N$, become smaller and smaller by decreasing ℓ, reaching a maximum averaged error as small as $3 - 4\%$ for $\ell/L = 1/8$. It is important to stress that this is a very good result, being the corrupted size still much bigger than the differentiable limit given by the Kolmogorov scale $\eta = 1/k_\eta \sim \ell/L = 1/64$, (see Fig. 3). In other words, we always work in the scale range where the fields are rough and non-differentiable with Holder continuity close to 1/3 [53], making the problem much different from the one of non-linear local fit that other classical inpainting methods can handle well [62–65]. In the rightmost panel of Fig. 8 we show the most stringent test one can perform, checking the point-by-point reconstruction properties by plotting the probability distribution function of the local L_2 error:

$$\Delta_v(x_1, x_2) = \frac{\sum_{i=1}^{3} [v_i^{truth}(x_1, x_2) - G_i^v(x_1, x_2)]^2}{E_{tot}}.$$

The panel shows that we obtain a pretty good performance to assimilate data even local-wise with very small probability to make big errors. Furthermore, when ℓ/L becomes small there is a sharp improvement for the *worst case* as shown by the sharp reduction in the right tail extension when comparing $\ell/L = 1/8$ with $1/2$.

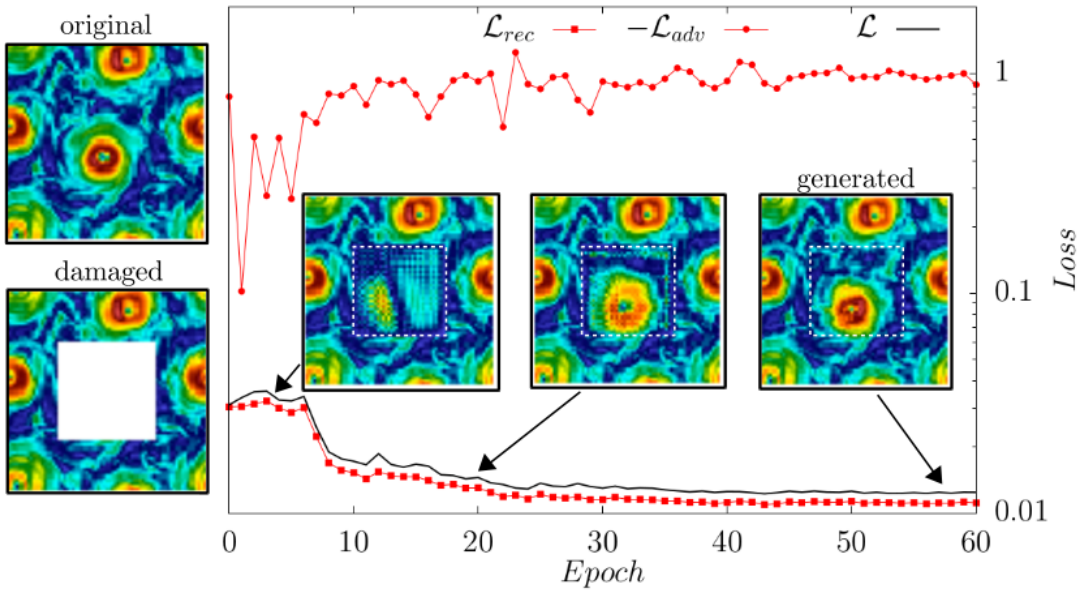

FIG. 5: Example of a training section for CE1. We show the evolution of $\mathcal{L}$, $\mathcal{L}_{rec}$ and $-\mathcal{L}_{adv}$, measured on the validation dataset, as a function of the epoch as well as the reconstructed configuration for $|\boldsymbol{v}|$ at three different steps during training. Notice the tendency to supply better and better inpainting by advancing the learning protocol. The two pictures on the left represent the ground truth in the whole plane and the damaged input.

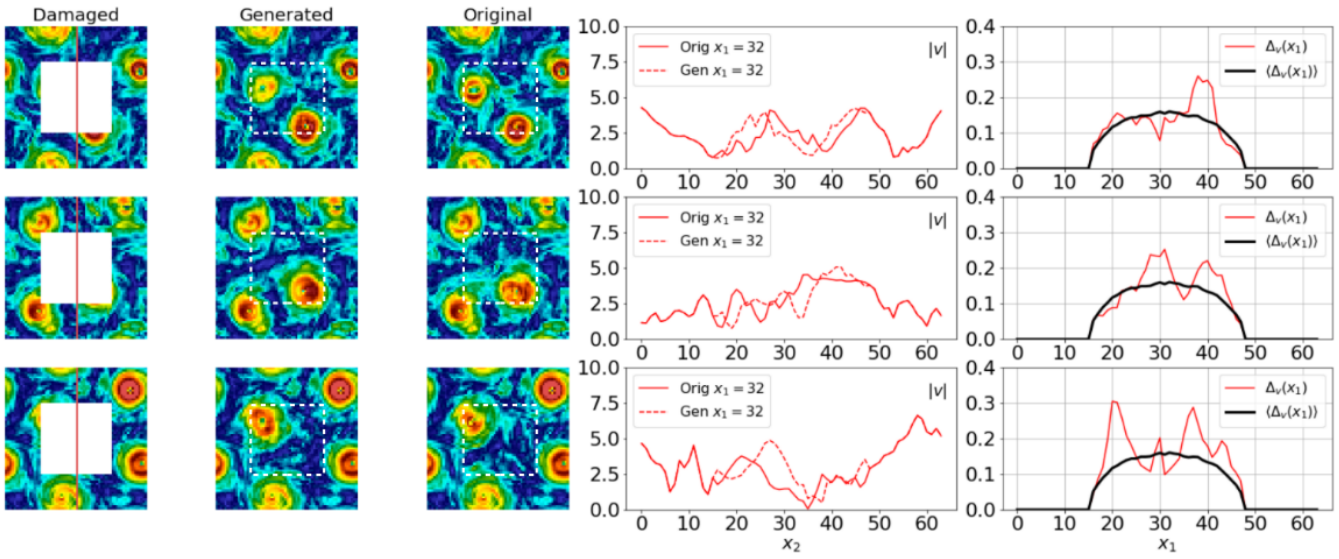

FIG. 6: Validation of gap filling by CE1 for three different generated fields (one for each row). Images show the magnitude of local velocity field, $|\boldsymbol{v}(\boldsymbol{x})|$. 1st column: damaged image in input. 2nd column: image generated in output. 3rd column: ground truth. 4th column: generated (dashed) and ground truth (solid) profiles of $|\boldsymbol{v}(\boldsymbol{x})|$ along the vertical line shown in the 1st column. 5th column: L_2 error, $\Delta_v(x_1)$ vs x_1, given by expression (5) for each image (red line) and the average error $\langle\Delta_v(x_1)\rangle_N$ over $N = 2048$ different images (black curve).

In Fig. 8 we perform also two less stringent statistical test concerning the reconstructed flow. First, in the left panel we show the 2d spectrum:

$$E(k) = \sum_{k<\mathbf{k}<k+1} \langle \hat{v}_i(\mathbf{k})\hat{v}_i(-\mathbf{k})\rangle_N \tag{6}$$

where $\mathbf{k} = (k_1, k_2)$ and we calculated the Fourier transform of the velocity field, $\hat{v}_i(\mathbf{k})$, over the whole 2d configuration with the gap filled as proposed by the output of CE1. In the same panel we also show the spectrum calculated

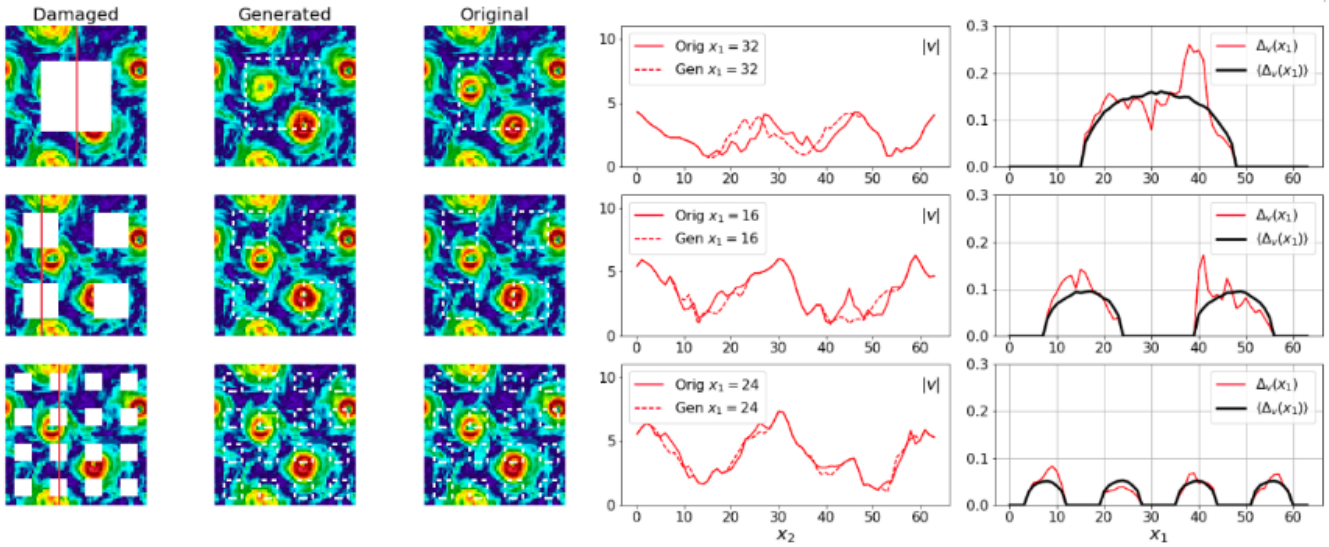

FIG. 7: The same as in Fig.(6) at changing the typical size of the damaged region, $\ell/L = 1/2, 1/4, 1/8$ from top to bottom rows.

