Imperial College London

Department of Earth Science and Engineering

MSc in Applied Computational Science and Engineering

Independent Research Project
Final Report

Data assimilation using Generative Adversarial Networks to determine COVID-19 infection risks in enclosed spaces using autoencoders for compression

by

Yushen Lin

yl2020@imperial.ac.uk
GitHub login: acse-yl2020

Supervisors:

Dr. Claire E. Heaney

Acknowledgement

I would like to express my gratitude and appreciation to my supervisors: Dr Claire E. Heaney, Prof. Christopher Pain and Dr Laetitia Mottet. I also wish to appreciate my friends Sanjana, Jamesson and Danhui in my own group, Jin,yu and Fan, Yang in autoencoder, YuJing, Wu in Virus modelling group, who give me kind helps and motivations. Thanks to their consistent support and precious guidance, I was able to complete the tasks of the independent research project successfully. And I would also like to appreciate Dr Marijan Beg and Percival, James R who provided me useful advice on writing project plan and final report. Finally, I would like to appreciate my family for mental and financial supports during the compilation of this project, in particular my mother Ping, Zou.

GitHub repository: https://github.com/acse-2020/acse2020-acse9-finalreport-acse-yl2020

Abstract

After the Coronavirus disease 2019 (COVID-19) outbreak in 2019, the importance of indoor air environmental safety has been increasingly recognised. However, usual simulations of airflow and air pollution concentration with high-resolution meshes e.g. Computational Fluid Dynamics (CFD), can be extremely computationally intensive. In this project, the proposed methods show great promise of accurately predicting CO_2 concentration in enclosed space, all set within a reduced-order model (ROM) framework. In order to make training stable, a new method of updating the weight in loss function is also proposed. Various compression methods will be investigated during the project, such as the Singular Value Decomposition-Autoencoder (SVD-AE) [38] and Space-Filling Curve - Convolutional Autoencoder (SFC-CAE) [18] and other linear dimensionality reduction approaches.

Our work shows great promise to speed the dynamic model and the data assimilation process. More importantly, by using autoencoders for dimensionality reduction instead of methods based purely on the linear compression methods, more effective compressed representations of data might be achieved.

Keywords: COVID-19, ROM, PredGAN, DA-PredGAN, carbon dioxide (CO_2), TensorFlow

1 Introduction

COVID-19 is a recently identified coronavirus that causes an infectious illness [33]. It is currently sweeping the world and has caused the death of over 4.3 million people worldwide [24].

With the process of urbanisation, the increasingly crowded living environment of modern large cities and the rapidly growing worldwide transportation network will accelerate the spread of airborne diseases [48, 29]. This problem already has been discovered after the outbreak of Severe Acute Respiratory Syndrome (SARS) epidemic in 2003 [29]. Many studies have shown that airborne diseases are related to factors such as ventilation and airflow [25, 29, 37]. Knowing the transmission routes of airborne diseases and accurately predicting the probability of airborne diseases with their influencing factors can help to take measures to reduce the probability of infection [25], which brings the attention from the researchers for the model to predict indoor air quality.

However, complex systems in high fidelity numerical simulations are extremely computationally intensive in order to achieve high quality grid (mesh) [34]. There are many researches combining machine learning (ML) with CFD in which could be accelerated. Guo et al. [17] applied deep Convolutional Neural Networks (CNN) on CFD model to resolve the problem of computationally expensive required by traditional CFD methods to model laminar flow in steady-state. And a novel and customised ConvNet framework [49] seems that it is promising to be strong candidates to replace existing Eulerian-based solvers, while their model brings speedup and similar visual results comparing with preconditioned conjugate gradient method. And Recurrent Neural Network (LSTM) has been built by Amendola et al. [3] as surrogate model of CFD simulation to predict the concentration of CO_2 in a room.

The idea of an autoencoder (AE) was first proposed in 1986, by Rumelhart et al. [43] introducing pioneering work on AE by using the input as the 'teacher' to overcome the problem of back propagation without supervision. It is one type of unsupervised feed-forward NN (refer to appendices 6.2). Due to computational complexity and sparse data, the model was proved challenging to optimise in high dimension and it is also known as 'curse of dimensionality' [6], which was one of the main restrictions of applying this network widely. This problem had existed until 2006, Hinton et al. [21] optimise restricted Boltzmann machine (RBM) [47] through each layer to achieve the abstract representation of original samples and features by using gradient descent.

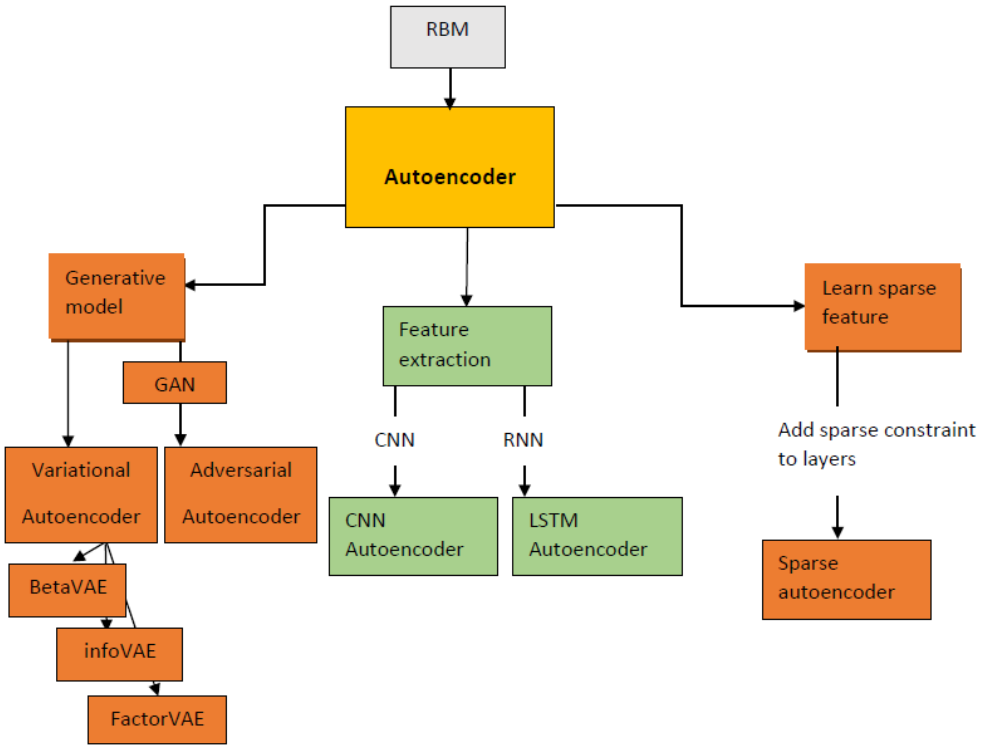


Figure 1: It shows that the trees of the methods that are invented from the original autoencoder and its origin.

However, fully connected autoencoder is restricted by the problem of computationally expense when the input vector is long [25]. In order to alleviate this problem, Phillips et al. [38] proposed a novel method of combining SVD with AE which is capable of lowering the difficulty of training the network and making overfitting less probable [38].

