Imperial College London Department of Earth Sciences and Engineering
MSc in Applied Computational Science and Engineering

INDEPENDENT RESEARCH PROJECT

Data Assimilation and Predictive Methods with Generative Models Applied to Fluid Flows

Github repo: acse-moj20/msc-irp
Email: mustapha.jolaade20@imperial.ac.uk

Supervisors: Vinicius L. S. Silva, Dr. Claire Heaney, Prof. Christopher Pain, *AMCG, ICL*

September 3, 2021

Abstract

The production of numerous high fidelity simulations in the field of fluid dynamics simulation has been one of challenging underpinning. The computational resources and time required to generate these simulations are far beyond desired expectations. With several successes of generative models, specifically generative adversarial networks (GANs) in image processing, noise reduction, etc., we explore the performance of the powerful generative capabilities of both GANs and Adversarial Auto-encoders to generate, assimilate and represent uncertainties in the modelling of fluid problems. We apply our technique to a test case of 2-dimensional Gaussian vortices governed by the Navier-Stokes equation in time. Our results show that while both GANs and AAE are able to assimilate original fluid data and even predict forward in time, to a reasonable error margin, generating new samples that have never before been seen by the models seems to be unattainable hence restricting the ability to complete confident uncertainties measurement for fluid problems. We recommend a more in-depth study to obtain a conclusive description of generative models applied to turbulent fluid flow.

Keywords: generative adversarial networks; adversarial auto-encoder; 2-dimensional turbulence; spatial-temporal predictions; data assimilation; deep learning.

1 Introduction

Historically, the study of fluid dynamics has involved massive amounts of data generated either from controlled experiments, field measurements or large-scale numerical simulations. The high volume of data, amongst other reasons, means these methods can be relatively slow and require a great deal of computational power to be able to model the underlying physics. While advancements in high performance computing research has boosted speed and accuracy of numerical simulation, obstacles still remain [1, 2, 3]. Thus, the development of computational frameworks that are accurate, robust, cheap and fast enough to model fluid dynamics remains a key aspect of computational science and engineering research. Given the existence of extensive (and expanding) data combined with headway in algorithm development like back-propagation algorithm [4] and principal component analysis (PCA) [5], machine learning (ML) concepts and tools such as deep generative models are receiving great attention among fluid scientists/dynamicists and are increasingly adopted as a major mode of scientific inquiry in the field of fluid modelling [2]. This surge in interest for data-driven approaches that solve fluid problems without the application of first principles has presented new challenges that differ from those tackled in typical applications of ML such as natural-language processing, image generation, etc. (CITE ...). In this research work, generative modelling, a branch of machine learning, is applied to a 2-dimensional turbulent fluid problem for the purposes of rapidly predicting forward in time and assimilating observed data while avoiding the high computational cost of traditional numerical methods.

Generative modelling has garnered a huge amount of interest in recent years [6]. In the field of fluid dynamics, deep generative models have been used in... CITE(). The main idea behind generative modelling is to build a statistical model around a given dataset that is capable of generating new sample instances that appear to be taken from the original dataset. These new samples can further be used for tackling problems related to the dataset. When the building process is based on deep networks (artificial neural networks such as CNN [7] or Recurrent Neural Network [4]) that use multiple layers to capture how patterns/ features of the dataset are organised or clustered, the resulting model is termed a deep generative model. Once a deep generative model has learned the structure of the training dataset, by being fed a random vector as input, its networks can generate desired samples from complex probability distributions in high-dimensional spaces [8]. Broadly speaking, a deep generative network can be trained using two main methods. The first is to use a Variational Auto-encoder (VAE) that uses stochastic variational inference to minimize the lower bound of the data likelihood [6]. The second involves the use of a Generative Adversarial Network (GAN) whereby two (neural network) players play a zero-sum game. The game seeks to minimize the distribution divergence between the model output and the real dataset by using real samples as a proxy for optimization. A novel third method born out of the amalgamation of these two methods is the use of

Adversarial Auto-encoder (AAE) [9]. In this research work, attention is given to both GAN and AAE as data-driven methods for prediction and modelling of spatial-temporal turbulent fluid flow.

Although non-intrusive modeling methods have been used for time-dependent turbulent fluid modeling in areas such as urban modelling [10], subsurface flow [11] and aerospace [12] including specifically for the Navier-stokes equation [13], this project is unique in that it takes advantage of adjoint-like properties of generative models [14] provided by Tensorflow's [15] automatic differentiation's interface to carry out efficient data assimilation of 2D turbulent fluid model.

The rest of this paper is structured as follows: the next section (Section 2) provides a description of the methodology adopted from [14] for spatial-temporal prediction and data assimilation with GANs. Here, we also discuss the order prediction procedure and the general structure of General Adversarial Network and the Adversarial Auto-encoder built for prediction and sample generation. Section 3 details the code structure implemented in running experiments and the software requirements needed to reproduce results. The subsequent section, Section 4, introduces the test fluid system and the relevant properties of the test data used for this work. Relevant discussion about the transformation carried out to make the data suitable for use are also detailed in this section. The obtained results from prediction and data assimilation are presented in section 5 while discussion about the results and other observations are presented in Section 6. Finally, conclusion and remarks about possible future work are then provided in section 6.

2 Methodology

The use of GANs for time series prediction, data assimilation (DA) and uncertainty quantification (UQ) of real world dynamical system has been proven to be successful for the spatial-temporal spread of COVID-19 using SEIRS type models [14]. Particularly, the method in [14] has been shown to be independent of the underlying system, thus this project will apply the same method for the 2D turbulent fluid model.

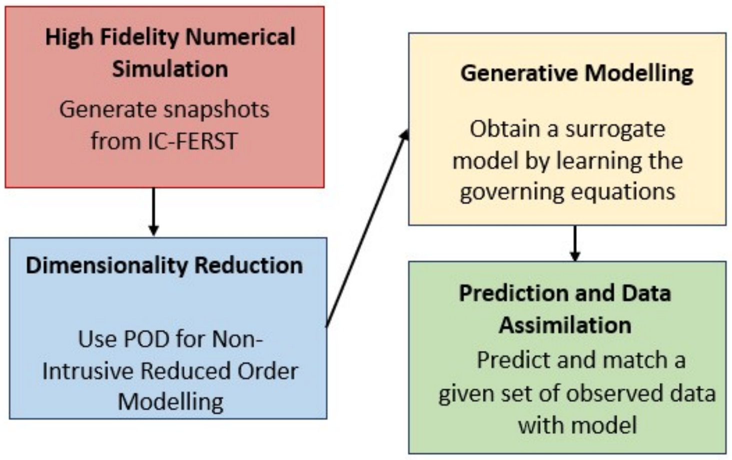

High Fidelity Numerical Simulation
Generate snapshots from IC-FERST
Dimensionality Reduction
Use POD for Non-Intrusive Reduced Order Modelling
Generative Modelling
Obtain a surrogate model by learning the governing equations
Prediction and Data Assimilation
Predict and match a given set of observed data with model

Figure 1: Methodology workflow.

This method commences with the reduced order modelling (ROM) of the fluid domain from a high-fidelity spatial domain to a lower dimensional representation. Then, a generative model capable of forward modelling and/or sample generation is built and trained to learn a mapping between input latent vector and the lower dimensional representation. Finally, the processes of DA and UQ that follow involve extending the power of the generative model to include assimilation of a given input of observed sensor data. The aim is then for the generative model to serve as a surrogate model that can match the given data without supplementary model training or any additional simulations of the high-fidelity model (HFM). More details about the methodology of each step is given below.

2.1 POD-based Non Intrusive Reduced Order Modelling:

The connection between physics-based machine learning and dimensionality reduction has been substantially studied and well-documented [16]. Results of these studies have shown that many methods used to obtain a low-dimensional subspace of a system are related to machine learning methods. To be able to generate forward predictions of the fluid system in this study, our proposed methodology involves the reduction of number of variables to a size the GAN/AAE can cheaply work with. In modern computational research, Reduced Order Modelling (ROM) is a well-known technique capable of achieving this [11]. By constructing reduced-order models that encapsulate the original features of the fluid systems while maintaining its underlying physics, it is possible to seek solutions to a model in a much less expensive way [17, 18], resulting in a considerable increase in computational efficiency [19]. A type of ROM is Non-Intrusive Reduced Order Modelling (NIROM) which is so named due to its non-dependent on the system under study. This model reduction approach uses proper orthogonal decomposition (POD) [20, 21] to derive a physics-inspired low-dimensional parameterization that represents the high dimension of the high-fidelity spatial domain of the fluid model (i.e. state of snapshots). POD is closely related to the method of PCA in statistics and was first used for turbulent flows by [22].

Thus, the dimensionality reduction aspect of our methodology is set within a NIROM framework that involved computing the POD basis vectors (via principal component analysis) using the snapshots of the input training data [20].