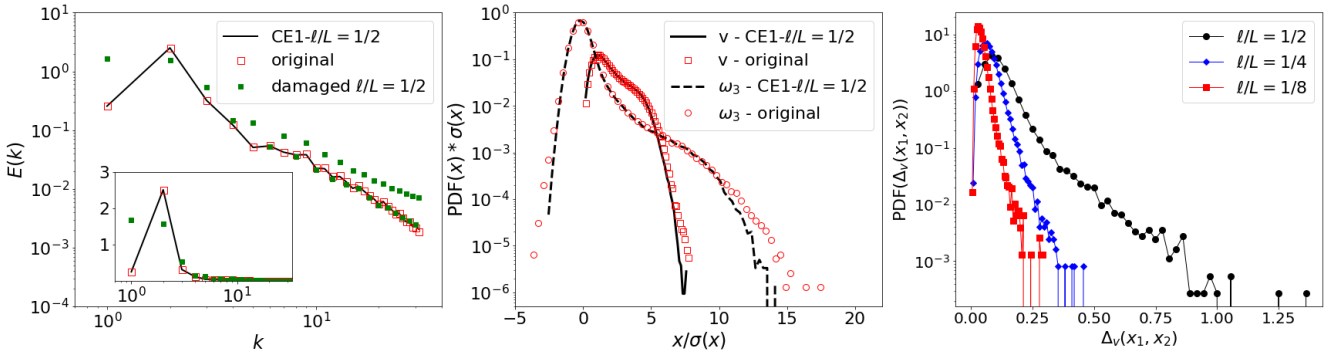

FIG. 8: Left: comparison between the 2d spectrum (6) of the ground truth original DNS data in the whole 2d plane and the data reconstructed by the CE1 with $\ell/L = 1/2$, as well as the spectrum calculated on the damaged image; the inset shows the same data in log-lin scale. Middle: comparison of standardised PDFs of velocity magnitude, $v = |\boldsymbol{v}|$ and vorticity ω_3. Right: PDF for the point-wise L_2 error, $\Delta_v(x_1, x_2)$ at changing ℓ/L.

on the corrupted input to quantify how much information (energy) we miss at different scales. Curves here are averaged over $N = 2048$ configurations. In the same figure, middle panel, we show the probability distribution functions (PDF) for the velocity magnitude, v, and for the out-of-plane vorticity, ω_3. All comparisons between generated and the ground truth are very good even for the strongly non-Gaussian distribution enjoyed by the ω_3 field.

B. Features ranking

In this section we discuss a different question, connected to use DA tools as a reverse engineering approach to perform features ranking, i.e. to assess the quality and quantity of the input data on the basis of the obtained output. To do that, one can change the set of fields supplied in the CE input and/or change the definition of the loss function in order to optimize some features of the generated output. Let us suppose, for example, that we want to reconstruct with high accuracy only the out-of-plane vorticity field, ω_3. One question one can ask is that weather it is better to work with (i) the three velocity channels as input and optimising the reconstruction against the corresponding three velocity outputs in the missing gap, as done in the previous section, (ii) w_3 as input and output (one channel only) or (iii) using v_1, v_2, v_3 as input but adding in the loss also a penalization against bad reconstruction of gradients. The idea is to test the importance of having both large and small-scales information in the CE1 input/output, i.e.

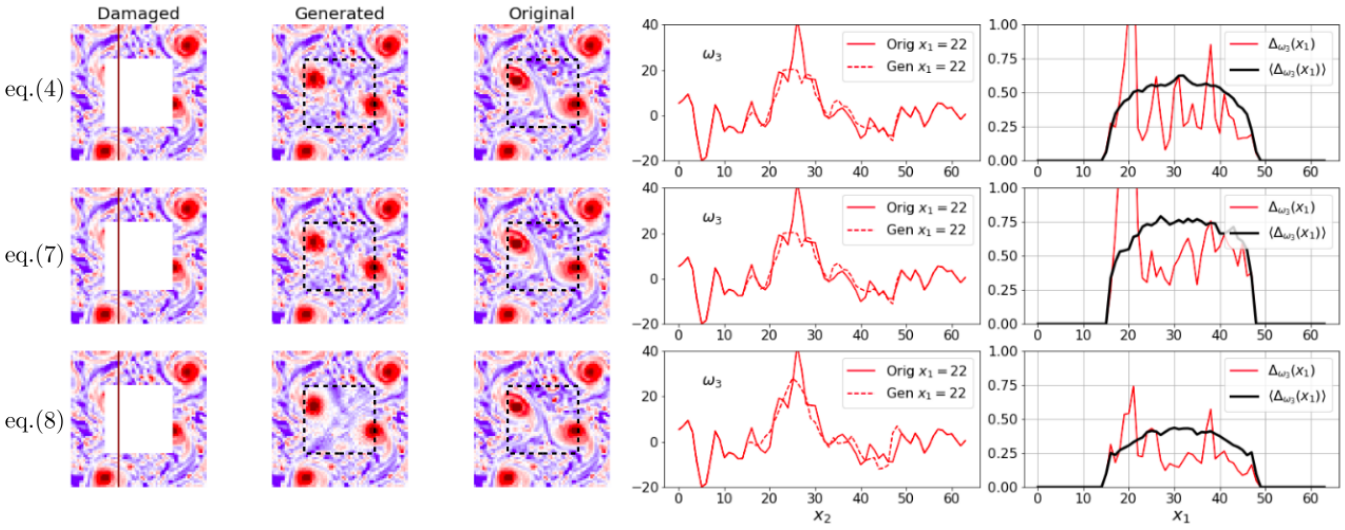

FIG. 9: Vorticity reconstruction with CE1. Comparison for vorticity generation using loss given by case (i), Eq. (4) first row; obtained by using vorticity in input and in output as in case (ii), Eq. (7), second row; and for case (iii), Eq. (8) third row. Errors are defined following Eq. (5)

semantic contents that have larger and smaller *influence size*. The two above cases labelled with (ii) and (iii) would correspond to the following loss function:

$$\mathcal{L}_{rec} = \mathbb{E}_{\mathcal{I}_\omega} ||\hat{M} \odot (\mathcal{I}_\omega - G[(1-\hat{M}) \odot \mathcal{I}_\omega])||_2 \tag{7}$$

for case (ii) and to

$$\mathcal{L}_{rec} = \alpha \mathbb{E}_{\mathcal{I}_v} \{||\hat{M} \odot (\mathcal{I}_v - G[(1-\hat{M}) \odot \mathcal{I}_v])||_2\} + (1-\alpha)\mathbb{E}_{\mathcal{I}_v} \{||\hat{M} \odot (\omega_3 \odot \mathcal{I}_v - \omega_3 \odot G[(1-\hat{M}) \odot \mathcal{I}_v])||_2\} \tag{8}$$

for case (iii), where with $\omega_3\odot$ we intend the operation to extract the ω_3 component out of the three v_1, v_2, v_3 original fields, $\mathcal{I}_v$ or of the ones generated by $G[\bullet]$ and we have introduced a parameter $\alpha \in [0:1]$ that weighs the loss due to reconstruction of $\boldsymbol{v}$ and the one due to ω_3. In the first row of Fig. (9) we show the results for vorticity reconstruction for case (i), with loss given by Eq. (3), while in second and third rows we show the same but for cases (ii) and (iii), which correspond to losses given by Eqs. (7) and (8), respectively. The latter obtained with mixing parameter $\alpha = 0.5$. Comparing results from first and second row, $\langle\Delta_{\omega_3}(x_1)\rangle \sim 55\%$ and 75% respectively, we see that in order to have an improved reconstruction of vorticity it is better to train with the whole set of three velocity field (first row, case i) than using vorticity directly (second row, case ii). This is somehow natural because for the former we have more information in input than in the latter. Moreover, vorticity is a highly skewed and non-Gaussian variable [53], properties that could be problematic to handle as input in the network as done in case (ii). Conversely, by comparing first and third rows, cases (i) and (iii), we learn that keeping the three velocity in inputs (max information) and penalising the output in terms of the L_2 norm for vorticity does help to give a better reconstruction decreasing $\langle\Delta_{\omega_3}(x_1)\rangle$. Let us notice, however, that the final result is never exceptionally satisfying, with averaged errors in the middle of the missing region that are always around $35\% - 40\%$, at the best. From the right panel of Fig. 10 we can indeed see that even for the better results, case (iii), the PDF of the point-wise reconstruction error has now a pretty extended right tail, indicating the partial failure to have a good local reconstruction, even though the spectral and PDF for the vorticity field are satisfactorily reproduced (left and middle panel). The difficulty to reconstruct point-wise the correct vorticity properties can be due to different reasons. First, the field is strongly non-Gaussian and with a very small spatial correlation length of the order of η, hence much smaller than ℓ. Second, as one can see from the first three columns of Fig. 9 it is enough a small mismatch for the position of the vorticity peaks between the ground truth and the reconstructed images to produce a big *local* error while the overall image is still very close to the original one. Third, vorticity field is obviously much more sensitive to small details, with a Fourier spectrum enhanced by a factor $\propto k^2$ wrt the kinetic energy case, something that is particularly critical in the presence of rough non-differentiable fluctuations as in our turbulent case.