Vincent et al. [51] proposed Denoising Autoencoder for alleviating the problem classic AE being easy to overfit, by adding random noise to the input layer of AE to improve performance of the model. Another way is to use regularisation, Rifai et al. [41] proposed an architecture that constrains the encoder by adding the Frobenius norm of the Jacobian matrix so that encoder could compute the representation through encoder mapping.

From the family of generative models, the Variational Autoencoder(VAE) [27] is a milestone which brings an idea to researchers of using probabilistic model to enhance the robustness of system. There are many well-known extensions based on VAE such as InfoVAE [53], Beta-VAE [20] and FactorVAE [11] etc.

In Figure 1, various autoencoders could be traced back to the original autoencoder network. When it comes to generative model, GAN is always mentioned by researchers. As Goodfellow proposed the most well-known unsupervised learning approach for generative modelling using deep learning methods [16], that entails automatically detecting and learning the regularities or patterns in input such that the model could produce new samples following the same distribution of the training dataset [46, 7]. In the GAN model, D and G are defined as discriminator and generator, where the objective of G is to compete against D whose objective is to classify the fake and real dataset. For detailed derivations refer to appendices 6.3.

However, with the development of deep learning in recent years, models such as CNN and Recurrent Neural Networks have been widely used in combination with CFD and prediction, very few use GAN to

make prediction in time and assimilate the data. To spatio-temporally predict by taking the advantages of GAN mentioned in section 6 of paper [16], a novel generative adversarial network named Predictive GAN (PredGAN) is proposed in [46], an iterative process is applied to the G for marching forward in time after the GAN has studied the evolution of the system. Furthermore, based on the model of PredGAN, the iteration process is extended in order to assimilate observed data and generate the corresponding model parameters. The proposed contributions seem promising to eliminate the further simulation of the high-fidelity numerical model within the time of data assimilation process [46].

1.1 Objectives

The existing research [30, 46, 45] of reducing the dimension of simulation data and then predicting in time or assimilating data mainly focus on linear compression methods e.g. PCA [26], which is the most common method used for dimension reduction. However, non-linear compression methods are not widely used and it starts to be exploited by some researchers [15, 42, 13]. As CO_2 could be utilised as a surrogate for SARS-CoV-2 concentrations inside since it is co-exhaled with aerosols carrying SARS-CoV-2 by COVID-19 infected individuals [37], the objectives of this project are to construct a reduced-order model of the CO_2 flow inside a room using various non-linear and linear compression methods for the dimensionality reduction and a GAN for the prediction. Then assimilated some data was collected at sensors from experiment [48] with the DA-PredGAN algorithm.

In this project, two non-linear dimensionality reduction methods (SVD-AE and SFC-CAE) are mainly investigated and different linear approaches are also applied for further comparisons based on the research [38, 18, 26, 9]. Also, distinct from the previous work [38, 46], The novelty of this project include a new way of updating weight in the loss function with the method SVD-AE, and the enhanced training of PredGAN on the larger data set comparing with proposed research [46] is also proposed (refer to section 2) with detailed trial observations.

The methodologies of dimensionality reduction and prediction will be presented in **section 2**. Code Metadata will be shown in **section 3** following with the results in **section 4**. Finally, **section 5** will discuss the results with observations and deliver the conclusions and limitations of the project including the future plan.

2 Software Description/Methodology

This section summarises the methodologies of scaling, compression and prediction. All those methods and results were coded and visualised in Python based on paper [46, 8] and code [45]. Google Colab is chosen as platform due to its superior ability of visualisation, GPU usage and data I/O processing and it also supports bash commands. The goal of this software is to gain better understanding of how COVID-19 will spread within a room in order to meet the urgent need to study the dynamics of airborne transmission [8], and the software is implemented using TensorFlow [1], PyTorch [35] and scikit-learn [36] in Python. The dimensionality reduction methods SVD-AE and SFC-CAE are built upon TensorFlow [1] and PyTorch [35] respectively.

2.1 Data and Scaling

The room where the experimental data had been collected is located in the Clarence Centre (belonging to London South Bank University) and has 3 windows, two on one side with one on the other facing the road in London (see appendices 6.4). Levels of CO_2 were collected by seven sensors located at different optimised positions within the room [48]. Provided by Laetitia Mottet, the given simulation data set consists of 455 (time series) vtu files. The concentration of CO_2 data could either be extracted using *vtktools* library [2] or *meshio* [44] provided in the software.

In general, learning algorithms including AE could gain great benefit from scaling the data set properly. The reason of the model could be beneficial from scaling is that size of gradient used to update the weights could be reduced during the training process and model turns to be stable, otherwise the higher value of representations in the data set will dominate the calculation of the distance and slow down the gradient descent. Based on trial observations, the AE-based models has been found that the input data into the model should be carefully pre-processed. The data has been first scaled between 0 and 1, however, the large amount of scaled data will be distributed to 0 (151,128 in 595,624 nodes if velocity data is added (refer to appendices 6.7), which may harm the performance of the prediction in GAN as it may lack of meaningful representations that will be learned by GAN. Therefore the scaling range are set between -1 and 1 by using *MinMaxScaler* in sklearn [36].

2.2 SVD-AE

2.2.1 Singular Value Decomposition

SVD is capable of reducing a large matrix into relative small matrix. For a matrix X, the SVD is defined as:

$$\underbrace{\mathbf{X}}_{Q\times W} = \underbrace{\mathbf{U}}_{Q\times Q} \times \underbrace{\mathbf{\Sigma}}_{Q\times W} \times \underbrace{\mathbf{V}^{\mathsf{T}}}_{W\times W} \tag{1}$$

where $\mathbf{\Sigma}$ represents a diagonal matrix that is contained singular values of X in descending order, U and V denote left- and right-singular vectors respectively. While column of U denotes how much information is contained in the corresponding basis function in the square of each singular value, the rows of U represents the coordinates of the objects in the space spanned by the new axes. The decomposition X in equation (1) is called singular value decomposition [38].

Low-rank approximation could be yielded from SVD with relative low reconstruction error, it is worth noting that the resultant matrices cannot be used to interpret the individual components of the actual data. The reason of using SVD to reduce the dimension first is that the autoencoder is computationally expensive when the input vector is long [25] [38], especially in the data of Clarence centre (595,624 nodes when velocity is added). To alleviate this problem, the length of the input vectors could be reduced by using SVD from the number of degrees of freedom (F) to the basis functions (B where $B \ll F$) [38]:

$$\mathbf{r} = \mathbf{B^T}\Phi \tag{2}$$

where B denotes the POD (proper orthogonal decomposition) basis functions, r the reduced variables and Φ is snapshot.

2.2.2 Autoencoder

As a type of feed-forward neural network that is able to learn the representation of the input data, there are m neurons that could be set in input and output layer and n neurons in hidden layers where $n < m$. The autoencoder consists of two networks, an encoder, mapping the input to a 'bottleneck' layer and a decoder mapping from the bottleneck (latent layer) to the output layer. The AE can be expressed as a composition of functions as shown in appendices 6.2 [19, 38].

AE could properly learn the 'identity map' [38] if the model is well trained. Our model has been optimised based on the data provided, and this involves training different number of neurons in model each with different set of hyperparameters. Then the latent variables in encoder will be reconstructed back to the original physical space and the mean square error (refer to 6.8) will be calculated for the analysis.