Consider a 3-dimensional field $\boldsymbol{\omega}$, which is dependent on some input parameter and varies in space, and time. We can define its function as $\boldsymbol{\omega} : \mathcal{X} \times \mathcal{T} \times \zeta \rightarrow \mathbb{R}$ where $\mathcal{X}$ is the spatial domain, $\mathcal{T}$ is the time domain, and an input domain ζ of initial parameters/condition. The aim of data-driven/non-intrusive dimensionality reduction is to find an approximate model for $\boldsymbol{\omega}$ from the data

$\mathcal{D} \subset \{\omega(\boldsymbol{x}, t, \boldsymbol{z}) \mid \boldsymbol{x} \in \mathcal{X}, t \in \mathcal{T}, \boldsymbol{z} \in \zeta\}$

which, in this case, are snapshots in time of the field. The desired approximate model of the field can be expressed as a linear expansion in the POD basis. This POD basis would be computed from many snapshots data developed as solutions of a high-fidelity model that describes the field. To compute the POD basis, we consider a snapshot data to be $\boldsymbol{\omega}(t; \boldsymbol{z}) \in \mathbb{R}^{n_x}$ where n_x is the dimension of the spatial domain (from finite discretization). Thus, the set $\{\boldsymbol{\omega}(t_i; \boldsymbol{z}_j) \mid i = 1, \cdots, n_t;\ j = 1, \cdots, n_z\}$ of snapshots at n_t different time levels/steps of $t_1,\ t_2,\ \cdots,\ t_{n_t} \in \mathcal{T}$ and n_z different initial input conditions of $\boldsymbol{z_1}, \boldsymbol{z_2}, \cdots, \boldsymbol{z_{n_z}} \in \zeta$ comprises of $n_s = n_t n_z$ snapshots. The snapshot matrix can be defined as $\boldsymbol{S} \in \mathbb{R}^{n_x \times n_s}$ with each row corresponding to a spatial location and each column representing a snapshot in the set. At this stage, PCA can then be introduced for dimensionality reduction. PCA seeks a transformation $\boldsymbol{T}$ that maps each vector $\{\boldsymbol{\omega}(t_i; \boldsymbol{z}_j)$ in $\boldsymbol{S}$ (i.e each snapshot) from the original dimensional space of n_x to a new space that only keeps the first r principal components using the first r eigenvectors of the transformed matrix [5]. We can express the transformation to obtain the principal components decomposition of $\boldsymbol{S}$ as

$$\boldsymbol{T_r} = \boldsymbol{S}\boldsymbol{W_r} \tag{1}$$

where $\boldsymbol{W_r}$ is a $n_s \times r$ weight matrix with columns equal to the eigenvectors of $\boldsymbol{S^T}\boldsymbol{S}$. This transformation produces the matrix $\boldsymbol{T_r}$ that has only r columns where $r << n_x$ and becomes the dimension of our POD basis resulting to a low-dimensional representation of the snapshots that is more computationally efficient. In highlighting the similarity of this method to a Singular Value Decomposition [23], we note that

$$\boldsymbol{T} = \boldsymbol{V}\boldsymbol{\Sigma} \tag{2}$$

where $\boldsymbol{V} \in \mathbb{R}^{n_x \times n_s}$ are the left singular vectors of $\boldsymbol{S}$ and $\boldsymbol{\Sigma} = diag(\sigma_i, \cdots, \sigma_{n_s})$ is the diagonal matrix of the singular values $\sigma_1 \geq \sigma_2 \cdots, \geq \sigma_{n_s} \geq 0$ of $\boldsymbol{S}$. The POD basis/principal components, $\boldsymbol{T_r}$ is simply the r largest singular values multiplied by their corresponding r left singular vectors of the snapshots. The idea is to maximize the variance of the original data while minimising the total least squared errors in the representation of the snapshots. The size of the POD basis/principal component r is chosen by specifying a tolerance in this error calculation. This user-specified tolerance, k also indicates how much information/energy of the data is captured by the resulting snapshot representation. We chose r such that:

$$\frac{\sum\limits_{k=1}^{r} \sigma_k^2}{\sum\limits_{k=1}^{n_s} \sigma_k^2} > k, k = 0.999 \tag{3}$$

A field learned from snapshots transformed into POD basis can be approximated using the same POD basis:

$$\boldsymbol{\omega}(t; \boldsymbol{z}) = \sum_{k=1}^{r} D_k \theta_k(t; \boldsymbol{z}), \tag{4}$$

where $\theta_k(t; \boldsymbol{z})$ is the POD expansion coefficients and D_r is the POD basis of dimension r, we can obtain an approximation of the field. This means given a snapshot filed we can compute its original state using the POD coefficients. Hence, once the dimension reduction is completed, the POD expansion coefficients $\boldsymbol{\theta_{k=1,\cdots,r}(t;\ z)}$ denote the model approximation and parameterization of a snapshot field $\boldsymbol{\omega}(t;\ \boldsymbol{z})$ at time t and input conditions $\boldsymbol{z}$. The coefficients are then employed in the training of generative models for prediction and data assimilation, which have both been proven to be successful in research areas that involve turbulent fluid flow [10] and appearance-based recognition [24]. Results of this aspect of the research work are shown and discussed in section 4.2. Other reduction methods like RBF-based NIROM [25] or POD-Galerkin method [26] which have been shown to be successful with turbulent flows may also be utilized.

2.2 Generative Models

Generative modelling is the process of training a machine learning model with a specific data to produce 'fake' data that mimics the probability distribution of the original training data. In this research study, we produce two generative models to perform time series prediction and data assimilation in the presence of observed sensor data. The two models utilized are: a Generative Adversarial Network (GAN)[27] and an Adversarial Auto-encoder (AAE)[9]. The choice of these models was based on proven success in the use of nonlinear fluid modelling. In the result section, a comparison is between the two models is presented.

2.2.1 Generative Adversarial Network:

A GAN is an artificial learning technology that is composed of two neural networks as shown in figure (REF GAN figure). The first network is a generator module (G) that generates fake versions of the data following training cycles using original data. The generator module does this by mapping a latent vector **z**, which is typically a randomly generated uniform noise to a desired output dataset $\boldsymbol{\theta}$ that resembles the distribution of the real dataset used for its training $\boldsymbol{\theta_r}$ (in this case, the POD expansion coefficients). The end goal of G is produce samples that are identical to the data distribution. The second network, a discriminator module (D), is used for distinguishing between a generated dataset and the original dataset. GANs have been adopted widely in several research areas, showing huge successes in practical applications including simulating fluid models [28]. The training process is essentially a game between these models competing as adversaries. While G generates fake samples from an input random distribution, D tries to distinguish between real samples drawn from the original distribution and the sample output from the generator. D does this by estimating a score which serves as the probability that a particular sample came from the original distribution i.e. $D(G(\boldsymbol{\theta_r})) = 1$. The training process of a GAN is a minimization-maximization problem that is based on a cross-entropy loss function $\boldsymbol{J}(D, G)$:

$$\min_G \max_D E_{\theta_r \sim p_{data}(\theta_r)}[logD(\theta_r)] + E_{\mathbf{z} \sim p_z(\mathbf{z})}[log(1 - D(G(\theta)))]$$

where $p_{data}(\theta_r)$ is the probability data distribution of the target output of real samples θ_r and $p_z(z)$ is the prior distribution for the random latent vector **z**. The training process involves:

- Updating D with gradients that maximize the discriminator function by differentiating with respect to parameters of the discriminator.
- Updating G with gradients that minimize the generator function by differentiating with respect to parameters of the generator.

Mode Collapse: A common problem in the use of GANs for sample generation is mode collapse. Typically, a GAN is trained to produce a wide variety of outputs that mimic the training data distribution. For example, if a GAN is trained with pictures of different dog breeds, we want a different dog for every random input to the dog generator. However, it is possible that the generator only produces a small set of realistic outputs and learns to generate only that seemingly credible output (or small set of outputs) to the discriminator. This happens because the 'lazy' dog generator over-optimizes for a small set of fake breeds indistinguishable by the discriminator, traps it in a local minimum of its loss function (i.e gradient update vanishes) and precludes it to learn to reject that small set of generated outputs when it is repeatedly generated. This kind of failure mode is called mode collapse.

The Wasserstein GAN (WGAN) [29] is a type of GAN that avoids this problem of mode collapse by circumventing the issue of vanishing gradients. This implies that the discriminator is trained to optimality, learning to reject any output/set of outputs the generator tries to stabilize on. The WGAN method introduces a new loss function that alternatively minimizes an approximation of the Earth Mover distance between the distributions completely avoiding mode collapse.

In developing a GAN for the generative modelling of this work, a typical Deep Convolutional GAN (DCGAN) was developed and trained to produce the target output. Following evidence of mode collapse (see Results SECTION) however, an Improved WGAN [30] was also developed by altering the loss functions of the original DCGAN. The WGAN also included a gradient penalty term that led to more diverse output from the generator. The architecture of the GAN models (DCGAN & Improved WGAN) developed is shown in Fig.2.