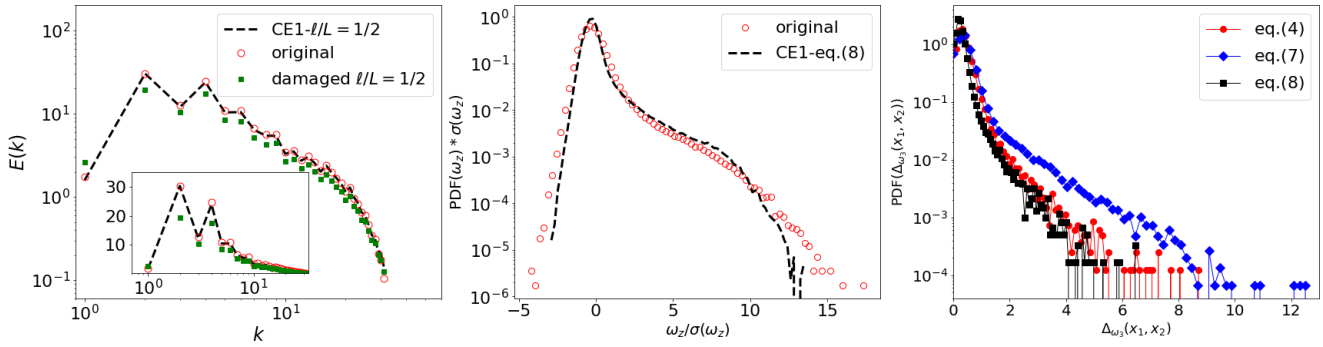

FIG. 10: Left: comparison between spectrum of vorticity for ground truth (original DNS data in the whole 2d plane), for the data reconstructed by the CE1 with $\ell/L = 1/2$ using the protocol based on the loss function of case (iii) given by Eq. (7) and for the spectrum calculated on the damaged input images, inset the same but in log-lin scale. Middle: comparison for standardised PDFs of vorticity ω_3. Right: PDF for the point-wise L_2 error, $\Delta_v(x_1, x_2)$ using the three different CEs trained with the loss functions for cases (i-iii).

IV. CONTEXT ENCODER WITH BACK-PROPAGATION TO THE INPUT DATA (CE2)

The second context encoder we analyze is based on the concept of constrained image generation [4]. The idea proceeds in two steps. First a GAN is trained to generate uncorrupted flow configurations in the whole 2d domain, see Fig. 11. The generation is conditioned on a set of random input values, $\mathbf{z} = (z_1, z_2, \ldots, z_T)$, with $T = 100$ in our case, which are transformed by the trained machine in realistic flow configurations, at least as realistic as being not distinguishable from the discriminator in the GAN itself. Then, the GAN is frozen and for the image reconstruction task a search for the optimal set of $\mathbf{z}$ inputs is performed, asking the machine to reproduce with high fidelity only the uncorrupted region. The idea is that once the GAN is trained to generate the entire turbulent snapshots, we just need to search for the closest configuration that can be generated by the machine and that matches the known data, $(1 - \hat{M}) \odot \mathcal{I}_v$. Moreover, also a penalty against generation of patches that are disconnected by the frame will be added, via the insertion of a discriminator loss (see below and Fig. 11). At difference form CE1, here we do not use the structure of the mask for training. After the Deep-GAN has been trained [66–68] we proceed to recover the optimal $\mathbf{z}^*$-encoding that is closest to the corrupted image. After $\mathbf{z}^*$ is obtained we can input it to the GAN to obtain the whole configuration also inside the gaps. In practice, in order to fill one damaged field, $(1 - \hat{M}) \odot \mathcal{I}_v$, we obtain $\mathbf{z}^*$ by *minimizing* the combined loss:

$$\mathcal{L} = \mathcal{L}_{rec} + \lambda_{adv}\mathcal{L}_{adv} \tag{9}$$

where

$$\mathcal{L}_{rec} = ||(1 - \hat{M}) \odot (\mathcal{I}_v - G[\mathbf{z}])||_2 \tag{10}$$

by back propagating with respect to the input data: $-\partial_{\mathbf{z}}\mathcal{L}_{rec}$, where we need to notice that in (10) we are using information only on the available data by applying the mask $(1 - \hat{M})$ to all velocity components. The second term in (9) is giving a penalty if the generated image on the whole domain does not look similar to a real turbulent configuration, by applying the judgement of the (previously trained) discriminator D:

$$\mathcal{L}_{adv} = \log(1 - D[G[\mathbf{z}]]) \tag{11}$$

where $\lambda_{adv} = 0.1$ is a parameter used to balance between the two losses and where the input data are updated to fool D, $-\partial_{\mathbf{z}}\mathcal{L}_{adv}$. In Fig. 12 we show the equivalent of Fig. 7 for three different experiments using CE2. As one can see from data in column 5, now there is a non zero error, $\Delta_v(x_1)$ also outside the missing region because CE2 generate the flow in the whole domain. The overall performances have a maximal averaged error $\langle\Delta_v(x_1)\rangle$ of the order of 40% roughly a factor 2 larger than CE1. It is important to stress again that we didn't perform any systematic optimization of network structures and/or learning parameters being primarily interested here to the proof of concept aspects. In Fig. 13 we show the comparisons against the DNS (ground truth) for spectra, velocity and vorticity PDFs, again with good overall results but worse performance with respect to CE1, especially concerning the PDF of the local L_2 norm (right panel), which presents now a fatter right tail.

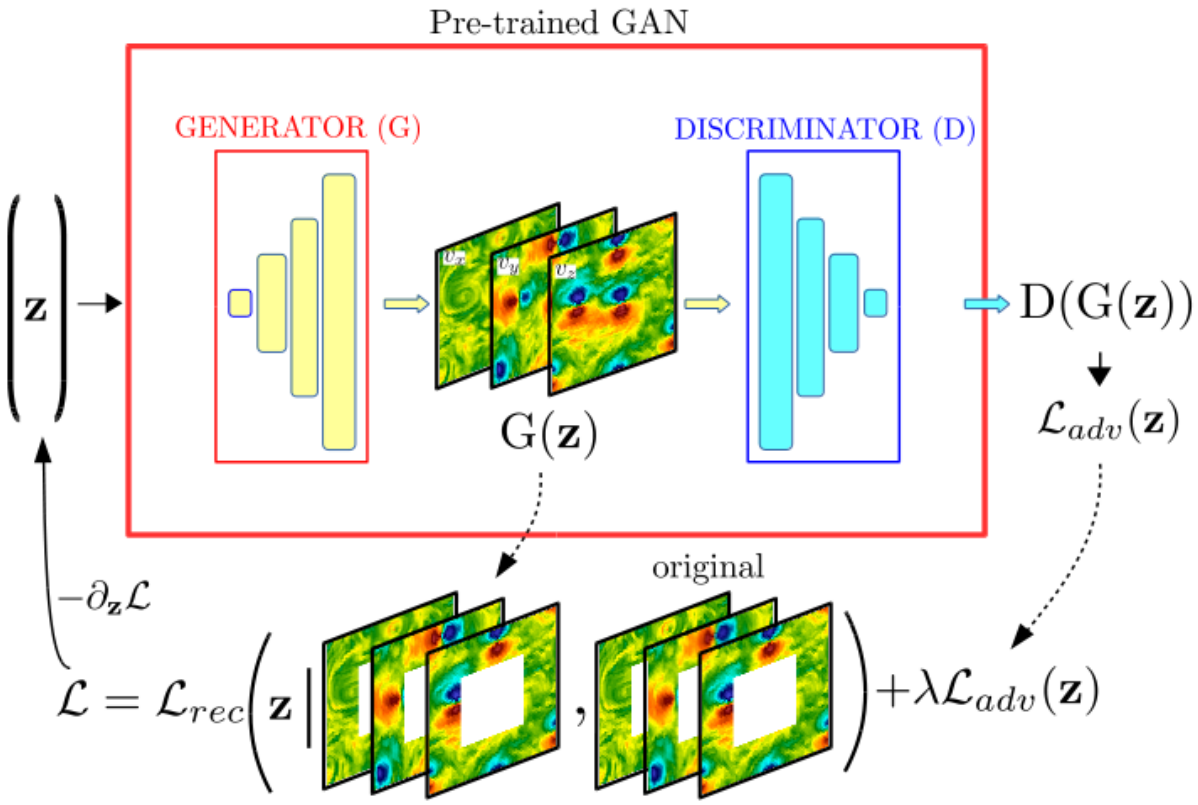

FIG. 11: Sketch of the Context Encoder (CE2) configuration. The machine is made of a typical Deep-GAN structure [4, 66] which is pre-trained to generate output images on the whole 2d plane, $G[\mathbf{z}]$ from a random input $\mathbf{z}$. To apply it for gap-filling, the GAN is frozen and a back-propagation to the input data is implemented. Only $\mathbf{z}$ is evolved to match the given corrupted image, $(1 - \hat{M}) \odot \mathcal{I}_v$, where we have data.