As proposed in [38], SVD-AE is promising to provide more accurate results of dimensionality reduction comparing with POD-based ROM and standard autoencoder. However, if training data is insufficient, the training of the model turns to become unstable and overfitting may occur. Here we introduce a custom loss function instead of calculating mean square errors between true and prediction values by using backend of *Keras* [1]:

$$\mathbf{L} = \operatorname{argmin} \sum_{i=1}^{m} c_{rt} ||\mathbf{x_r} - \mathbf{x_t}||^2 \tag{3}$$

where L denotes the new loss function in the AE to minimise, x_r, x_t are reconstructed (prediction) and true values (input) in AE and c_{rt} is *cosine similarity* of x_r and x_t metrics as weight. The symbol $||\cdot||$ represents Euclidean distance. The cosine between x_r and x_t is used as the measurement of similarity between both:

$$\mathbf{c_{rt}} = \exp\left(\frac{-\cos(x_r, x_t)}{w}\right) \tag{4}$$

where w is hyper-parameter to tune and set to 1 that is selected based on trial observations, which means that small difference between x_r and x_t are less penalised and the large difference are more penalised.

For the model of autoencoder a $d - d - \frac{d}{2} - \frac{d}{2^d} - ... - l$ architecture is applied, where d denotes the dimensionality of the data (the number of basis functions in this case) and l is the dimension of the bottleneck layer. The models were trained using a decreasing learning rate, starting from relative large value and decreasing it gradually through epochs. And *EarlyStopping* is used as a indicator of stopping the training once the monitored metric has stopped improving. The activation function and optimiser are chosen as *Relu* [32] and *Nadam* [12] which generally provide better results based on trial observations.

2.3 Space Filling Curves - Convolutional Autoencoder

As general CNN technology is based on the structured grids (2D or 3D), there is huge difficulty of applying this to data on unstructured meshes. One way of addressing this issue is to use space-filling

curves (SFC) to transform/re-order the data [18]. SFCs show great promise that it is able to provide optimised ordering by traversing all the points on the unstructured meshes continuously, and it structurally reorders data in multi-dimensional space that maps to 1D space. For a simple Hilbert curve construction based on SFC, refer to appendices 6.5. The training of CNNs could be beneficial from the structured data as its filters will detect for structurally coherent features [18]. One continuous curve will be generated with nodes which are near to one another as neighbors for every node within the curve. To identify local features in CNNs, nodes on the SFC should be ordered so that they are physically adjacent to one another, allowing any corresponding coherence in the input to be recognised. As shown in appendices 6.5, the contour plot of 2D unstructured mesh based on a coordinate sequence at 20 levels of colormap is shown.

2.3.1 Hyper-parameters and architecture selections

The model is optimised by tuning following parameters: optimizer, kernel size, activation functions, number of channels and layers. The optimised model is promising to reduce the dimension to 43 and reconstruct back to physical space (visualisation refers to 6.6), and hyper-parameters are set as following:

optimizer	activation function	kernel size	learning rate	batch size
Adamax	Tanh	176	1×10^{-3}	16

Table 1: optimised hyper-parameter settings for training CNNs.

The architectures of SFC-CAE based on three SFCs is shown in Table 1. the data in multi-dimensional is first transformed to 1D space in SFC orderings. It is worth noting that there are limited connections between some areas of the domain when utilising the SFC method as it is true when the points are close on SFC that is also suitable for ND space ($N > 1$) but the reverse is not correct. In order to reduce the loss of the CNN, 3 SFCs are used in 3D Clarence data as aforementioned ($CO2_{ppm}$ and velocity fields). The CAE is followed with sparse layer 1 which takes the input in the size of (148906,4) where 148906 is the number of nodes and 4 represents the data of CO_2 (148906,0) and velocity (includes x, y and z axis). Then the channels will be copied and concatenated at second dimension, 4 will be doubled to 8, which results in the input size of second layer turns to 1191248 (after flatten to 1d as the properties of SFCs). After the second layer, the model is followed with 4 Conv1d layers with 176 kernel sizes and 16 channels. The calculation of each Conv1d layer could refer to appendices 6.8. Then decoder will be symmetric as encoder by using TransConv1d to match with each other as shown in Table 2. During the training, *Tanh* activation function is used based on tests as its ability of strongly mapping negative inputs to negative and the zero inputs to where near zero.

layers	input size & ordering	kernel size	channels	stride	padding	output padding	output size & ordering	activation
1-FEM (ANN input)	(148906, 4, FEM)	1 Identity	4	1	0	0	(148906, 4, SFC$\mathcal{C}$) $\forall\, \mathcal{C} \in \{1,2,3\}$	Identity
Encoder								
Copying channels and concatenate at the 2nd dimension to form (148906, 8, SFC$\mathcal{C}$), flatten it to 1D.								
2-ExpandNN-SFC$\mathcal{C}$	(1191248, 1, SFC$\mathcal{C}$)	3 Variable (3×1191248)	1	1	0	0	(1191248, 1, SFC$\mathcal{C}$)	Tanh
Reshape (1191248, 1, SFC$\mathcal{C}$) to form (148906, 8, SFC$\mathcal{C}$)								
3-Conv1d-SFC$\mathcal{C}$	(148906, 8, SFC$\mathcal{C}$)	176	16	8	88	0	(18614, 16, SFC$\mathcal{C}$)	Tanh
4-Conv1d-SFC$\mathcal{C}$	(18614, 16, SFC$\mathcal{C}$)	176	16	8	88	0	(2327, 16, SFC$\mathcal{C}$)	Tanh
5-Conv1d-SFC$\mathcal{C}$	(2327, 16, SFC$\mathcal{C}$)	176	16	8	88	0	(291, 16, SFC$\mathcal{C}$)	Tanh
6-Conv1d-SFC$\mathcal{C}$	(291, 16, SFC$\mathcal{C}$)	176	16	8	88	0	(37, 16, SFC$\mathcal{C}$)	Tanh
7-FC	1776 ($= 37 \times 16 \times 3$)						222	Tanh
7-FC	222						43	Tanh
Decoder								
9-FC	43						222	Tanh
10-FC	222						1776	Tanh
Split the data into 3 sequences as the input of layer 11-TransConv1d-SFC$\mathcal{C}$ $\forall\, \mathcal{C} \in \{1,2,3\}$, convert from 592 to (37, 16)								
11-TransConv1d-SFC$\mathcal{C}$	(37, 16, SFC$\mathcal{C}$)	176	16	8	88	3	(291, 16, SFC$\mathcal{C}$)	Tanh
12-TransConv1d-SFC$\mathcal{C}$	(291, 16, SFC$\mathcal{C}$)	176	16	8	88	7	(2327, 16, SFC$\mathcal{C}$)	Tanh
13-TransConv1d-SFC$\mathcal{C}$	(2327, 16, SFC$\mathcal{C}$)	176	16	8	88	6	(18614, 16, SFC$\mathcal{C}$)	Tanh
14-TransConv1d-SFC$\mathcal{C}$	(18614, 16, SFC$\mathcal{C}$)	176	16	8	88	2	(148906, 8, SFC$\mathcal{C}$)	Tanh
Apply inverse SFC orderings to (148906, 4, SFC$\mathcal{C}$) $\forall\, \mathcal{C} \in \{1,2,3\}$, flatten into (1191248, 1, FEM$\mathcal{C}$)								
15-ExpandNN-FEM$\mathcal{C}$	(1191248, 1, FEM$\mathcal{C}$)	3 Variable (3×1191248)	1	1	0	0	(1191248, 1, FEM$\mathcal{C}$)	Tanh
Reshape (1191248, 1, FEM$\mathcal{C}$), and separate self-concat channels, to form (148906, 8, FEM$\mathcal{C}$) $\times$ 2								
16-FEM (ANN input)	(148906, 4, FEM$\mathcal{C}$) $\times$ 6	$\sum_{i=1}^{6} \omega_i$·FEMi	4	1	0	0	(148906, 4, FEM$\mathcal{C}$)	Tanh

Table 2: The architecture of SFC-CAE with 3 SFC orderings. '3 Variable' would give the nearest-neighbour smoothing. The number of latent layers are set to 43 which also can be adjusted from 1 to any proper value. Whenever a non-identity activation term is used a bias is added to each neuron.