Wasserstein GAN:

The WGAN loss function uses a Earth mover distance criteria to enforce match of a prior data distribution. The loss function for this type of GAN is given as follows where the second term is a gradient penalty that replaces weight clipping to achieve Lipschitz continuity (gradient with norm at most 1 everywhere). The discriminator in this GAN works as a critic and it is same as the discriminator architecture shown above.

$$L = \mathbb{E}_{\hat{\mathbf{x}} \sim \mathbb{P}_g}\left[D\left(\tilde{\mathbf{x}}\right)\right] - \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_r}\left[D\left(\mathbf{x}\right)\right] + \lambda \mathbb{E}_{\hat{\mathbf{x}} \sim \mathbb{P}_{\hat{\mathbf{x}}}}\left[\left(||\nabla_{\tilde{\mathbf{x}}} D\left(\tilde{\mathbf{x}}\right)||_2 - 1\right)^2\right]$$

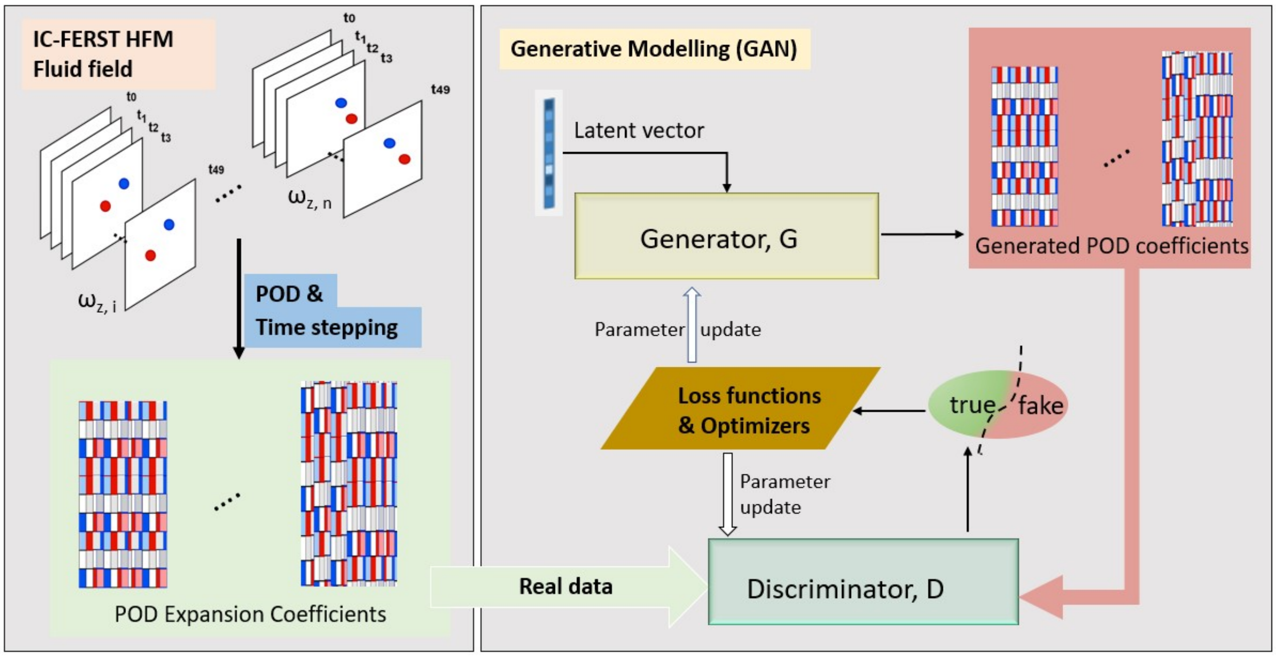


Figure 2: Generative modelling using GAN. In this workflow, real samples obtained from POD-based NIROM are utilized as training data for the discriminator module of a GAN. Fake data produced by the generator, G from an input latent vector is simultaneously used in the training process, with loss back propagated through both neural network modules. Details on the architecture and training parameters of the network can be found in the appendix.

The generative model (GAN and/or WGAN) developed and trained using the above workflow and design can be used for time-series/forward prediction without any changes to its structure. This is also the case when the model is utilized for the assimilation of given observation/sensor data.

2.2.2 Adversarial Autoencoder

A second type of generative model built and implemented in this work is an adversarial autoencoder (AAE). Similar to a GAN, the adversarial auto-encoder was proposed to to as a generative model that seeks to match an aggregated posterior distribution of its hidden latent vector with a prior distribution. To be able to function as a deep generative model, the AAE is trained to perform variational inference that enables its decoder to learn a statistical model that maps between the imposed prior and the data distribution. The AAE has a wide range of applications including semi-supervised classification, unsupervised clustering and data visualization [9]. In the field of computational fluid dynamics, Cheng et. al [31] studied the capability of an advanced deep-AAE for parameterizing nonlinear fluid flow and utilized it in the prediction of a water collapse test case. Here, we develop an AAE and test it for prediction and data assimilation of nonlinear turbulent flow.

An AAE is essentially an integration of a Varitional Autoencoder (VAE) [6] and a GAN. On one level of the AAE is a standard autoencoder that uses a neural network to learn the reconstruction of an input image from a hidden latent code. The other level is a separate network that is very similar to the discriminator module, D of a standard GAN. This level is trained to predict whether a sample data (snapshot) is generated from the autoencoder or a user-specified distribution Due to this combination, the AAE is trained with a dual objective - the adversarial training criterion and the minimisation of a reconstruction error.

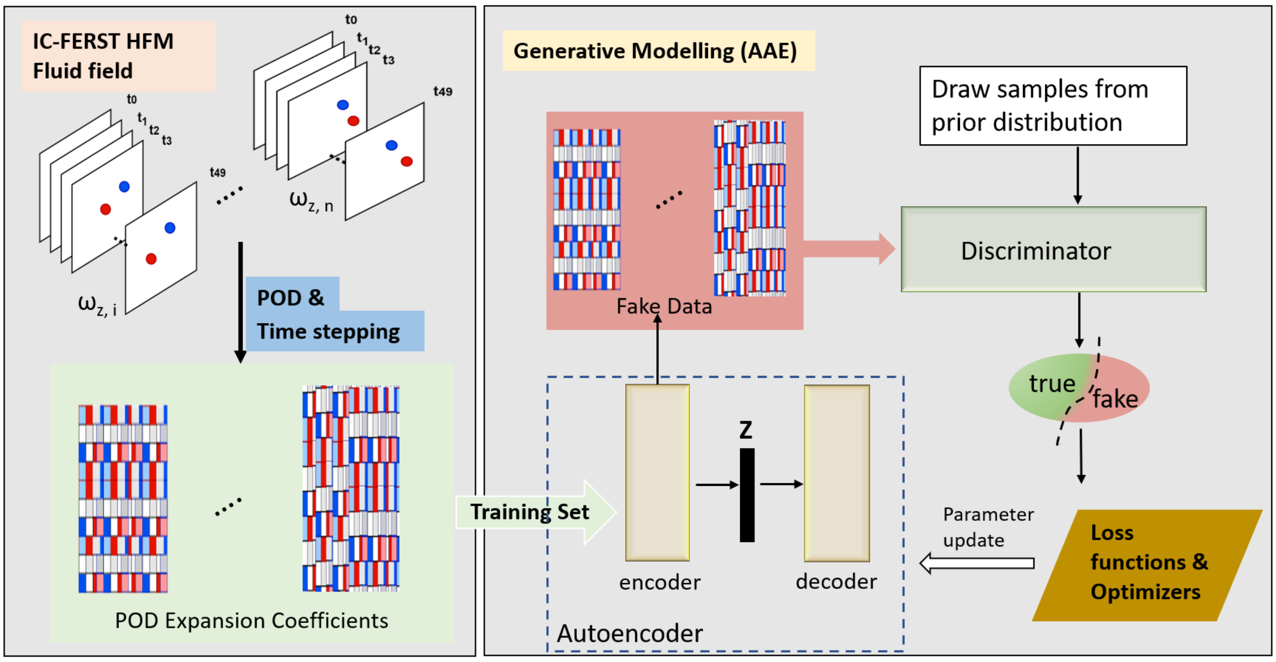


Figure 3: Generative modelling using Adversarial Autoencoder. The prediction ability of the AAE lies in its encoder. The training process of this workflow attempts to match output of the autoencoder with a prior distribution. The discriminator is simply a network of fully-connected layers trying to critic against samples generated from the autoencoder. See the Appendix for details on model architecture and training parameters.

Unlike for the GAN, this workflow requires a crucial modification when utilizing it for data assimilation. Since the POD expansion coefficients of a given sensor data can not be computed using the POD basis of the original snapshots, the data is included in its original form. Thus, the autoencoder is trained with additional columns corresponding to the potential number of sensor points. When assimilating data, the observed sensor data fill the positions of these columns. See the Appendix for the architecture of the Autoencoder.

2.3 Space-time predictions using generative models

The goal of this independent research is to show that generative models such as GANs and AAEs can be utilized for the time-series prediction and data assimilation of nonlinear turbulent fluid models. This section discusses the proposed methodology and an algorithm for its implementation. In comparison with existing applications of GANs for time series prediction, the method used for forward prediction in this work is unique because of its use of automatic differentiation to optimize a newly proposed loss function [14]. This newly proposed methodology is further tested on a 2-Dimensional turbulent fluid model to obtain a surrogate model that is accurate and computationally cheap. The next subsections discuss the proposed method and its components.

2.3.1 Prediction using GANs

The ability of a GAN to produce realistic samples that seem to belong to a prior distribution is leveraged in this project to predict forward in time. A GAN trained with P consecutive time levels of compressed variables/POD coefficients is used for this process. The prediction process takes the first $P-1$ time levels as input to the GAN and the time level P or greater than P (in the case of multi-step prediction is generated as output. The steps are as follows:

1. Latent vector of dimension $P-1$ is randomly generated to serves as input to the GAN. This

commences the prediction at time level $P-1$.

2. The iteration for time is set at $P-1$.
3. The optimisation counter is set to 0.
4. The generator is used to map the randomly generated latent variable to a fake sample with time level up til $P-1$.
5. the difference between this prediction and the known values (up till $P-1$) is computed using the PredictionGAN loss function $\mathcal{L}_p$ in [14]. We want to minimize this loss.
6. The gradient of this loss with respect to the latent variables is then computed and back-propagated through the optimizer while a single increment is added to the optimizer counter.
7. Steps 4 to 6 are repeated to achieve convergence.
8. The resulting latent vector is then used as input to the generator while the predicted time is added to the solution series of time levels.
9. For multiple time level prediction, the iteration of time is increased from $P-1$ to P and step 3 to step 8 are repeated till the final time level.