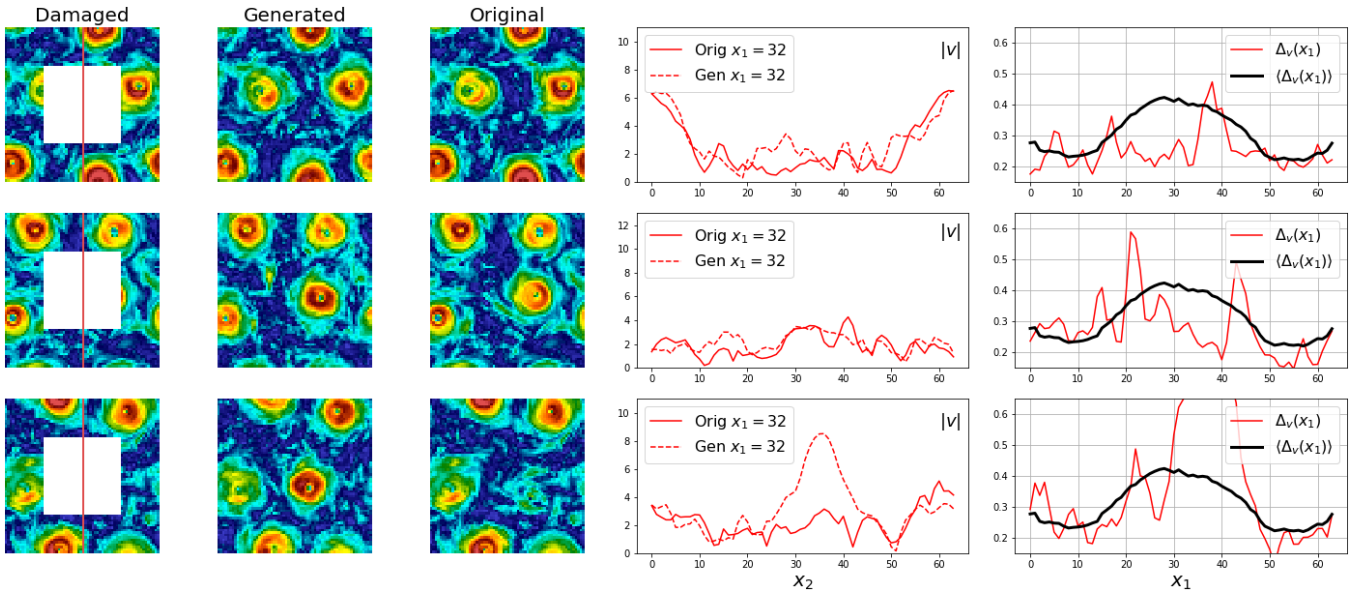

FIG. 12: The same of Fig. 6 but for CE2. Notice the 3rd row, where the network puts a vortex in the gap region which is not present in the original case, hence the big error $\Delta_v(x_1)$ for $x_1 \sim 40$.

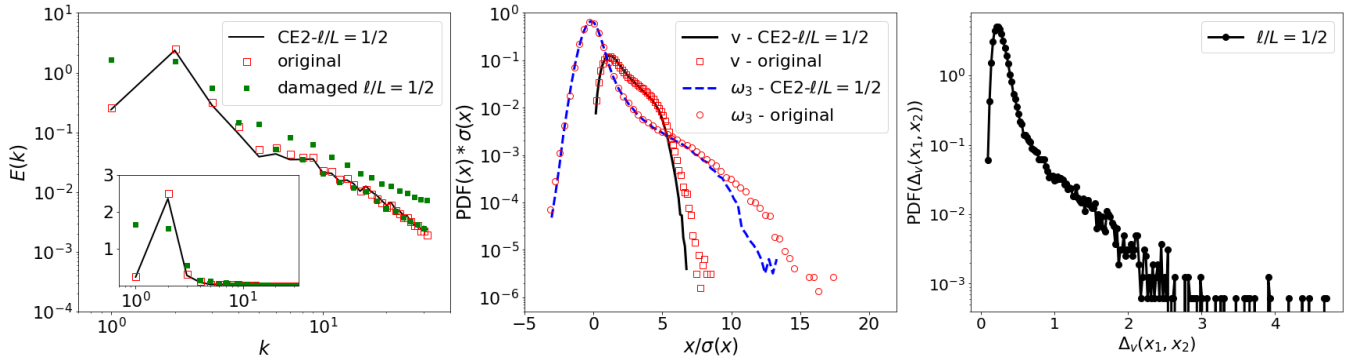

FIG. 13: The same as in Fig. 8 but for CE2. Comparison for spectra (left), PDF of velocity and vorticity (middle). Right: point-wise L_2 error calculated in the whole area.

V. EQUATION-INFORMED DA: NUDGING

In this section we confront the results obtained with CE1 and CE2 against what one can achieve using Nudging [23–25], a DA scheme popular among numerical weather forecast community and in economic fields [69]. Nudging is distinguished from the methods presented above in two senses: it relies entirely on solving the equations of motion and it is also applicable to reconstruct trajectories in phase space. The scheme works by introducing a relaxation term to the equations of motion, penalizing the flow when it deviates from a given reference state, set by the data one wants to reconstruct/assimilate. In rotating turbulence it takes the following form:

$$\frac{\partial \boldsymbol{v}}{\partial t} + \boldsymbol{v}\cdot\boldsymbol{\nabla}\boldsymbol{v} + 2\Omega\hat{x}_3\times\boldsymbol{v} = -\boldsymbol{\nabla}p + \nu\nabla^2\boldsymbol{v} + \gamma(1-\hat{M})_{x_3}\odot(\boldsymbol{v}-\boldsymbol{v}_{\rm ref}) \tag{12}$$

where $\gamma = 100$ is the intensity of the nudging, $(1-\hat{M})_{x_3}\odot$ is the same filtering operator used in the previous section, replicated identically along the x_3 direction, such that the nudging term acts only on the locations where the reference damaged flow, $\boldsymbol{v}_{\rm ref}$, is known. The idea behind the method is simple, instead of using a machine to learn (local and non-local) information in the 2d images we ask the NSE to provide the *semantic* background to fill the gaps, providing the whole 3d field in the whole volume $\boldsymbol{v}(\boldsymbol{x},t)$ and for all times. Notice that NSE need to be evolved on the original volume size, even if the input is masked to be a subset of a 2d slice. Notice also that usually Nudging is used to reproduce also temporal evolution, while here we will implement it to reconstruct one given, frozen in time, 2d velocity snapshot (which is replicated identically along the x_3). Another important difference wrt Context Encoders is that we do not need any training section using hundred thousands examples from the ground truth as in ML applications, provided that the equations used are the ones that have also generated the corrupted images. It is also key to realize that we do not need to supply any external mechanical forcing mechanism to (12), i.e. the reconstruction is obtained without knowing the external stirring. Nudging was shown to be able to reconstruct, with different degrees of accuracy, high Reynolds number three dimensional turbulent flows [25], two dimensional flow [70, 71] and Rayleigh-Bernard convection [72]. Moreover, Nudging can make up for inferring missing terms in the equation, such as forcing or even rotation [73]. In the following we show the performance of nudging to reconstruct missing hole in the middle of a 2d slice exactly as we did before for the two Context Encoders. We performed a long numerical integration of (12) on the original box size, with 256^3 collocation points and supplying the 2d damaged input given by $(1-\hat{M})_{x_3}\odot\mathcal{I}_v$, where $\mathcal{I}_v$, denoted here $\boldsymbol{v}_{ref}$ to follow the notation used by the community, is the three velocity components in a 64^2 2d plane (x_1,x_2). As said, the image is extended along the vertical direction in order to form a 3d volume (but no vertical fluctuations are introduced as we just stack up the same image). The resulting reference field for the nudging algorithm is constant in time. In this way we are not providing any extra information (vertical or temporal fluctuations) compared with the CE1 and CE2 cases discussed above. With this in mind, the result coming from the nudging is obtained as an average both in time and along the vertical direction.