2.4 Other methods

Apart from the methods of compression aforementioned, the additional methods had been evaluated for further comparison. The principle component analysis (PCA) [26] and Non-negative matrix factorisation (NMF) [9] had been tested in this project. The detailed explanations and comparisons refer to appendices 6.6 and section 4.

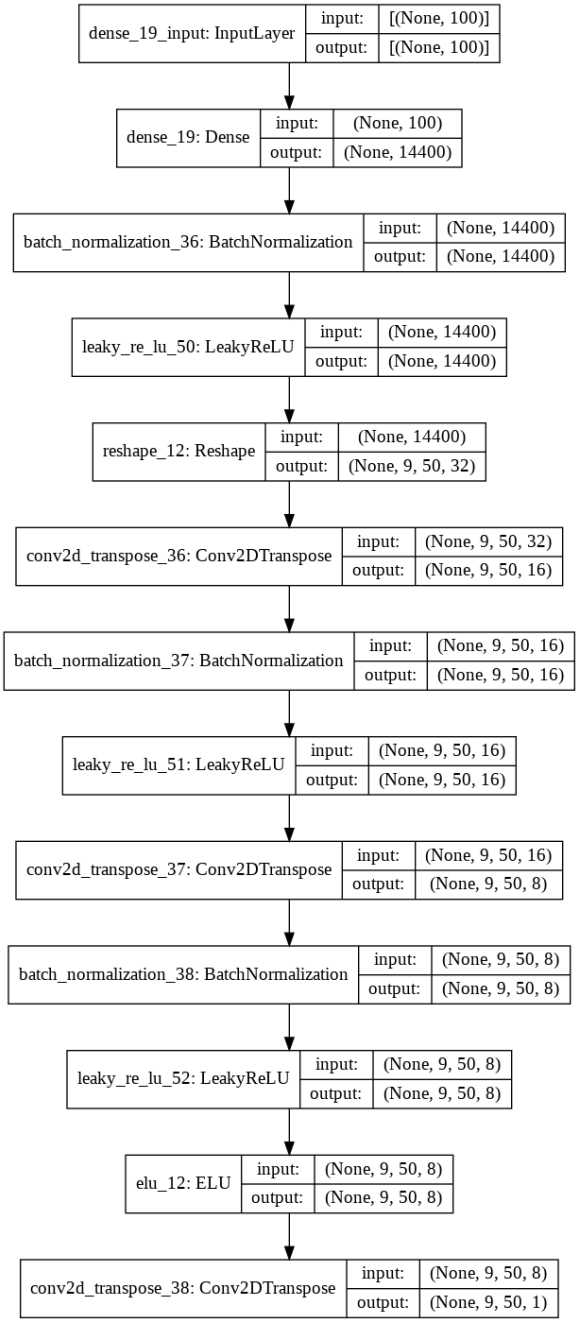
2.5 PredGAN

PredGAN is built upon Deep Convolutional Generative Adversarial Networks (DCGANs) [39] and code [45]. Distinct from the previous work, we also implement the 'enhanced training' as mentioned in objectives. The trained PredGAN is expected to generate pod coefficients or latent variables from non-linear dimensionality reduction approaches embedded with sensor values in consecutive time levels as following form:

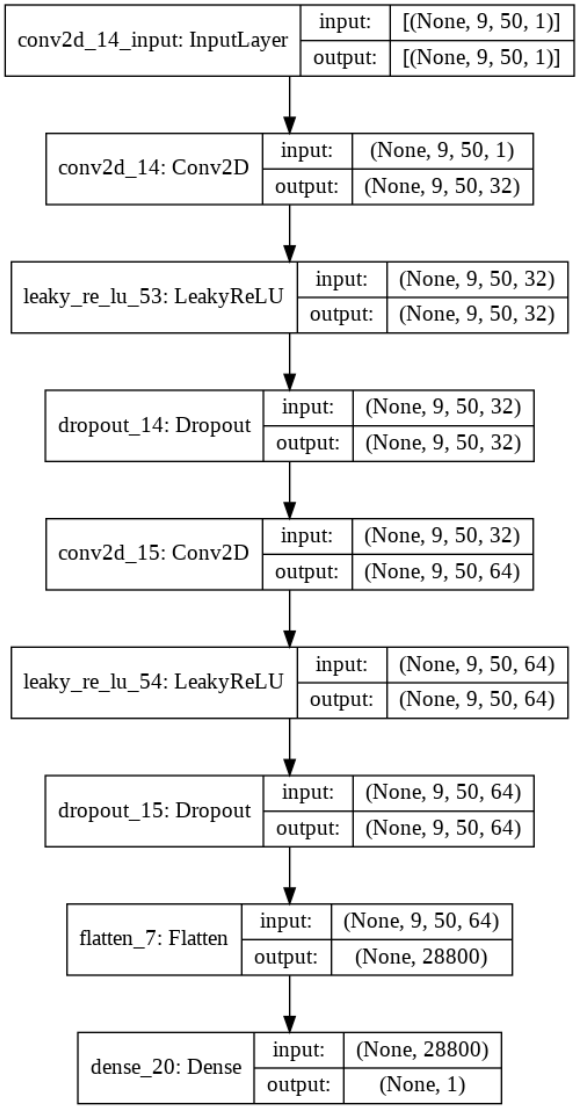
$$\widetilde{v}^m = \begin{pmatrix} (\alpha^m - n + 1)^T, (o^m - n + 1)^T \\ (\alpha^m - n + 2)^T, (o^m - n + 2)^T \\ \vdots \\ (\alpha^m - 1)^T, (o^m - 1)^T \\ (\alpha^m)^T, (o^m)^T \end{pmatrix},$$

where $(\alpha^m)^T = [\alpha_1^m, \alpha_2^m, ..., \alpha_{N_{latent}}^m]$, $(o^m)^T = [o_1^m, o_2^m, ..., o_{N_{sensors}}^m]$. N_{latent} denotes the number of latent representations, α_i^m is the ith latent variables at time level m, $N_{sensors}$ is the number of sensors, and o_i^m is the observation data of the ith sensor at time level m.

By using enhanced training in the GAN, the 43 POD coefficients (by keeping 99.5% explained variances after applying PCA, the latent representations from non-linear dimensionality reduction are used in this project) are combined with flow fields at 7 sensor locations that are extracted from the CFD data by taking the node IDs of seven sensors provided. This type of training could increase the efficiency of the training because the reconstruction of the primitive variables from POD coefficient is not required and the assimilation of data could directly obtained without simulations from high-fidelity model as the sensor observations are not directly carried out in the dimensionality reduction methods.