2.3.2 Prediction using Adversarial Autoencoder

To predict with an Autoencoder, the following steps were followed:

1. Since the autoencoder does not require a latent variable as input, we use the first $P-1$ time levels as input.
2. To predict the time level P, the last $P-1$ time levels is used as an initial guess, and passed into the autoencoder to give a prediction. The aim is for the output of the autoencoder to attempt to match the true snapshot at time P given its training.
3. The output prediction is now used as input guess for the time level P and the time series is passed through the autoencoder till convergence is reached.
4. The final output is the snapshot prediction at time P.
5. For multiple time level predictions, the process is repeated from Step using the last P time levels. Note that this includes the new prediction at time P.

2.4 Data assimilation through time

In addition to spatial-temporal modeling of fluid models, the ability to assimilate observed data into fluid models is highly desirable as it provides practical predictions with narrower uncertainty ranges [32, 33, 34]. Data assimilation is as an inverse problem that allows for the calibration of uncertain model parameters in order to generate results that match observed data within some tolerance [34, 35]. However, data assimilation can be computationally intractable especially for complex models in high-dimensional spaces [33]. In this work we used a GAN generative model to perform data assimilation on a 2D fluid data set. The methodlogy applied here is adopted from [14]. Results is shown in section 4.4.

3 Code Metadata

Main software packages/libraries and data storage facilities the project will utilize over the course of its development include:

- Tensor Flow (Apache License 2.0).
- Scikit-learn (version 0.20 or higher).
- Visualization packages (Matplotlib, Seaborn, etc.)
- Scipy and numpy

The machine learning models that serve as the backbone of this project will be built on Google Colab platform while making use of its 12GB Nvidia Tesla K80 GPU for processing. The high fidelity simulations of fluid systems that serve as training data set will be stored as array files on a personal google drive. To obtain a full list of packages, users can refer to the environment.yml file in the project repository.

The code was developed on Jupyter Notebook. Three notebooks are provided for this research work. 'PredGAN.ipynb', 'PredAAE.ipynb' for prediction using WGAN-GP and AAE respectively. A third notebook 'DA_GAN.ipynb' for data assimilation is also provided. The notebooks for prediction follow the workflow shown in the methodologies above 2 and 3. The following are details of these prediction notebooks and the functions/workflow they contain:

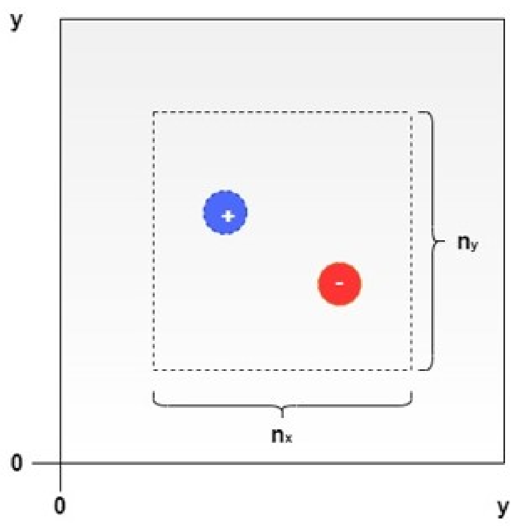
1. Utility Functions:
   (a) Error computation: *calculate_error*.
   (b) Visualization methods: *plot_vortex, plot_linevortex, plot_obs_data*
   (c) Pre and Post Processing of data: *concat_timesteps, transform_data, invert_data*.
2. Dimensionality reduction workflow:
   This uses the PCA method of the Sklearn package.
3. WGAN Training. Training methods include:
   (a) Build (or Load) models.
   (b) Training methods: *discriminator and generator loss methods, optimization and backpropagation methods, history logging methods.*
4. Prediction with trained model:
   This included generating samples and predicting both single and multiple time levels. The prediction optimization algorithm is structured as follows:
   (a) Class *Pred_optimizer():*
   (b) class methods - *opt_step, a call method, compute_loss*
   (c) Functions *pred_time_lvl* and gen_pred_time_lvl.

4 Implementation and Results

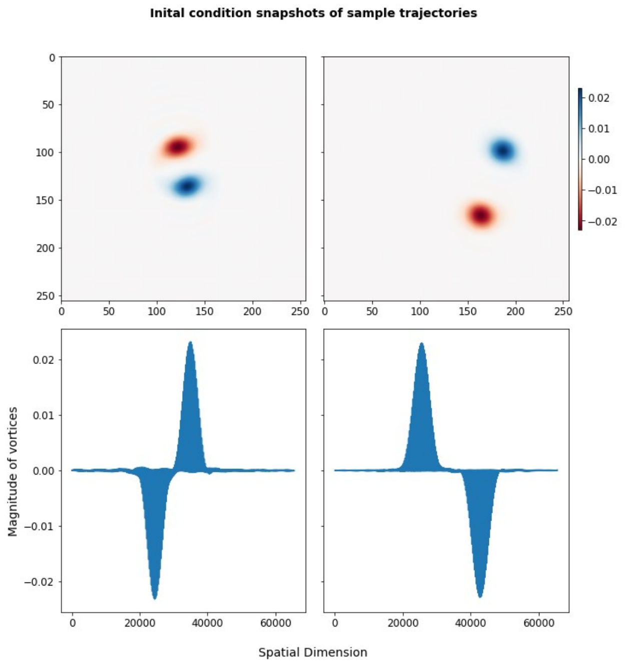
4.1 Study case: Parameter-varying flow in a periodic box

In order to train a GAN capable of time-series prediction, DA and UQ of the 2D turbulent fluid problem, a training dataset is required. This dataset must represent unconditional simulations from high-fidelity model of the problem. By collaborating with a different research group, a raw dataset of x velocity component, y velocity component and pressure measurements of 2-dimensional incompressible Navier-Stokes simulations has been obtained. This particular dataset has been chosen because it represents a parameter-varying flow in a fixed-wall box.

Given that the convolutional layers of a neural network are designed to detect object/features anywhere in an image, this data set is proposed since the large-parameter variations implicit in its generation is of a similar nature as object randomly located in an image [36]. The first step in the project is to transform the velocity dataset into vorticity data so it represents initial Gaussian (randomly initialized) vortices that decay due to viscosity changes. Following this transformation, the vorticity data are then compressed by carrying out a Proper Orthogonal Decomposition (POD).



(a) Spatial domain of snapshots.



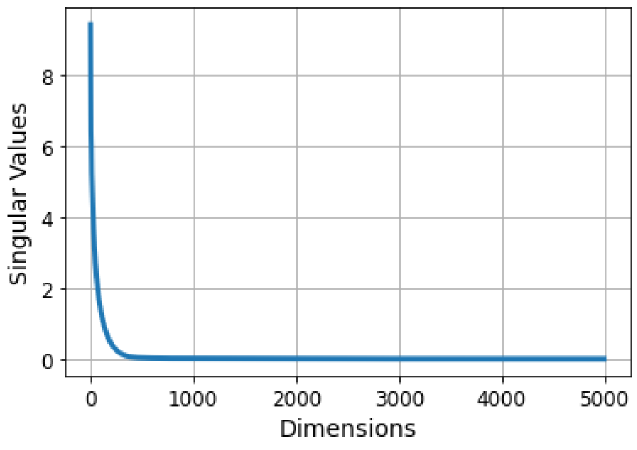
(b) Sample initialized trajectories.

Figure 4: Two-dimensional Gaussian vortices in a square domain. The positive and negative vortices are of equal strength and each snapshot $S \in \mathbb{R}^{y \times y}$ where y = 256, are randomly initialized within a predefined subdomain $n_x \times n_y$. The images on the bottom right show the magnitude of the vortices projected over a 1-D domain.

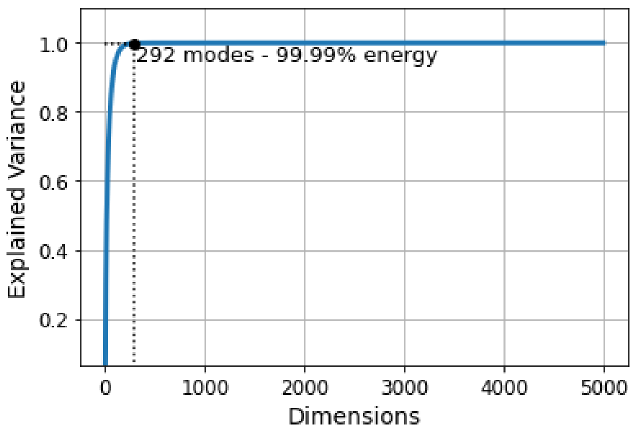
The simulations were run on Imperial College-Finite Element Reservoir Simulator (IC-FERST) using the following criteria:

- Turbulent flow with Re=5000
- Constant viscosity
- No slip walls boundary conditions.
- Spatial domain is a 256 by 256 grid.
- Time domain of 100s discretised using a time-step of 0.01s
- Data snapshots are collected every 2 s (i.e. 50 snapshots for every sample).