In Fig. 14 we show the results of one of our experiments at changing ℓ/L following what done for the CE1 and shown in Fig. 7. As for CE2 case, nudging is introducing errors also where data are supplied, because NSE evolves in the whole (3+1)d domain. From the data shown in column 5, we can see that the averaged error $\langle\Delta_v(x_1)\rangle$ is more noisy wrt to the CE1 and CE2 corresponding cases. This is due to the fact that we performed only three experiments with different nudging fields because of the much heavier computational overhead introduced by the need to solve NSE for each test. Nevertheless, we notice that the overall errors are of similar order wrt to Figs. 7-12, although some small-scale reconstruction deficiencies can be spotted for the bigger gap case $\ell/L = 1/2$ (first row). Some of

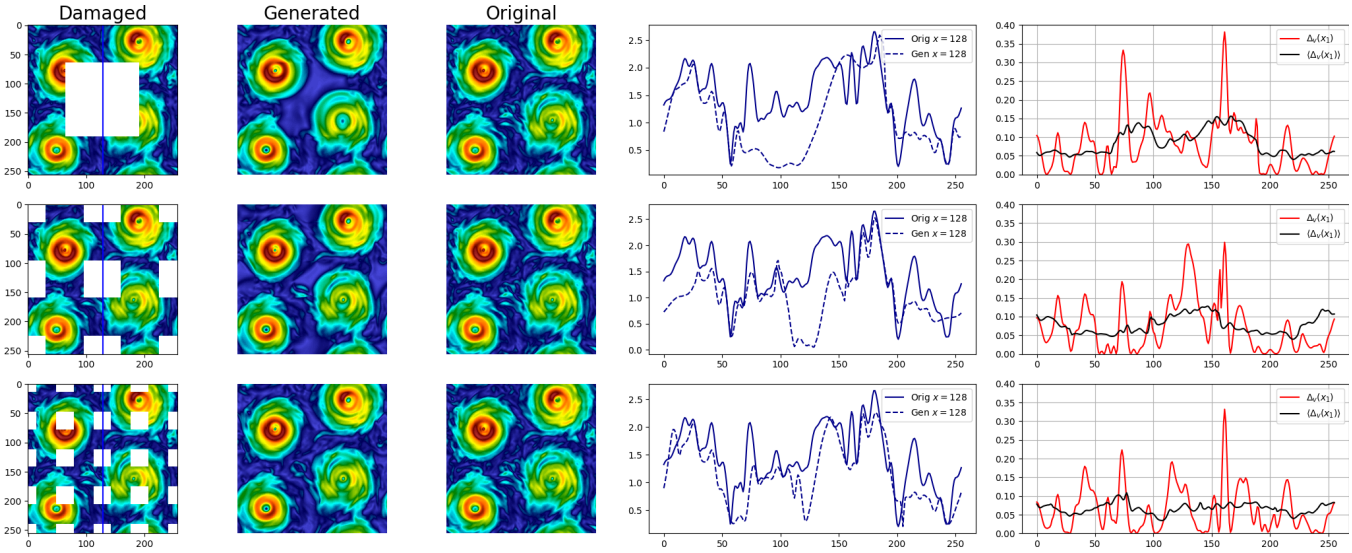

FIG. 14: The same as in Fig.(7) but for nudging. Here in the 2nd column we show the average in time and along the x_3 direction of the nudged-NSE (12) output, $\langle |\boldsymbol{v}(x_1, x_2)|^2 \rangle_{t,x_3}$. In the 5th column, the black curve is obtained with a further average over three different nudging experiment.

these problems could be mitigated if the reference field is not constant in time [25].

VI. CONCLUSIONS

We have presented a series of investigations meant to test the applicability of tools widely used by the CV community for image inpainting to turbulent data reconstruction from the TURB-Rot database [21]. At difference from the typical set of images database where CV algorithms are trained, our case is made of a series of complex *scenes*, with random properties and multi-scale fluctuations. We have compared two different Context Encoders, called CE1 and CE2. The main difference from the two is that CE1 is able to provide (once trained) a guess for the missing data without any further inference. Contrary, CE2 is based on a two step training, first the machine learns how to generate ungappy images and second it specialises to further fill the gap, needing a new training phase for each new filling. Results among the two are very satisfactory with CE1 performing better, even though a caveat must be made because we didn't try to push the hyper-parameters settings to a level that cannot be improved further. To this end we provide the whole database online and open such as to trigger the interested community to improve our results [21]. We found that the two algorithms are good also for *semantic* filling, i.e. when a big part of the image is missing. We found that large-scale features (velocity components) are reconstructed better than small-scale ones (vorticity) and that for the latter it is key to provide the whole set of velocity components in input to improve the inpainting.

Finally, we compared ML tools with Nudging, an equation-informed tool well established among the applied communities in numerical weather forecasts, geophysics and others. For our application, Nudging requires a new simulation of a 3d fully developed turbulent flows for each data reconstruction experiment and thus it is computationally much heavier than the two ML algorithms (if already trained). The trade-off is that Nudging will provide for a field reconstruction that is fully respectful of all symmetries and kinematic constraints enjoyed by the PDE, something that is not always easy to achieve with Convolutional Neural Networks as discussed in the introduction. Another advantage of Nudging is that it does not need training, it works even with one damaged configuration only, it relies on the temporal evolution of the NSE to explore the phase space and provide a realistic prior for the missing information (see [74] for a recent attempt to fuse nudging with recurrent neural network). It would be interesting to compare it with other ML tools that are also based on one single image analysis to infer the statistics of the missing region [5, 75]. It is important to notice that the application we have selected, fully developed turbulence with rotation, presents highly non-trivial aspects, e.g. multi-scale non-differentiable fields with fluctuations over 2-3 decades, without a mean flow, or boundaries. As a result, many techniques based on texturing [64], non-linear interpolation [76] or modal analysis [52], as well as patch methods that search similar patterns in the available image [63] could be in troubles to get to the same accuracy. Moreover, we reconstruct only one slice of the whole 3d domain, so without any hint on the 3d features (or on temporal evolution). In conclusions, we have

provided a first attempt for flow reconstruction by using state-of-the-art context encoders based on Deep-GAN, and applying it to a paradigmatic *hard* problem for fluid dynamics. When applying ML tools to turbulence, we need to use quantitative benchmarks to assess the performance of different networks, being visual comparisons or the spectral properties obviously not enough for such complex physics. In particular, for DA and tools there is the need to assess with local L_2 norm the goodness of the results, being interested to reconstruct non-Gaussian fluctuations in the *correct position* and with the *correct intensity*.
We hope this study will trigger other systematic validation and application of ML algorithms to complex turbulent features.

Appendix A: CE1

For the implementation of the CE1 we used TensorFlow [77] libraries with a Python interface, optimized to run on GPUs. As introduced in Sec. III, the overall architecture is composed by a Generator plus an adversarial Discriminator network. The Generator can be further divided into an encoder and a decoder part, see Fig.15. The encoder is asked to identify all the relevant features of the three damaged images corresponding to the three velocity components. We have an input of size $64 \times 64 \times 3$, where we masked with 0 values the gap region of size $\ell \times \ell$. The encoder is based on several convolutional layers with stride 2 that reduce the dimension of the input up to a 4×4 image. The last layer of the encoder, composed of 256 convolutional filters, is reshaped in a way to form a one-dimensional *features vector* of size 4096 given in input to the decoder part of the generator. A 20% dropout is added to the features vector such as to reduce over-fitting. The decoder architecture that completes the Generator part, is symmetric with respect to the encoder, and it is made of several layers of transposed convolutions such as to bring the features vector to the size of the original missing region, in our case of $32 \times 32 \times 3$. A first evaluation of the quality of the generated patch is measured by the L_2 norm of the difference between the generated and the original portion of the image, as we shown in eq. (2). The quality of the generated image is generally blurred and to reach a sharper reconstruction a joint loss with an adversarial architecture is required [3]. The discriminator, shown at the bottom of Fig.15, is asked to recognize the original from the generated image without looking at the context but only comparing the data in the missing region and the generated part [3]. The discriminator architecture similarly to the encoder is based on several convolutional layers that transform the input of size $32 \times 32 \times 3$ to a one-dimensional vector that is then transformed by a fully connected layer and a softmax activation function to the binary probability of the input image to be original or generated. All convolutional layers in the network are connected with LeakyRELU activation functions with a coefficient of 0.2. The total loss defined in eq. (4) is then constructed by weighting the reconstruction and the adversarial terms with the following factors, $\lambda_{rec} = 0.999$ and $\lambda_{adv} = 0.001$. From the evolution of the two losses during a typical training experiment as shown in Fig. 5, we can see that $\mathcal{L}_{adv}$ is almost two orders of magnitude higher than $\mathcal{L}_{rec}$, hence from the choice of λ_{adv} and λ_{rec} discussed above we can conclude that the discriminator loss was found to be optimal when its weight is around 10% of the total loss $\mathcal{L}$. It is important to stress that to achieve a good training with the factors discussed above, we have normalized the input data to be in the $[-1; 1]$ range, namely we rescaled the intensity of each pixel by subtracting first the mean, $m = \frac{\max(\mathcal{I}_v)+\min(\mathcal{I}_v)}{2}$, and then normalising by $\delta = \frac{\max(\mathcal{I}_v)-\min(\mathcal{I}_v)}{2}$, where the maximum and the minimum are calculated over the whole training set of size N_{tra}. The Generator output was then shifted and rescaled back to match the original range of the velocity fields. To reach a better balance between the Generator and the Discriminator parts we used a learning rate for G 2 times higher than that of the D. In particular the context encoder was trained with a learning rate of 2×10^{-4} while the discriminator learning rate was fixed at 1×10^{-4}. The optimizer used to train the network is ADAM with parameters $\beta_1 = 0.5$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$, see [79]. The mini-batch used in our training is composed of 128 images. Hence, because $N_{tra} = 81920$, each epoch consists of 640 mini-batches. After every mini-batch is analysed the network weights are updated back-propagating the mean error estimated on the mini-batch. For the application to case (ii) of Sec. III B the structure of the network is the same except that in input and output we have one image instead of three.