(a) flowchart of architectures of generator.



(b) flowchart of architectures of generator.

Figure 2: Architectures of PredGAN: generator(left) and discriminator(right).

For starting the prediction at time level $n-1$, the latent vector v will be treated as random noise, while the current latent variables will be evaluated by the generator in GAN:

$$\mathbf{G}(^{\mathbf{i}}\mathbf{v}^{\mathbf{m}}) = {}^{\mathbf{i}}\phi^{\mathbf{m}} \tag{5}$$

where i denotes iteration counter of optimisation, $m = n-1$ is time iteration counter and ϕ is the output from generator. Then new latent variables at the new iteration counter of optimisation ${}^{i+1}\phi^{m}$ could be updated by minimising the gradient of the loss ζ_p that is calculated w.r.t the current ${}^{i}\phi^{m}$ and defined as:

$$\zeta_{\mathbf{p}} = \sum_{k=m-n+1}^{m-1} (\hat{a}^k - {}^i a^k)^T \Lambda_a (\hat{a}^k - {}^i a^k) + \sum_{k=m-n+1}^{m-1} (\hat{o}^k - {}^i o^k)^T \Lambda_o (\hat{o}^k - {}^i o^k) \tag{6}$$

where Λ_a is the matrix that determines the relative importance of latent variables from non-linear dimensionality reduction methods of size N_{latent}, while it sets to constant rather than the singular values proposed [46] based on considerably trial and error, and Λ_o is a square matrix of the size $N_{sensors}$. All these processes will repeat until converged. The gradient of loss function is determined w.r.t the input latent values ${}^i\phi^m$, which incurs the process of back-propagation. Once the optimisation of minimising loss function is converged, the next time step prediction is therefore determined as the known solution $\hat{a}^k = a^k$. The realistic and promising model then could be obtained by making full pass predictions through stepping in time, and carrying on to iterate the process of predictive optimisation steps until final time level is reached [46].

2.6 DA-PredGAN

Based on the equation 6 and assuming it is Γ, the functional of forward and backward march will be modified to equation 7 by adding one more term accounting for the data mismatch between observations and generated values from generator as proposed in [46].

$$\zeta_{\mathbf{da,f}}(\mathbf{v^m}) = \mathbf{\Gamma} + \sum_{k=m-n+1}^{m-1} \zeta_{obs} (u^k - (u^{obs})^k)^T \Lambda_u^k (u^k - (u^{obs})^k), \tag{7}$$

where ζ_{obs} is the scalar that determines how much relevance should be given to the data mismatch, Λ_u^k denotes the matrix, and its diagonal values are the weights of observed data in size of observed data (we choose 0.0005 for latent variables from SFC-CAE based on trial and error).

The purpose of the first term in equation above is to link time levels. We are performing this optimisation at each time level, the continuities between all the time levels should be provided, as we need a way of passing the information from previous time levels backwards, therefore the time level will not be abandoned in the future as it could be considered as a way of back-propagating the residues through the time. The middle term in the original paper [46] stops the model parameters from changing too rapidly on each time step, however the model parameters will not be analysed in this project. DA-PredGAN is modified based on PredGAN itself and it identifies in different ways:

1. Because the DA algorithm attempts to determine the values on the sensor locations, the input is no longer defined as the prior.

2. Forward and backward both march in time.

Between the predicted and observed data at different sensor locations, the average data mismatch could be obtained through the backward march. Then iterations of marching forward and then backward will stop once the default setting of the number of iterations reach the maximum or the value of relaxation factor drops below its default setting (0.05).

It is worth mentioning that if the compressed variables are from POD, the POD compressed variables have a linear relationship between solution (CO_2 and the sensors) and primitive variables. If it is linear, we could look at the appropriate rows of the matrix, in which the rows associate with the nodes in finite element mesh. However, there is no linear relationship between compressed latent variables and primitive variables in SFC-CAE, which brings the challenge of tuning the model as the latent representations are hard to interpret.

GitHub repository: https://github.com/acse-2020/acse2020-acse9-finalreport-acse-yl2020

3 Code Metadata

This project was carried out and built under Google Colab with Python ($\geq$3.5). The data is collected into vtu files which could be either downloaded from shared link in Google Drive or Dropbox by using Library *Wget* ($\sim$=3.2) provided in software as the data from Clarence centre exceeds the Maximum storage limit of GitHub repository (1GB), and *vtktools* ($\geq$ 9.0) [2] and *meshio* [44] provide users different ways of extracting data from vtu files. For standard libraries, libraries of *matplotlib* ($\sim$=3.2.2) are used for visualisation, *numpy* ($\sim$=1.19.5) used for data processing, *joblib* for dumping and loading large arrays, *Scikit-learn* for scaling data, analysis and dimension reduction [36] ($\geq$0.20), *Tensorflow* [1] ($\geq$2.0) and *Pytorch* ($\geq$1.8.0) [35] are for building the models of deep neural networks and GAN. The *h5py* ($\geq$3.4.0) [10] is also included for reducing the RAM usage in software. The SFCs approach is built upon Fortran code provided for efficiency.

In the GitHub repository[1], the corresponding .ipynb files and source codes of proposed methods mentioned in section 2 are uploaded in *src* directory as following:

- Compressions and training: extract data from vtu files and proposed dimensionality reduction approaches on dataset, then train the GAN based on the latent variables from different methods.
- DA-PredGAN: implement DA-PredGAN algorithm.
- PredGAN: train an GAN network to predict and assimilate data.
- SFC-CAE: implement SFC-CAE approach for dimensionality reduction, including .py files for the models and Fortran code for SFCs.
- trial and error: the trial and error of different models and hyper-parameter settings of the generator, discriminator and autoencoder are put into this folder.

Users could directly execute .ipynb files in the files aforementioned for their own purposes. Figures of experimental results can be found in Figures directory and .ipynb files. The detailed explanations are given in readme file.

4 Results

The section will mainly illustrate results from different dimensionality reduction approaches and predictions from PredGAN and DA-PredGAN [46]. PCA [26], NMF [9] and SFCs-CAE [18] are utilised to compress the data from 595,624 nodes to 43 latent representations, then the well-trained PredGAN is used to predict the CO_2 distribution within the room.

The performance of the dimensionality reduction methods are visualised in ParaView (refer to Apendices 6.6), all the dimensionality reduction approaches could reconstruct the latent variables (43) back to physical space that are close to true data in CO_2 field (refer results of velocity filed in appendices 6.5), which shows great promise of compressing $\sim$ 0.6 million nodes with complex continuous Galerkin stencil to 43 or 16 (any appropriate numbers greater than 0) by observing the reconstruction of physical space in ParaView.