4.2 POD compression and order reduction



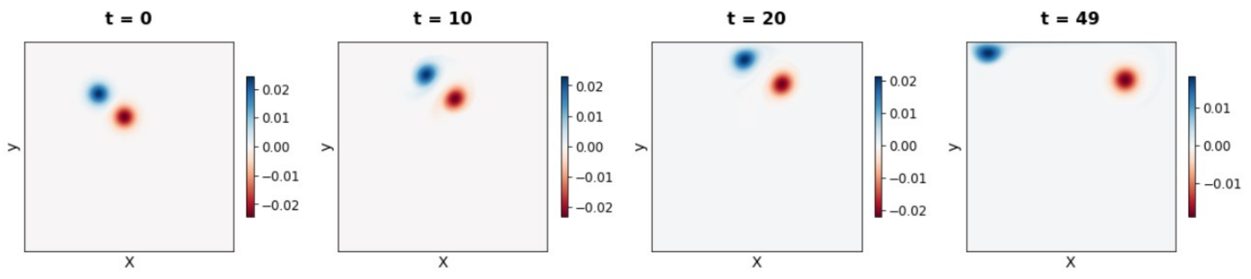
(a) Singular Values.



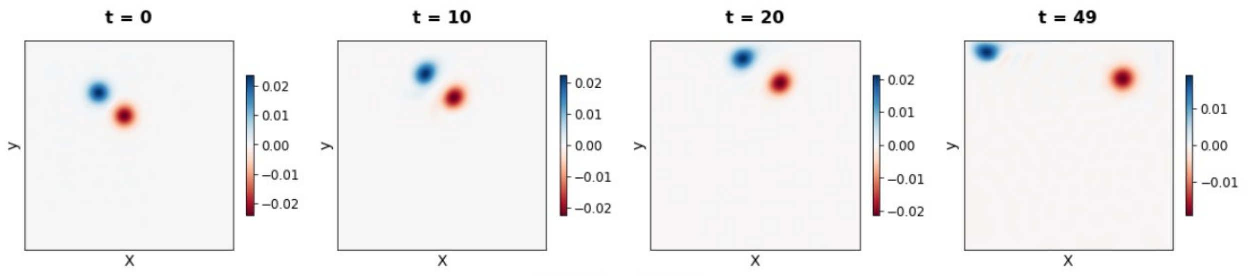
(b) Cumulative energy.

Figure 5: POD singular values and relative cumulative energy for the 2-D Gaussian vorticity filed snapshot set

Each snapshot is no longer a 256 by 256 array but now represented using 292 features (POD coefficients). The cumulative information/energy retained measured using explained variance is over 99.99%. A visual comparison of the compression is shown in Fig. **??** below.



(a) Original snapshots.

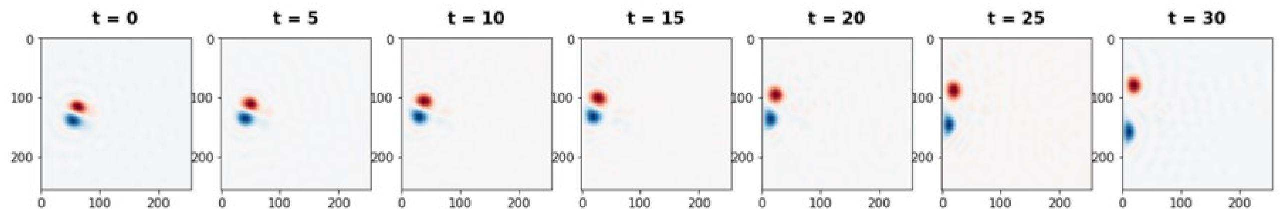


(b) ROM Recovered snapshots.

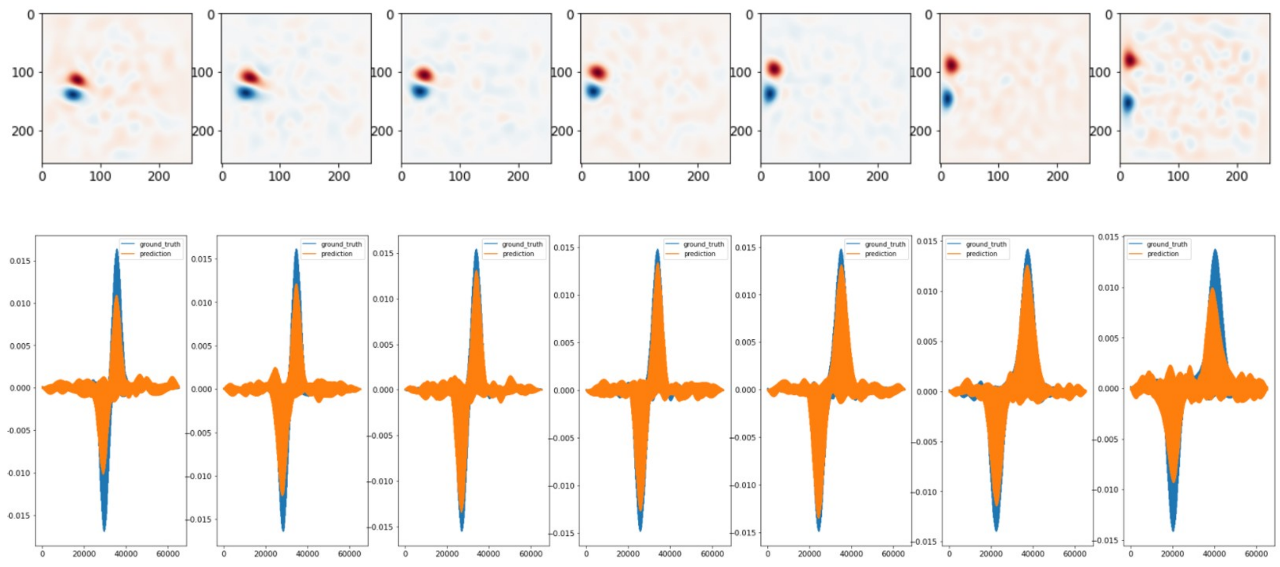
Figure 6: Original and recovered snapshots following POD-based NIROM. The reduction was specified to retain 99.99% information from the original snapshots. The reduction decreased the dimension from 256-by-256 to 292 POD coefficients.

4.3 Prediction using GAN forward model

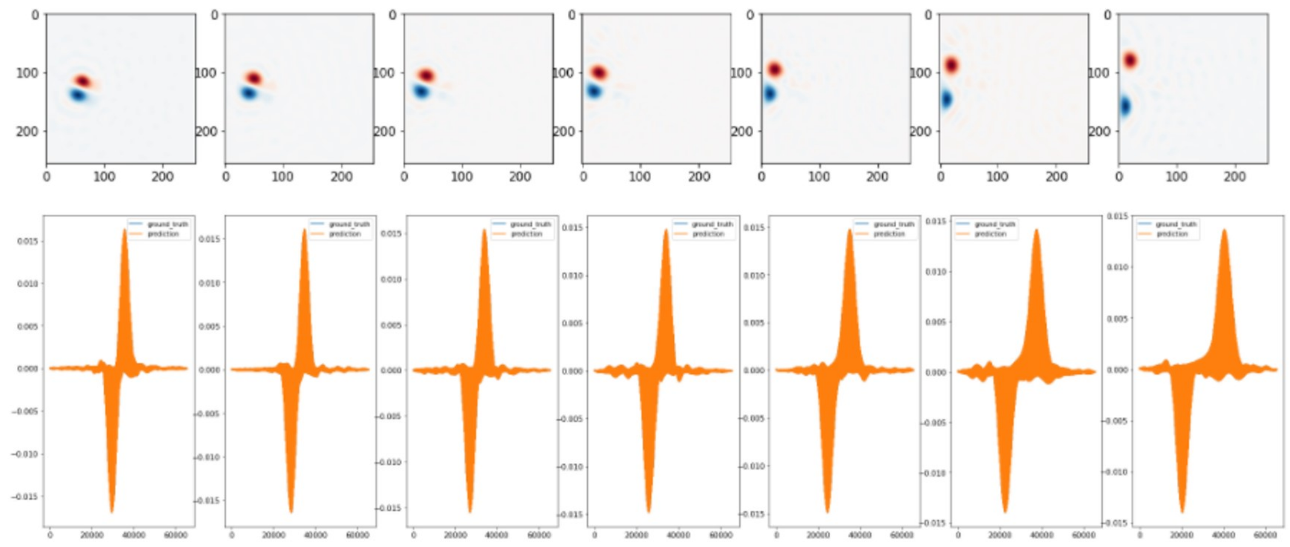
One time level prediction:



(a) original snapshots of trajectory (test data)



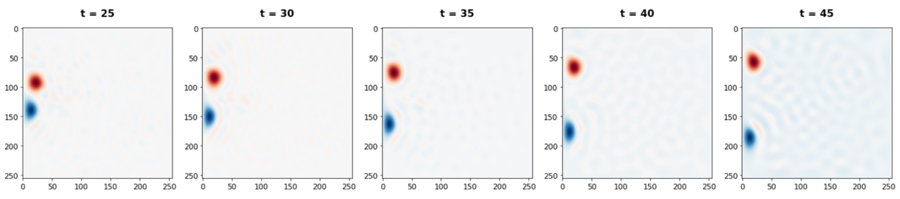
(b) WGAN-GP Prediction of trajectory.