Appendix B: CE2

As introduced in Sec. IV, the CE2 architecture is a GAN composed by a Generator plus an adversarial Discriminator network, as shown in Fig.16. We started from the code in [80], which is a python program using the TensorFlow library, with minor changes to adapt it to our dataset. In detail, the generator part is a network aimed to produce a 3-dimensional velocity field starting from an arbitrary vector, $\mathbf{z}$, with T-real entries ($T = 100$ in our implementation)

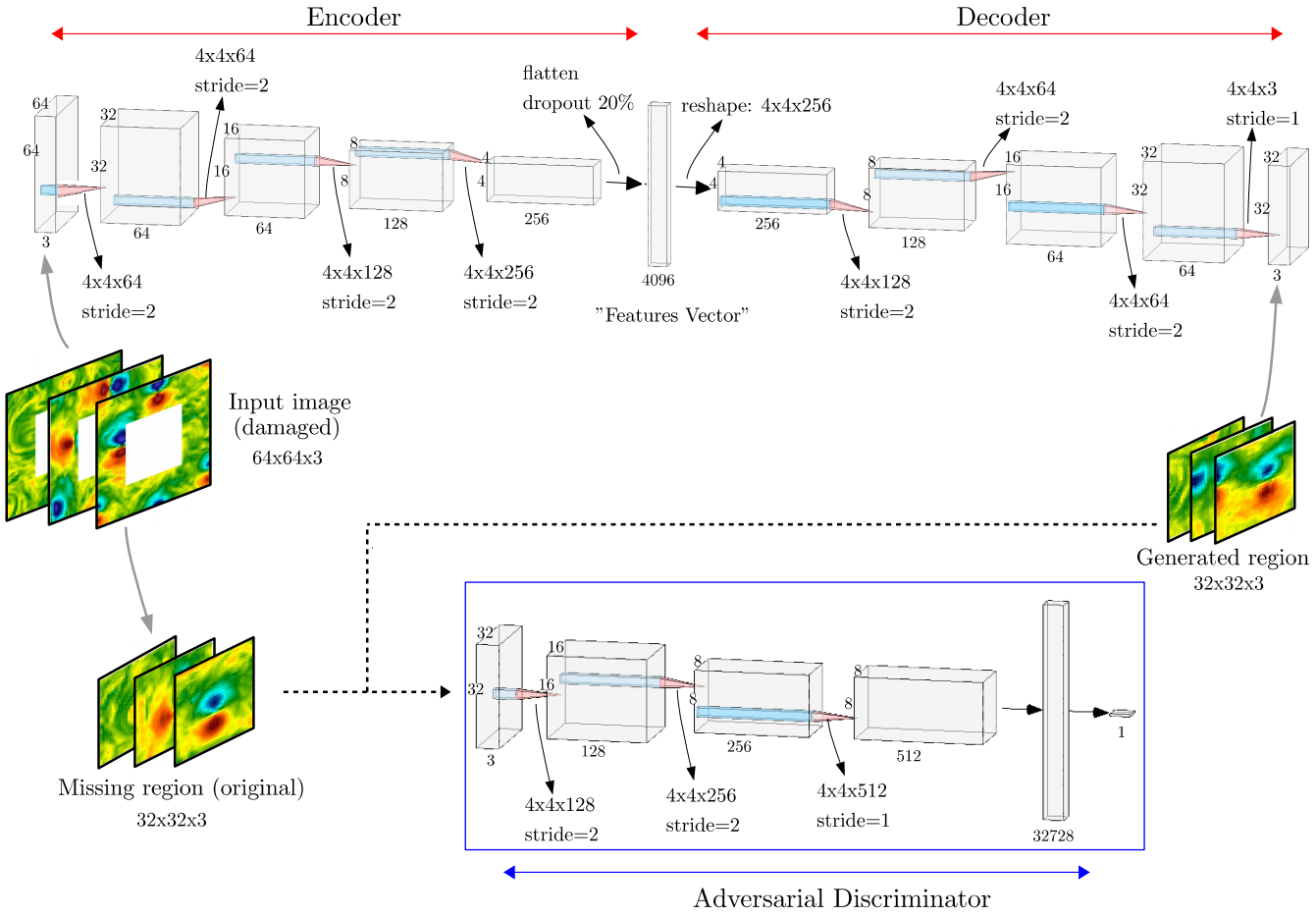

FIG. 15: Sketch of the context encoder, CE1, architecture used for semantic inpainting with joint reconstruction and adversarial loss. Here and in the following figure visualization is obtained using NN-SVG software, [78].

which should prescribe the desired features. It is similar to the decoder part of the CE1, plus the insertion of a fully connected layer from the input to an intermediate layer of size 8192 that then is reshaped as $4 \times 4 \times 512$ volume. The latter is used as the first stage of the next layers: with four successive transposed convolutions of size 5×5 with stride=2 (with ReLu activation) we go from $4 \times 4 \times 512$ to $8 \times 8 \times 512$ to $16 \times 16 \times 256$ to $32 \times 32 \times 128$ to finally obtain, using 3 transposed convolutions, the output velocity field of size $64 \times 64 \times 3$. The Discriminator network, takes a 3 component velocity field of dimension 64×64 and outputs the probability that the input is real. Similar to the analogous in the CE1 network, using convolutional layers of size 5×5 with stride=2 (with LeakyRELU activation) we reduce the dimension of the input down to a 4×4 image with 512 channels. After reshaping to a vector of size 8192, the last stage is made of a fully connected layer and a sigmoid activation function, providing the probability to be real or created by the generator.
The generator component of the network is trained to *minimize* the function:

$$\mathcal{L}_{gen} = \mathbb{E}_{\mathbf{z}}\{\log(1 - D[G[\mathbf{z}]])\} \tag{B1}$$

in order to fool the discriminator with its fake output. The discriminator component of the network is trained to *maximize* the function:

$$\mathcal{L}_{adv} = \mathbb{E}_{\mathcal{I}_v,\mathbf{z}}\{\log(D[\mathcal{I}_v]) + \log(1 - D[G[\mathbf{z}]])\} \tag{B2}$$

in order to assign probability 1 to real fields and probability 0 to generated fields. As with CE1, we have normalized independently the 3 components of the input data to be in the $[-1 : 1]$ range. The Generator output was then re-scaled to match the original range of the velocity fields. Both networks are trained with the ADAM optimizer,

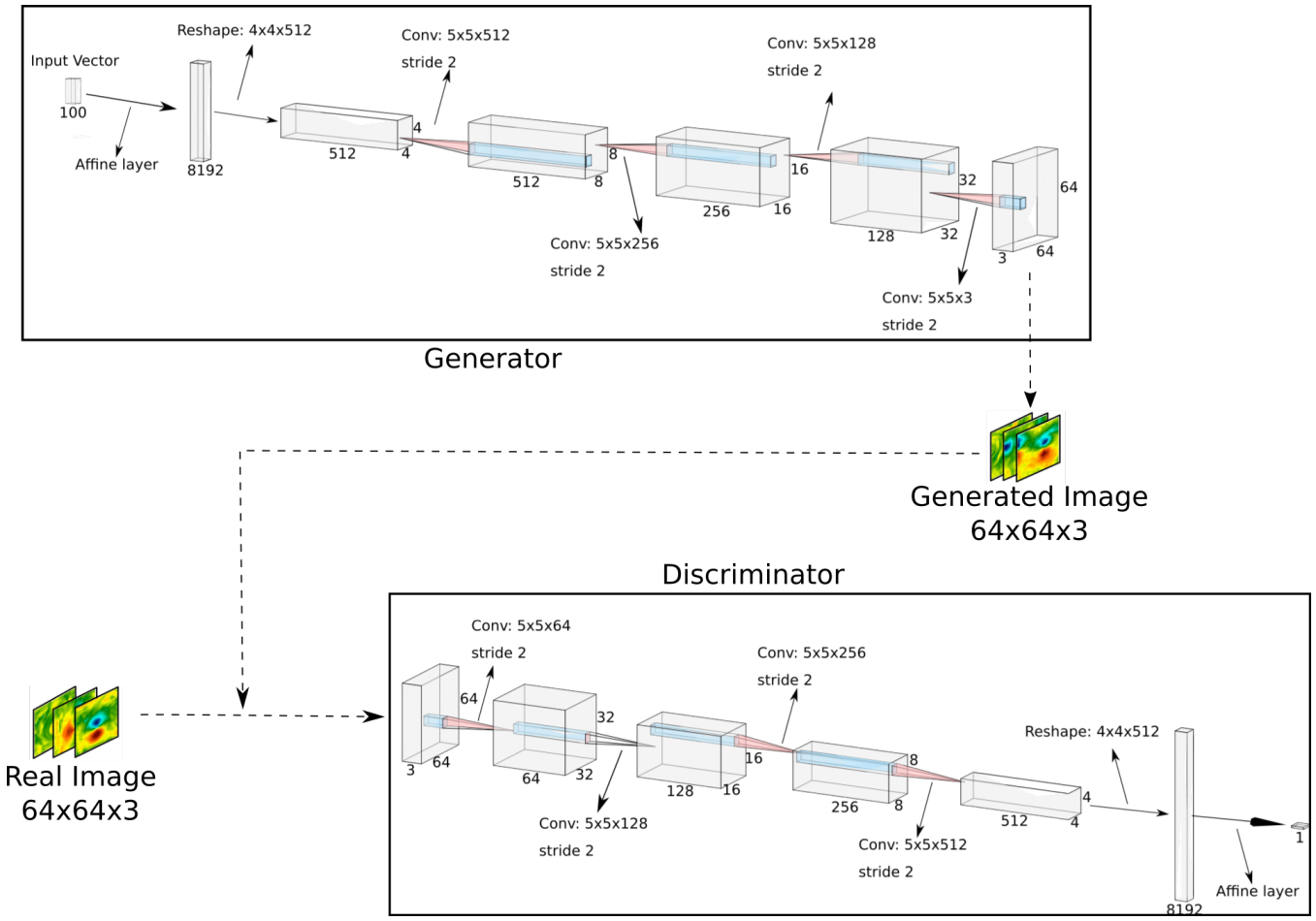

FIG. 16: Structure of the CE2 network: the generator from a 100-dimensional input vector produces a 64x64x3 image, the discriminator decides if an input 64x64x3 image is coming from real data or is generated.