For the SVD-AE part, the ROM based on proposed loss function with different strategies of training (refer to section 2.2) and original one [38] are analysed. The former provides more stable training and lower MSE between true and predicted values under same hyper-parameter settings comparing with the original one:

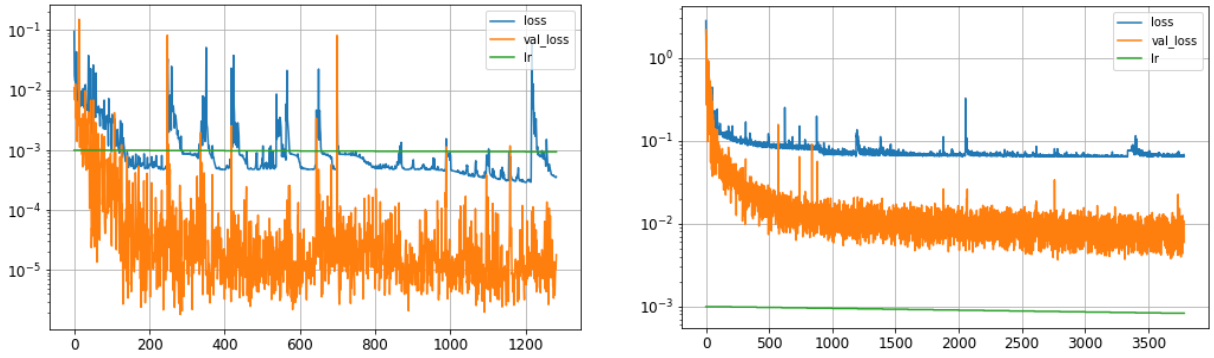
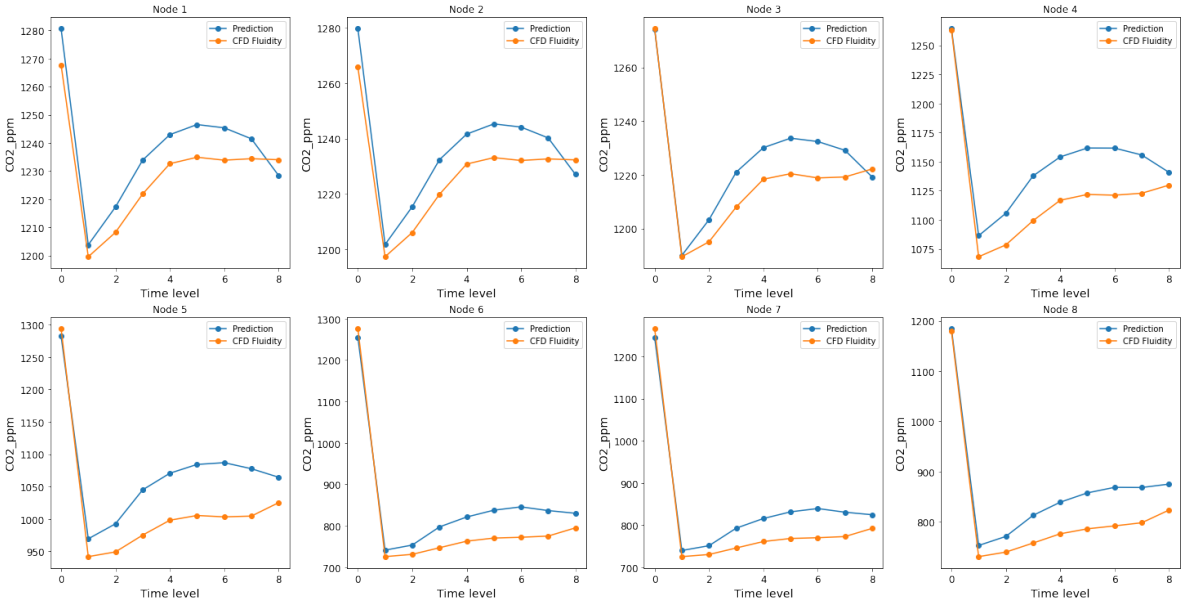


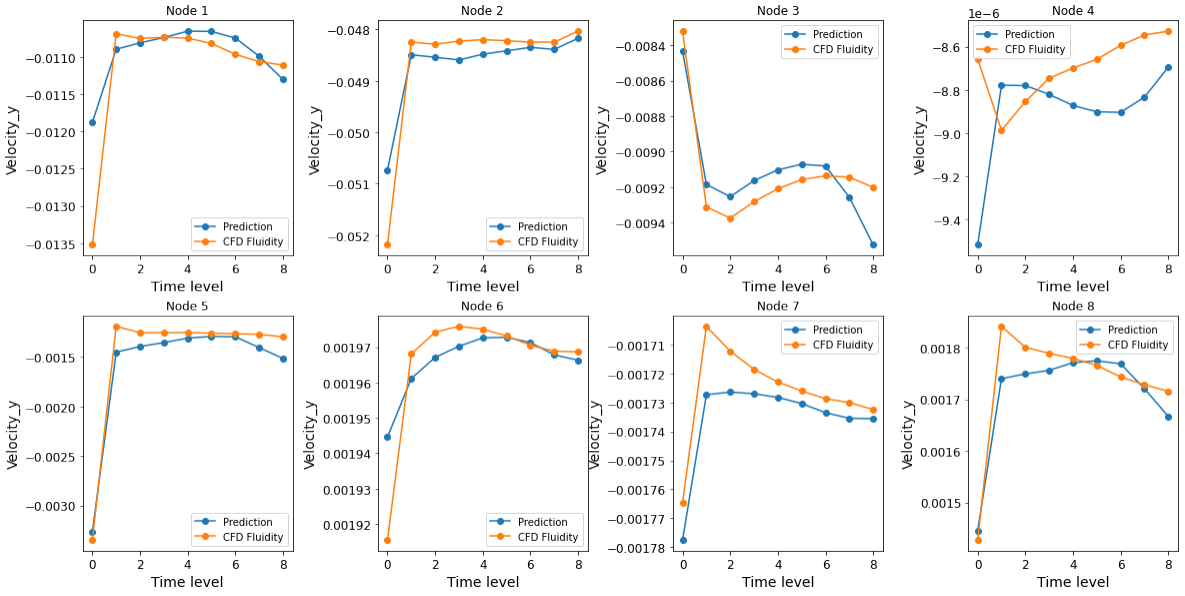
Figure 3: Training plots of SVD-AE with different loss functions and hyper-parameter settings.

The left Figure shows that MSE-based loss function and the right uses proposed loss function (refer to section 2.2) on SVD-AE model. The training turns to be more stable and gain better performance, while the MSE of reconstructed solutions are 102.955 and 20.308 respectively. The *EarlyStopping* is used during the training, this results in the iteration of the left stops earlier than the right one as its validation loss (orange line) stops drop.

Figure 4 illustrates that the forward prediction of the concentration of CO_2 and velocity in one time level. Each plot in the subplot represents the selected nodes that are arbitrarily chosen over time. Figure 5(a) shows that the predictions of CO_2 concentration at sensor locations and Fig 5(b) shows that the predictions through the random nodes. Both yellow lines represent the data from Fluidity and the blue are predictions from PredGAN. The result from DA-PredGAN could also refer to Figure 6 as it shows the comparisons of predictions in single point at 7 different sensor locations over time. Then Figure 7 shows that the predictions of level of CO_2 after DA-PredGAN algorithm at sensor locations, while the blue and yellow lines are the result from PredGAN and DA-PredGAN respectively, comparing with the data from Fluidity (green lines) and experiment (red lines). As the room was fully ventilated during the measurement of the data, the concentration of CO_2 kept decreasing.