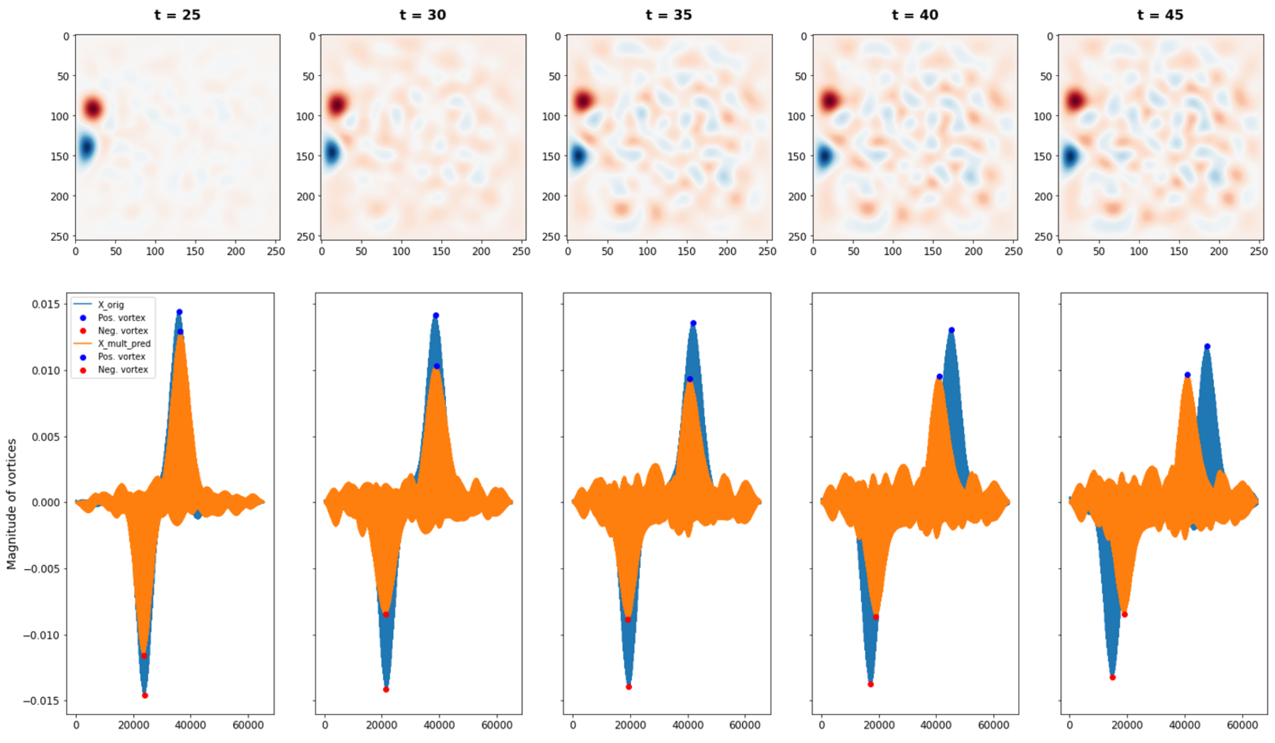
(c) AAE Prediction for trajectory.

Figure 7: Prediction of one time level (t=30) using WGAN-GP and AAE on a sample trajectory from train dataset. The first row of snapshots for each method shows the prediction. The second row visualizes the mismatch in the magnitude and location of the vortices for each prediction. The first 6 time levels are used to start the prediction while t=30 is the actual prediction. The AAE is shown to have a better performance. *Av. MSE - AAE: 1e-6; GAN: 2e-3*.

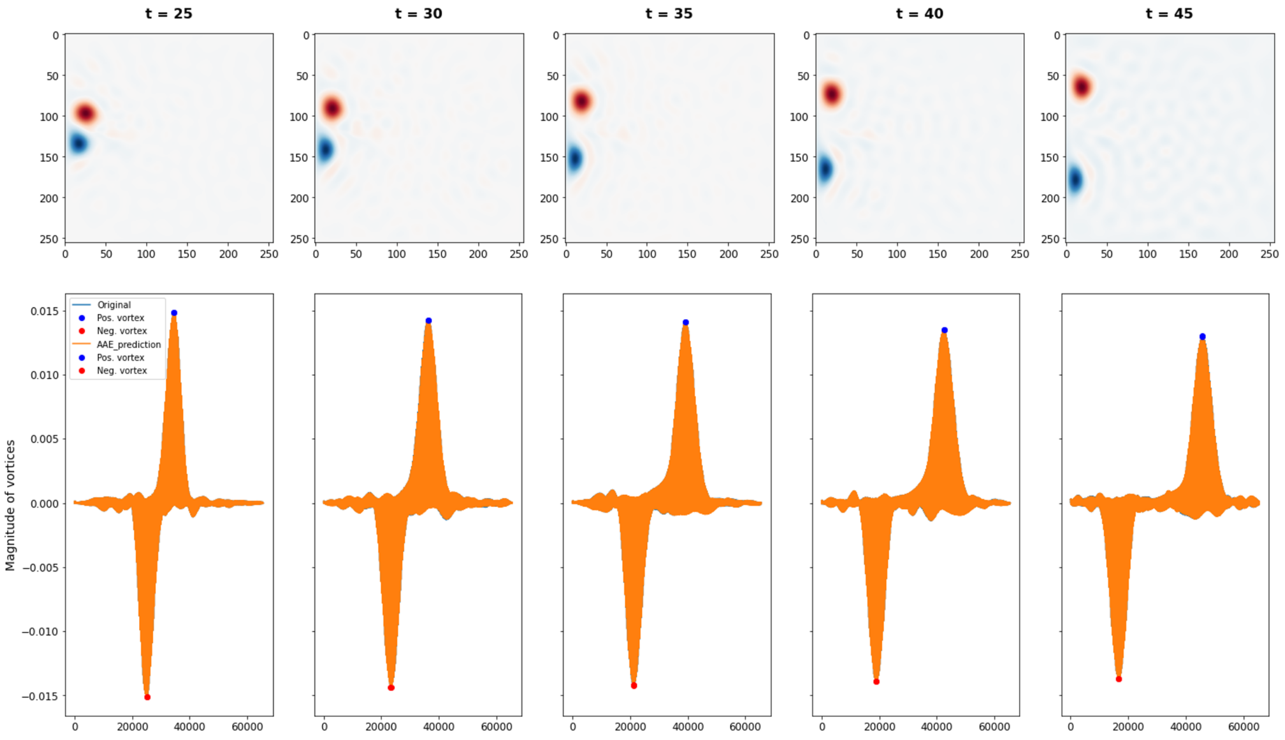
Multiple time levels prediction:



(a) original snapshots of trajectory (train data)



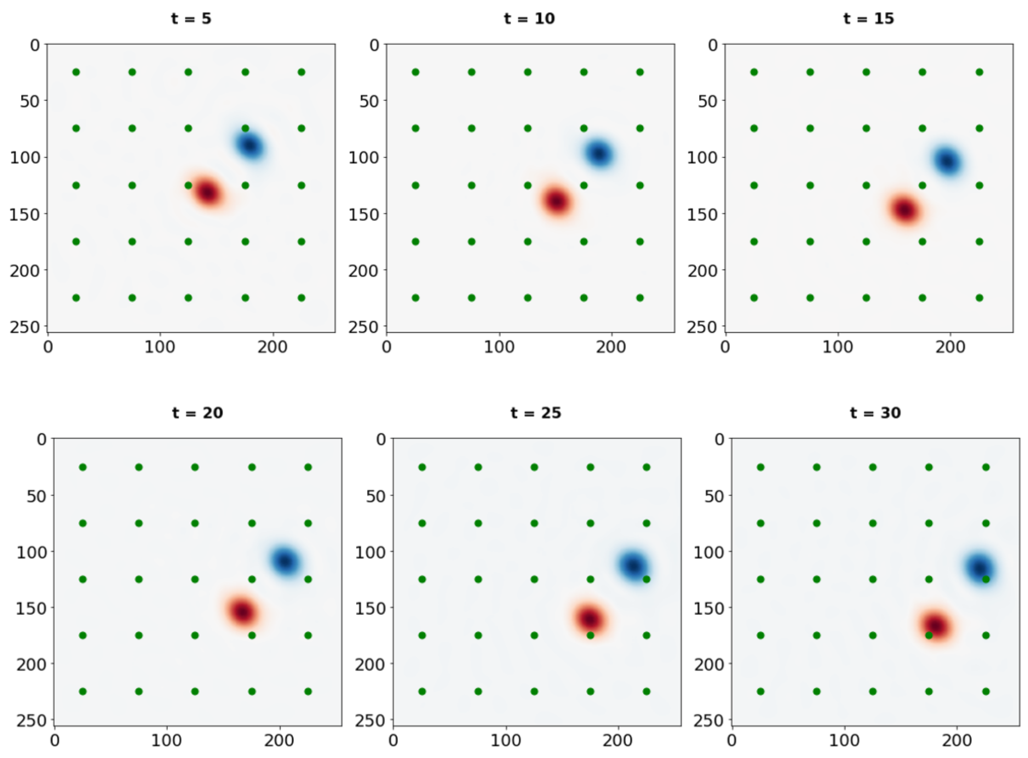
(b) WGAN-GP Prediction of trajectory.