[79], using a learning rate of 2×10^{-4}, $\beta_1 = 0.5$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. The training was performed using a batch size equal to 256, so the 81×1024 planes of the dataset were divided into 324 minibatches for a total of 100 Epochs. After the Deep-Gan is trained, its weights are frozen and used for the reconstruction phase: to recover the optimal $\mathbf{z}^*$-encoding that is closest to each corrupted image, the loss given in eq. 9 is minimized by back propagation to the input data using an ADAM optimization. In our test, we used a set of N=1024 images to be completed (selected outside the train dataset for the GAN), each minimized indipendently (batch size=1), with a learning rate of 2×10^{-2}, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$ for a high number of epochs (10 000).

Appendix C: Database

Data are key for machine learning; hence, our aim to produce and curate benchmark datasets (and software) to foster interest among researchers in related fields. Complex flows and complex fluids benchmarks are more challenging than the *traditional* image datasets found in computer vision and machine learning: our data have high-resolution, multi-scale features, are inherently stochastic, without long term spatial and temporal correlations, and -often- with multi-scale non differentiable properties.

We share the believe that we are entering a new era in fluid mechanics research. Decades (centuries) of great theoretical, numerical and experimental developments based on first principles, brute-force simulations of the equations of motion or accurate observation of nature are now confronting with data-driven tools and analysis.

The database used is SMART-Rot a subset of data deployed in Smart-TURB [21] and it is prepared in four steps:

- The DNS simulations at a resolution of 256^3 grid points are truncated in Fourier space retaining till $k_f \leq 32$, then transformed back in real space downscaling them to 64^3

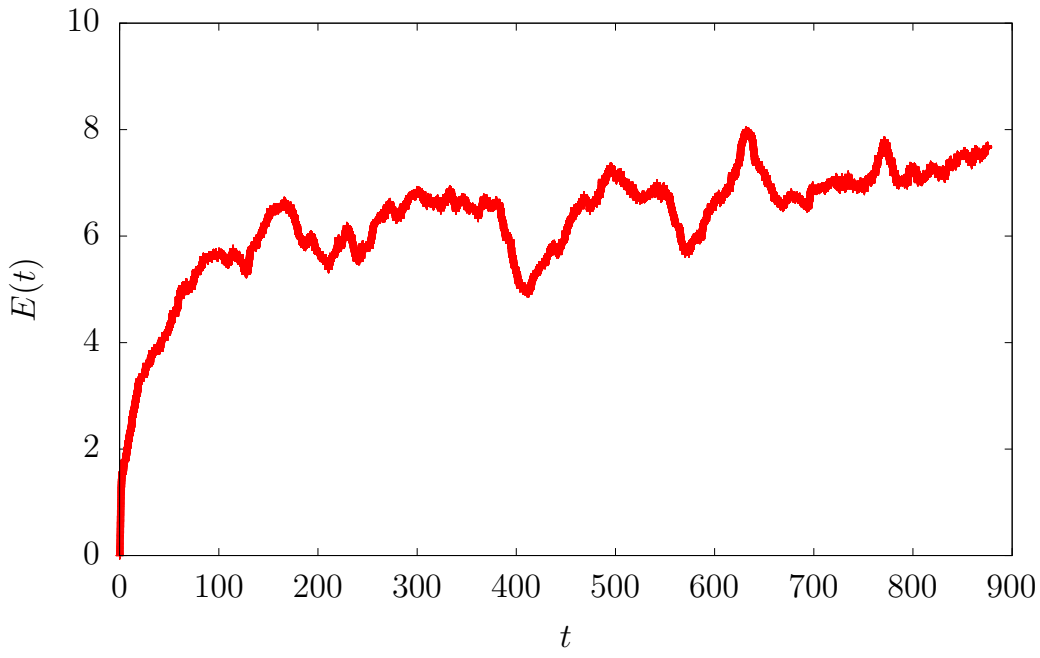

FIG. 17: Energy evolution for the turbulent flow generated during the simulation performed to generate the database of 600 different configuration at a resolution of 256^3 grid points. The velocity fields are extracted from time $t = 276$, to reach the flow thermalization, up to the final time of $t = 876$ every $t = 1$ which corresponds to 2000 simulation time-steps.

- From the whole time evolution, a number of 600 snapshots is used, with large temporal separation to decrease correlations (see Fig. 17).
- For each configuration, 16 horizontal cuts in the plane (x_1, x_2) are selected. To increase the dataset, we shift each plane in 11 different random ways such as to preserve periodic boundary conditions and to obtain a total of $16 \times 11 = 176$ planes
- Finally, the dataset is composed of $600 \times 176 = 105600$ planes, which are organized in a random order to avoid any time-correlation between successive planes.

The database of the configurations was then divided into train set, the first 80×1024 planes, and validation set, the next 20×1024 planes.
The full database TURB-Rot is made public, available for download using the SMART-Turb portal [21]. The portal uses the concept of "Dataset" to aggregate resources related to the same simulation: we have released both the original full resolution of 3d DNS snapshot at 256^3 grid points and the database of 105600 (x_1, x_2)-planes with size 64×64 used by CE1 and CE2, in a dataset named $\mathbf{TURB - Rot}$.

[1] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*, volume 793. John Wiley & Sons, 2019.

[2] Mark Asch, Marc Bocquet, and Maëlle Nodet. *Data assimilation: methods, algorithms, and applications*, volume 11. SIAM, 2016.

[3] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[4] Raymond A Yeh, Chen Chen, Teck Yian Lim, Alexander G Schwing, Mark Hasegawa-Johnson, and Minh N Do. Semantic image inpainting with deep generative models. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5485–5493, 2017.

[5] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9446–9454, 2018.

[6] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[7] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. *arXiv:1508.05326*, 2015.

[8] Qi Wang, Yosuke Hasegawa, and Tamer A. Zaki. Spatial reconstruction of steady scalar sources from remote measurements in turbulent flow. *Journal of Fluid Mechanics*, 870:316–352, July 2019. Publisher: Cambridge University Press.

[9] Vincent Mons, Qi Wang, and Tamer A. Zaki. Kriging-enhanced ensemble variational data assimilation for scalar-source identification in turbulent environments. *Journal of Computational Physics*, 398:108856, December 2019.

[10] Eugenia Kalnay. *Atmospheric Modeling, Data Assimilation and Predictability*. Cambridge University Press, 2003.

[11] Markus Reichstein, Gustau Camps-Valls, Bjorn Stevens, Martin Jung, Joachim Denzler, Nuno Carvalhais, et al. Deep learning and process understanding for data-driven earth system science. *Nature*, 566(7743):195–204, 2019.

[12] Alberto Carrassi, Marc Bocquet, Laurent Bertino, and Geir Evensen. Data assimilation in the geosciences: An overview of methods, issues, and perspectives. *Wiley Interdisciplinary Reviews: Climate Change*, 9(5):e535, 2018.

[13] Peter Bauer, Alan Thorpe, and Gilbert Brunet. The quiet revolution of numerical weather prediction. *Nature*, 525(7567):47–55, 2015.

[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[15] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[17] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.

[18] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[19] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013.

[20] Cristian C. Lalescu, Charles Meneveau, and Gregory L. Eyink. Synchronization of chaos in fully developed turbulence. *Phys. Rev. Lett.*, 110:084102, Feb 2013.

[21] Luca Biferale, Fabio Bonaccorso, Michele Buzzicotti, and Patricio Clark di Leoni. TURB-Rot. A large database of 3d and 2d snapshots from turbulent rotating flows. `http://smart-turb.roma2.infn.it`. *arXiv:2006.07469*, 2020.

[22] Alexandros Alexakis and Luca Biferale. Cascades and transitions in turbulent flows. *Physics Reports*, 767:1–101, 2018.

[23] James E. Hoke and Richard A. Anthes. The initialization of numerical models by a dynamic-initialization technique. *Monthly Weather Review*, 104(12):1551–1556, 1976.

[24] S. Lakshmivarahan and John M. Lewis. Nudging methods: A critical overview. In *Data Assimilation for Atmospheric, Oceanic and Hydrologic Applications (Vol. II)*, pages 27–57. Springer, Berlin, Heidelberg, 2013.

[25] Patricio Clark Di Leoni, Andrea Mazzino, and Luca Biferale. Synchronization to Big Data: Nudging the Navier-Stokes Equations for Data Assimilation of Turbulent Flows. *Physical Review X*, 10(1):011023, February 2020. Publisher: American Physical Society.

[26] Gal Berkooz, Philip Holmes, and John L Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual review of fluid mechanics*, 25(1):539–575, 1993.

[27] Steven L Brunton and J Nathan Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2019.