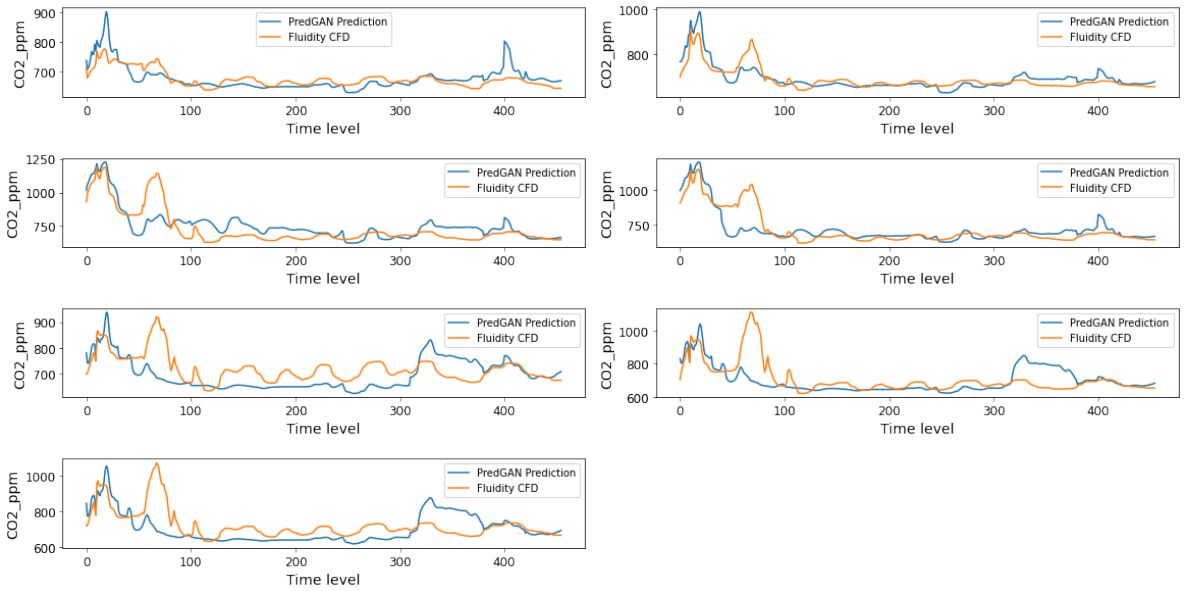


(a) Prediction of CO_2 concentration.

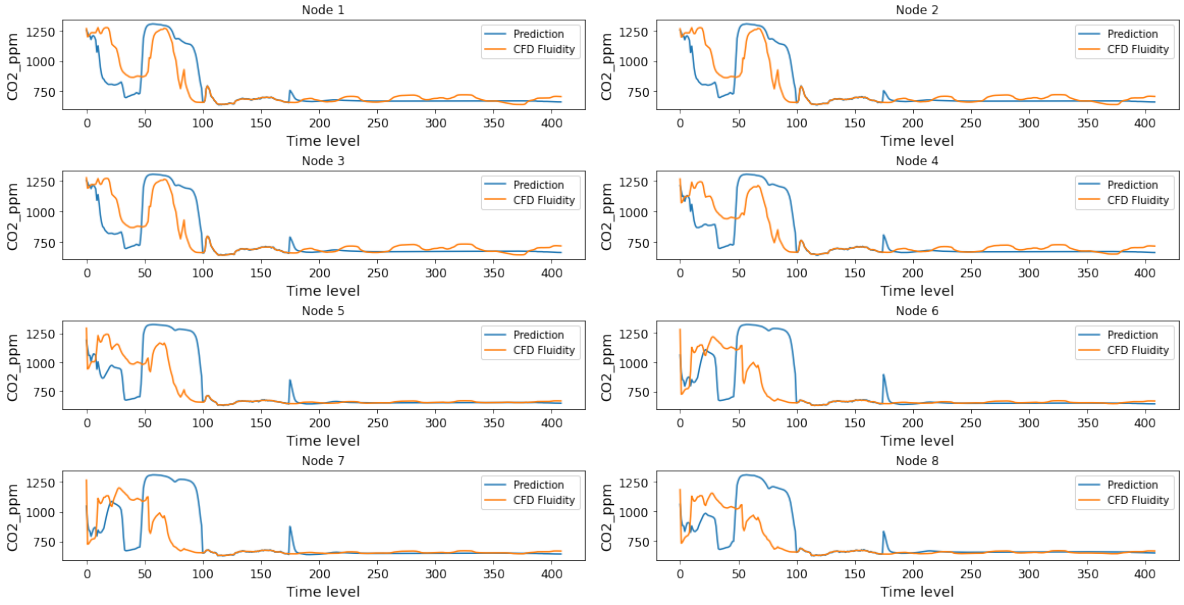


(b) Prediction of velocity in y-axis.

Figure 4: Single point prediction: (a) Forward prediction of in one time level of CO_2 concentration. (b) Prediction of Velocity in y-axis on the same nodes as the level of CO_2. The yellow and blue lines represent data from Fluidity and predictions. Refer to other x and z axis of velocity field in appendices 6.6

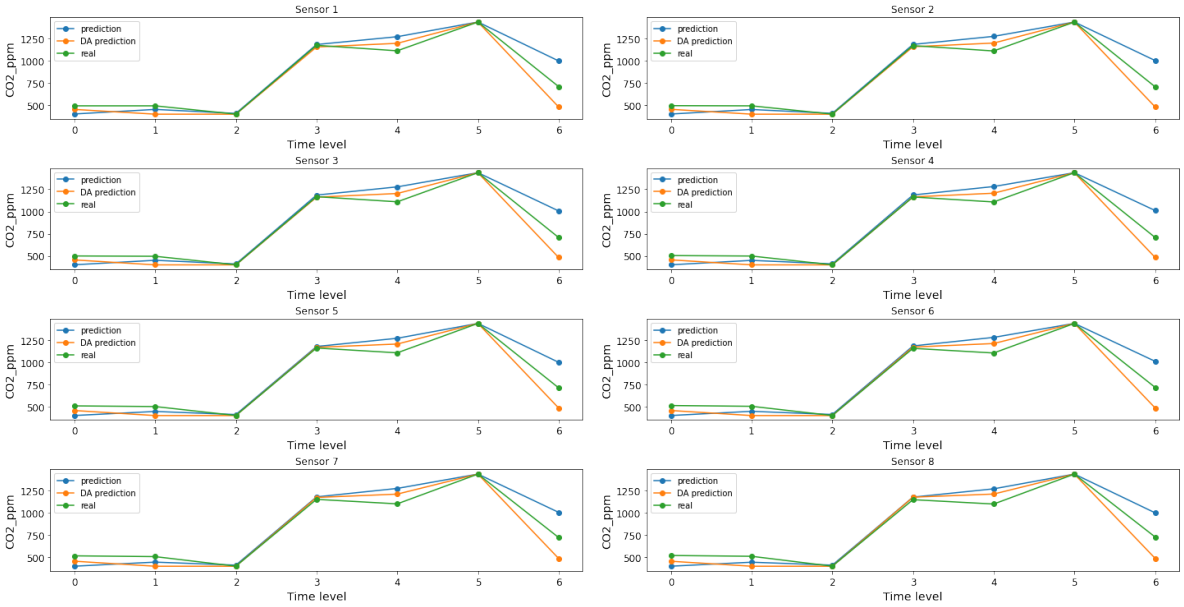


(a) Prediction of CO_2 at sensor locations throughout whole domain.