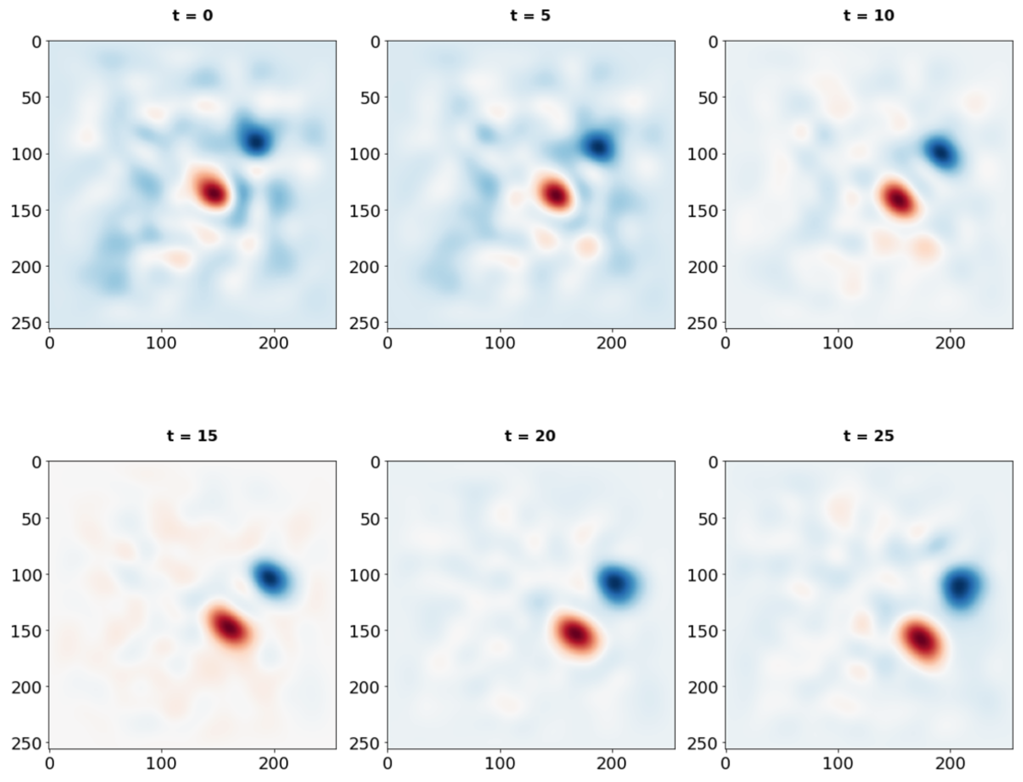
(c) AAE Prediction for trajectory.

Figure 8: Multiple time level prediction (t=35 to t=45) using WGAN-GP(b) and AAE(a) on a sample trajectory. Time levels (t=0 to t=30) are passed into the model to start prediction while the other time levels are generated outputs.

4.4 Data Assimilation Comparison



(a) Sensor locations (green dots) overlaid on a sample snapshot.



(b) assimilation using WGA-GP

Figure 9: Data assimilation using WGAN-GP. Using observed sensor data placed and locations in the domain, we show results of the assimilated data using the built WGAN-GP. The methodology uses a forward and backward march while optimizing POD coefficient loss and observation data loss

5 Discussion and Conclusion

The use of machine learning techniques for fluid modelling problems remains very promising. The low cost, speedup and relative accuracy provided by ML tools are attractive features in the study of forward modelling, data assimilation and uncertainty quantification. Specifically, the use of generative models remains relevant for fluid problems. In this work, an exploratory study is done to understand the capabilities of two generative models- Generative Adversarial Network (WGAN-GP) and Adversarial Autoencoder - for prediction and data assimilation. The results of the work provides some interesting results. One main results is in the prediction power of GAN compared to AAE. Visually, it is clear that the GAN fails to predict accurately moving forward in time with. While the AAE also experiences a similar situation, there is much less mismatch between original snapshots and predictions. Additionally, with the event of mode collapse, we conclude that depending on the system under study a Vanilla DC-GAN may be insufficient. More study is recommended with abundant data and training cycles to better understand the capabilities of generative modelling for predicting and assimilating data in turbulent flow modelling.

References

[1] B.E. Launder and D.B. Spalding. The numerical computation of turbulent flows. *Computer Methods in Applied Mechanics and Engineering*, 3(2):269–289, 1974. ISSN 0045-7825. doi: https://doi.org/10.1016/0045-7825(74)90029-2. URL https://www.sciencedirect.com/science/article/pii/0045782574900292.

[2] Steven L. Brunton, Bernd R. Noack, and Petros Koumoutsakos. Machine Learning for Fluid Mechanics. *Annu. Rev. Fluid Mech.*, 52:477–508, 2020. ISSN 00664189. doi: 10.1146/annurev-fluid-010719-060214.

[3] Vasily V. Titov, Frank I. González, E. N. Bernard, Marie C. Eble, Harold O. Mofjeld, Jean C. Newman, and Angie J. Venturato. Real-time tsunami forecasting: Challenges and solutions. *Nat. Hazards*, 35(1):41–58, 2005.

[4] D. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[5] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

[6] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014.

[7] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. ISSN 14764687. doi: 10.1038/nature14539.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[9] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial Autoencoders. 2015. URL http://arxiv.org/abs/1511.05644.

[10] D. Xiao, C.E. Heaney, L. Mottet, F. Fang, W. Lin, I.M. Navon, Y. Guo, O.K. Matar, A.G. Robins, and C.C. Pain. A reduced order model for turbulent flows in the urban environment using machine learning. *Building and Environment*, 148:323–337, 2019.

[11] M.A. Cardos, L.J Durlofsky, and P. Sarma. Development and application of reduced-order modeling procedures for subsurface flow simulation. *International Journal for Numerical Methods in Engineering*, 77(9):1322–1350, 2009. doi: 10.1002/nme. URL https://doi.org/10.1002/nme.2453.

[12] Nils Thuerey, Konstantin Weißenow, Lukas Prantl, and Xiangyu Hu. Deep Learning Methods for Reynolds-Averaged Navier Stokes Simulations of Airfoil Flows. *AIAA J.*, 58(1):25–36, 2020.

[13] D. Xiao, F. Fang, A. G. Buchan, C. C. Pain, I. M. Navon, and A. Muggeridge. Non-intrusive reduced order modelling of the Navier-Stokes equations. *Comput. Methods Appl. Mech. Eng.*, 293:522–541, 2015. doi: 10.1016/j.cma.2015.05.015.

[14] Vinicius L S Silva, Claire E Heaney, Yaqi Li, and Christopher C Pain. Data Assimilation Predictive GAN (DA-PredGAN): applied to determine the spread of COVID-19, 2021. URL http://arxiv.org/abs/2105.07729.

[15] 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, Dan 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. URL http://tensorflow.org/. Software available from tensorflow.org.

[16] Renee Swischuk, Laura Mainini, Benjamin Peherstorfer, and Karen Willcox. Projection-based model reduction: Formulations for physics-based machine learning. *Comput. Fluids*, 179:704–717, 2019. ISSN 00457930. doi: 10.1016/j.compfluid.2018.07.021. URL https://doi.org/10.1016/j.compfluid.2018.07.021.

[17] Omer San and Romit Maulik. Neural network closures for nonlinear model order reduction, 2017.

[18] Z. Wang, I. Akhtar, J. Borggaard, and T. Iliescu. Proper orthogonal decomposition closure models for turbulent flows: A numerical comparison. *Computer Methods in Applied Mechanics and Engineering*, 237-240:10–26, 2012. ISSN 00457825. doi: 10.1016/j.cma.2012.04.015. URL http://dx.doi.org/10.1016/j.cma.2012.04.015.

[19] D. Xiao, P. Yang, F. Fang, J. Xiang, C. C. Pain, I. M. Navon, and M. Chen. A non-intrusive reduced-order model for compressible fluid and fractured solid coupling and its application to blasting. *J. Comput. Phys.*, 330:221–244, 2017. ISSN 10902716. doi: 10.1016/j.jcp.2016.10.068.

[20] Lawrence Sirovich. Turbulence and the dynamics of coherent structures. III. Dynamics and scaling. *Q. Appl. Math.*, 45(3):583–590, 1987. ISSN 0033-569X. doi: 10.1090/qam/910464.

[21] Philip Holmes, John L Lumley, Gahl Berkooz, and Clarence W Rowley. *Turbulence, coherent structures, dynamical systems and symmetry*. Cambridge university press, 2012.

[22] J.L. Lumley. Atmospheric turbulence and radio wave propagation. *Atmospheric Turbulence and Radio Wave Propagation*, pages 166–178, 1967. cited By 94.

[23] V. Klema and A. Laub. The singular value decomposition: Its computation and some applications. *IEEE Transactions on Automatic Control*, 25(2):164–176, 1980. doi: 10.1109/TAC.1980.1102314.

[24] Jian Yang, D. Zhang, A.F. Frangi, and Jing yu Yang. Two-dimensional pca: a new approach to appearance-based face representation and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(1):131–137, 2004. doi: 10.1109/TPAMI.2004.1261097.

[25] D. Xiao, F. Fang, C. Pain, and G. Hu. Non-intrusive reduced-order modelling of the Navier-Stokes equations based on RBF interpolation. *Int. J. Numer. Methods Fluids*, 79(11):580–595, 2015. ISSN 10970363. doi: 10.1002/fld.4066.

[26] Saddam Hijazi, Giovanni Stabile, Andrea Mola, and Gianluigi Rozza. Data-driven POD-Galerkin reduced order model for turbulent flows. *J. Comput. Phys.*, 416, sep 2020. ISSN 10902716. doi: 10.1016/j.jcp.2020.109513.