[28] Thomas Duriez, Steven L Brunton, and Bernd R Noack. *Machine learning control-taming nonlinear dynamics and turbulence*, volume 116. Springer, 2017.

[29] Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52:477–508, 2020.

[30] Kai Fukami, Koji Fukagata, and Kunihiko Taira. Super-resolution reconstruction of turbulent flows with machine learning. *Journal of Fluid Mechanics*, 870:106–120, July 2019.

[31] Kai Fukami, Yusuke Nabae, Ken Kawai, and Koji Fukagata. Synthetic turbulent inflow generator using machine learning. *Physical Review Fluids*, 4(6):064603, June 2019. arXiv: 1806.08903.

[32] Jared L. Callaham, Kazuki Maeda, and Steven L. Brunton. Robust flow reconstruction from limited measurements via sparse representation. *Physical Review Fluids*, 4(10):103907, October 2019.

[33] Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, February 2020. Publisher: American Association for the Advancement of Science Section: Report.

[34] Julien Brajard, Alberto Carrassi, Marc Bocquet, and Laurent Bertino. Combining data assimilation and machine learning to emulate a dynamical model from sparse and noisy observations: a case study with the Lorenz 96 model. *Geoscientific Model Development Discussions*, pages 1–21, May 2019.

[35] Alexander Wikner, Jaideep Pathak, Brian Hunt, Michelle Girvan, Troy Arcomano, Istvan Szunyogh, Andrew Pomerance, and Edward Ott. Combining machine learning with knowledge-based modeling for scalable forecasting and subgrid-scale closure of large, complex, spatiotemporal systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(5):053111, 2020.

[36] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[37] Jin-Long Wu, Karthik Kashinath, Adrian Albert, Dragos Chirila, Heng Xiao, et al. Enforcing statistical constraints in generative adversarial networks for modeling chaotic dynamical systems. *Journal of Computational Physics*, 406:109209, 2020.

[38] Rui Wang, Adrian Albert, Karthik Kashinath, and Mustafa Mustafa. Towards physics-informed deep learning for spatiotemporal modeling of turbulent flows. *AGUFM*, 2019:IN32B–13, 2019.

[39] N Benjamin Erichson, Michael Muehlebach, and Michael W Mahoney. Physics-informed autoencoders for lyapunov-stable fluid flow prediction. *arXiv:1905.10866*, 2019.

[40] Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1):1–10, 2018.

[41] Karthik Kashinath, Philip Marcus, et al. Enforcing physical constraints in cnns through differentiable pde layer. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

[42] Arvind T Mohan, Nicholas Lubbers, Daniel Livescu, and Michael Chertkov. Embedding hard physical constraints in convolutional neural networks for 3d turbulence. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

[43] Arvind Mohan, Don Daniel, Michael Chertkov, and Daniel Livescu. Compressed convolutional lstm: An efficient deep learning framework to model high fidelity 3d turbulence. *arXiv:1903.00033*, 2019.

[44] Steffen Wiewel, Moritz Becher, and Nils Thuerey. Latent space physics: Towards learning the temporal evolution of fluid flow. In *Computer Graphics Forum*, volume 38, pages 71–82. Wiley Online Library, 2019.

[45] Byungsoo Kim, Vinicius C Azevedo, Nils Thuerey, Theodore Kim, Markus Gross, and Barbara Solenthaler. Deep fluids: A generative network for parameterized fluid simulations. In *Computer Graphics Forum*, volume 38, pages 59–70. Wiley Online Library, 2019.

[46] Liu Yang, Sean Treichler, Thorsten Kurth, Keno Fischer, David Barajas-Solano, Josh Romero, Valentin Churavy, Alexandre Tartakovsky, Michael Houston, Prabhat, and George Karniadakis. Highly-scalable, physics-informed GANs for learning solutions of stochastic PDEs. *arXiv:1910.13444 [physics, stat]*, October 2019. arXiv: 1910.13444.

[47] Geir Evensen. *Data Assimilation: The Ensemble Kalman Filter*. Springer Science & Business Media, 2006.

[48] P. L. Houtekamer and Fuqing Zhang. Review of the ensemble kalman filter for atmospheric data assimilation. *Monthly Weather Review*, 144(12):4489–4532, 2016.

[49] Michael L Stein. *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media, 2012.

[50] Hasan Gunes, Sirod Sirisup, and George Em Karniadakis. Gappy data: To krig or not to krig? *Journal of Computational Physics*, 212(1):358–382, 2006.

[51] Daniele Venturi and George Em Karniadakis. Gappy data and reconstruction procedures for flow past a cylinder. *Journal of Fluid Mechanics*, 519:315–336, 2004.

[52] Isabel Scherl, Benjamin Strom, Jessica K Shang, Owen Williams, Brian L Polagye, and Steven L Brunton. Robust principal component analysis for modal decomposition of corrupt fluid flows. *Physical Review Fluids*, 5(5):054401, 2020.

[53] Uriel Frisch. *Turbulence: the legacy of AN Kolmogorov*. Cambridge University Press, 1995.

[54] Peter Alan Davidson. *Turbulence in rotating, stratified and electrically conducting fluids*. Cambridge University Press, 2013.

[55] Antoine Campagne, Basile Gallet, Frédéric Moisy, and Pierre-Philippe Cortet. Direct and inverse energy cascades in a forced rotating turbulence experiment. *Physics of Fluids*, 26(12):125112, 2014.

[56] Annick Pouquet and Raffaele Marino. Geophysical turbulence and the duality of the energy flow across scales. *Physical review letters*, 111(23):234501, 2013.

[57] Ehud Yarom and Eran Sharon. Experimental observation of steady inertial wave turbulence in deep rotating flows. *Nature Physics*, 10(7):510–514, 2014.

[58] Pierre Sagaut and Claude Cambon. *Homogeneous turbulence dynamics*, volume 10. Springer, 2008.

[59] Michele Buzzicotti, Hussein Aluie, Luca Biferale, and Moritz Linkmann. Energy transfer in turbulence under rotation. *Physical Review Fluids*, 3(3):034802, 2018.

[60] Luca Biferale, Fabio Bonaccorso, Irene M Mazzitelli, Michel AT van Hinsberg, Alessandra S Lanotte, Stefano Musacchio,

Prasad Perlekar, and Federico Toschi. Coherent structures and extreme events in rotating multiphase turbulent flows. *Physical Review X*, 6(4):041036, 2016.

[61] Leslie M Smith and Fabian Waleffe. Transfer of energy to two-dimensional large scales in forced, rotating three-dimensional turbulence. *Physics of fluids*, 11(6):1608–1622, 1999.

[62] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 417–424, 2000.

[63] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. In *ACM Transactions on Graphics (ToG)*, volume 28, page 24. ACM, 2009.

[64] Alexei A Efros and Thomas K Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1033–1038. IEEE, 1999.

[65] Stanley Osher, Martin Burger, Donald Goldfarb, Jinjun Xu, and Wotao Yin. An iterative regularization method for total variation-based image restoration. *Multiscale Modeling & Simulation*, 4(2):460–489, 2005.

[66] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[67] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494, 2015.

[68] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv:1312.6034*, 2013.

[69] Shlomo Benartzi, John Beshears, Katherine L Milkman, Cass R Sunstein, Richard H Thaler, Maya Shankar, Will Tucker-Ray, William J Congdon, and Steven Galing. Should governments invest more in nudging? *Psychological science*, 28(8):1041–1055, 2017.

[70] Masakazu Gesho, Eric Olson, and Edriss S. Titi. A Computational Study of a Data Assimilation Algorithm for the Two-dimensional Navier-Stokes Equations. *Communications in Computational Physics*, 19(4):1094–1110, April 2016.

[71] Aseel Farhat, Evelyn Lunasin, and Edriss S. Titi. Abridged Continuous Data Assimilation for the 2d Navier–Stokes Equations Utilizing Measurements of Only One Component of the Velocity Field. *Journal of Mathematical Fluid Mechanics*, 18(1):1–23, March 2016.

[72] A Farhat, NE Glatt-Holtz, VR Martinez, SA McQuarrie, and JP Whitehead. Data assimilation in large prandtl rayleigh–benard convection from thermal measurements. *SIAM Journal on Applied Dynamical Systems*, 19(1):510–540, 2020.

[73] Patricio Clark Di Leoni, Andrea Mazzino, and Luca Biferale. Inferring flow parameters and turbulent configuration with physics-informed data assimilation and spectral nudging. *Physical Review Fluids*, 3(10):104604, 2018.

[74] Suraj Pawar, Shady E Ahmed, Omer San, Adil Rasheed, and Ionel M Navon. Long short-term memory embedded nudging schemes for nonlinear data assimilation of geophysical flows. *arXiv:2005.11296*, 2020.

[75] Marc T Henry de Frahan and Ray W Grout. Data recovery in computational fluid dynamics through deep image priors. *arXiv:1901.11113*, 2019.

[76] Jianhong Shen and Tony F Chan. Mathematical models for local nontexture inpaintings. *SIAM Journal on Applied Mathematics*, 62(3):1019–1043, 2002.

[77] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[78] Alexander LeNail. Nn-svg: Publication-ready neural network architecture schematics. *Journal of Open Source Software*, 4(33):747, 2019.

[79] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[80] Brandon Amos. Image Completion with Deep Learning in TensorFlow. `http://bamos.github.io/2016/08/09/deep-completion`. Accessed: 2019.