[27] Ian Goodfellow. NIPS 2016 Tutorial: Generative Adversarial Networks. 2016. URL http://arxiv.org/abs/1701.00160.

[28] M. Cheng, F. Fang, C. C. Pain, and I. M. Navon. Data-driven modelling of nonlinear spatio-temporal fluid flows using a deep convolutional generative adversarial network. *Comput. Methods Appl. Mech. Eng.*, 365, 2020.

[29] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. 2017. URL http://arxiv.org/abs/1701.07875.

[30] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein GANs. *Adv. Neural Inf. Process. Syst.*, 2017-Decem:5768–5778, 2017. ISSN 10495258.

[31] M. Cheng, F. Fang, C. C. Pain, and I. M. Navon. An advanced hybrid deep adversarial autoencoder for parameterized nonlinear fluid flow modelling. *Comput. Methods Appl. Mech. Eng.*, 372, 2020. ISSN 00457825. doi: 10.1016/j.cma.2020.113375.

[32] Michael Ghil and Paola Malanotte-Rizzoli. Data assimilation in meteorology and oceanography. *Advances in Geophysics*, 33:141–266, 1991.

[33] Meng Tang, Yimin Liu, and Louis J. Durlofsky. A deep-learning-based surrogate model for data assimilation in dynamic subsurface flow problems. *J. Comput. Phys.*, 413:109456, 2020. doi: 10.1016/j.jcp.2020.109456. URL https://doi.org/10.1016/j.jcp.2020.109456.

[34] Vinícius Luiz Santos Silva, Alexandre Anozé Emerick, Paulo Couto, and José Luis Drummond Alves. History matching and production optimization under uncertainties: Application of closed-loop reservoir management. *Journal of Petroleum Science and Engineering*, 157:860–874, 2017. doi: 10.1016/j.petrol.2017.07.037.

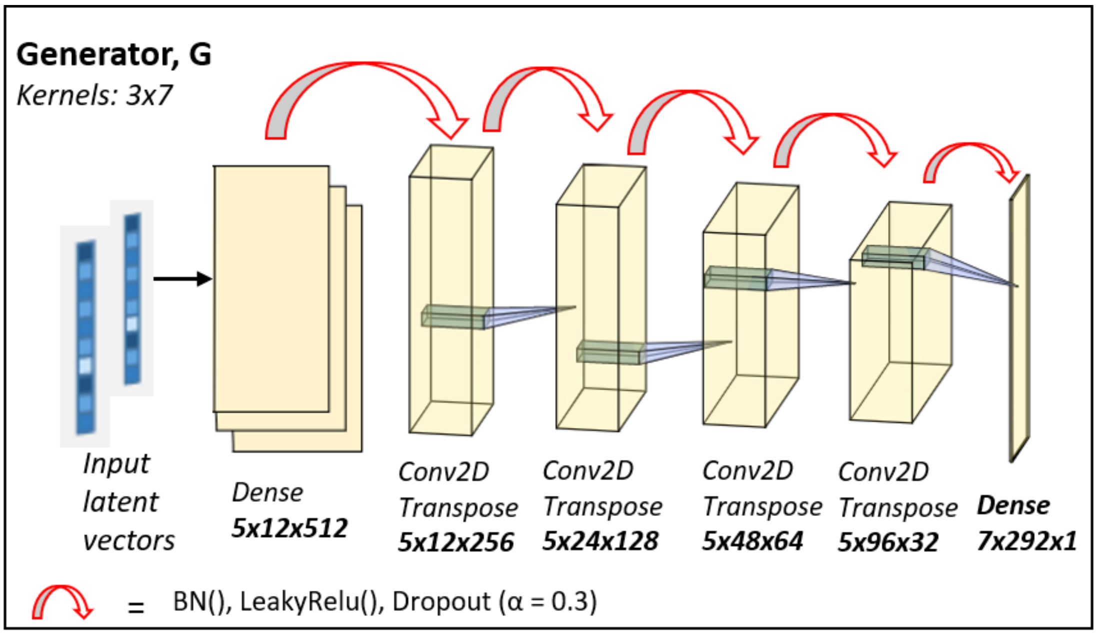
[35] Albert Tarantola. *Inverse problem theory and methods for model parameter estimation*. SIAM, 2005.

[36] Francisco J. Gonzalez and Maciej Balajewicz. Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems. 2018. URL http://arxiv.org/abs/1808.01346.

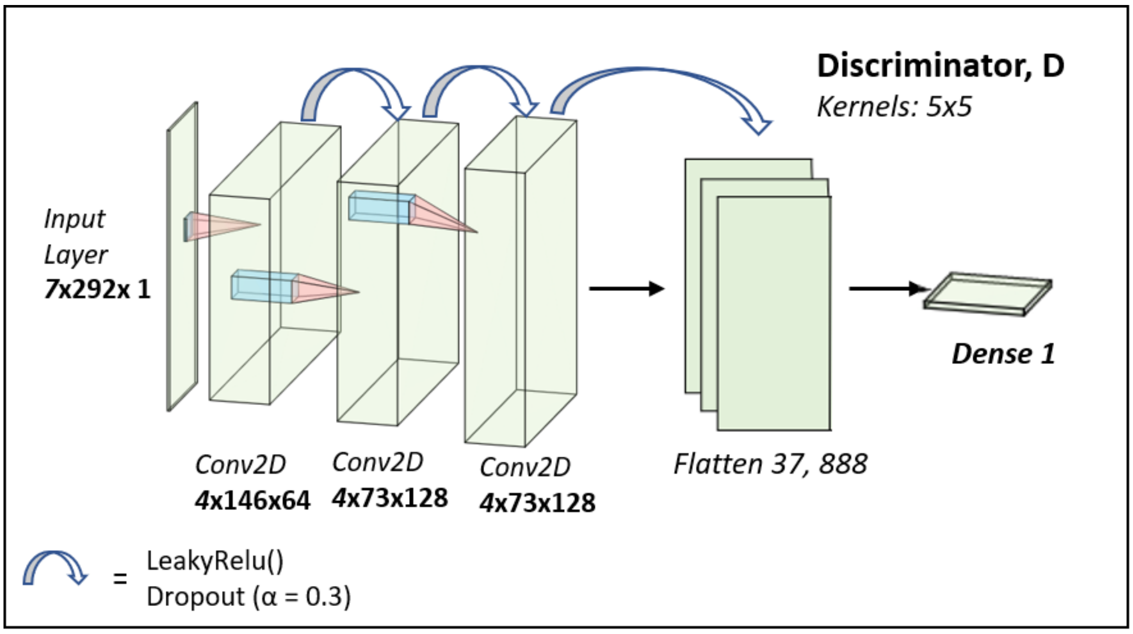
Appendices

A Generative Adversarial Network

A.1 Architecture and Parameter



(a) WGAN-GP Generator



(b) WGAN-GP Discriminator

Figure 10: GAN Architecture and training hyperparamters used in the development of the WGAN-GP generative model.

A.2 Mode Collapse using DCGAN:

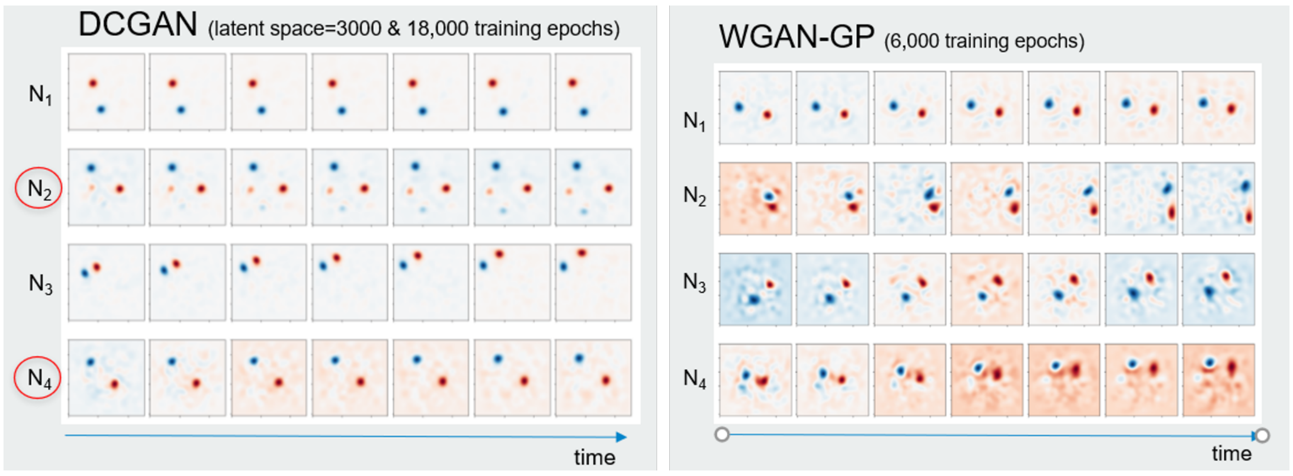


Figure 11: While mode collapse is difficult to measure, evidence can be obtained from visual judgment. Here the output fake snapshots generated from the WGAN-GP is compared to output from the DCGAN where snapshots (circled in red) are of the same mode. This occurred because the generator found a mode that can be used to easily trick the discriminator. The similarity between the modes in tthis data made this a more probable issue.