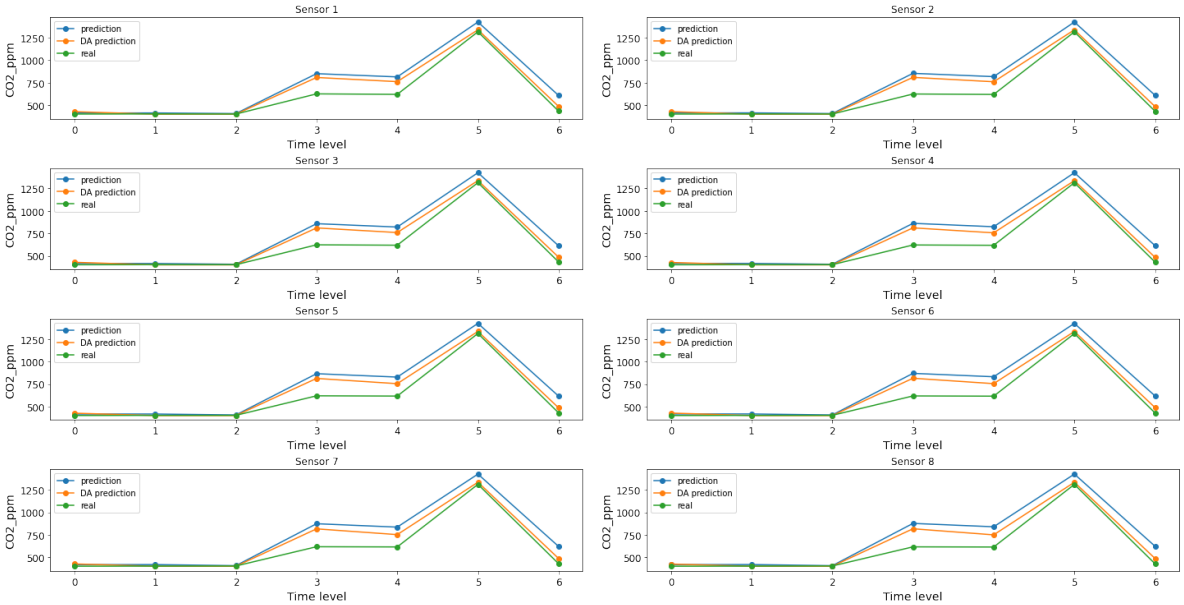


(b) Prediction of CO_2 throughout whole domain.

Figure 5: Predictions from PredGAN of all the time levels at sensor locations and at random selected nodes. (a) shows that the predictions of CO_2 concentration at node 0, 200, 400, 600, 800, 1000, 1200 and 1400 (1 time step $\sim$ 2seconds). (b) shows the predictions of the level of CO_2 at same nodes.



(a) Prediction of CO_2 concentration.



(b) Prediction of CO_2 concentration.

Figure 6: Forward prediction from PredGAN and DA-PredGAN at sensor locations in one time level comparing with data from Fluidity at arbitrary chosen time levels at at node 0, 200, 400, 600, 800, 1000, 1200 and 1400. (b) shows the same predictions but at different nodes at 0, 400, 800, 1200, 1600, 2000, 2400 and 2800.

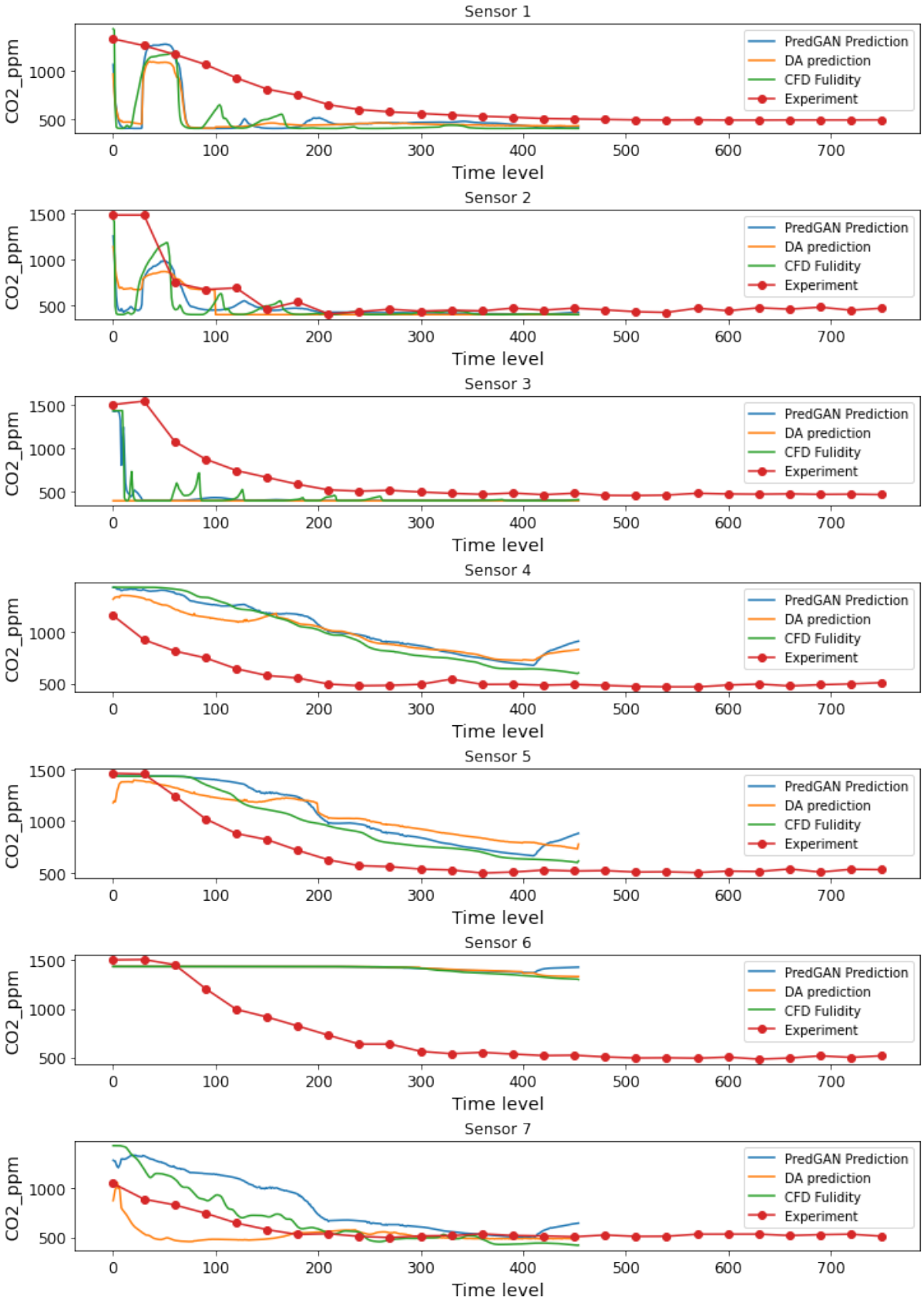


Figure 7: PredGAN and DA-PredGAN predictions through all time levels with experiment data comparing with data from Fluidity at sensor locations. The Fluidity data (green lines) is what expected to predicted.

5 Discussion and Conclusion

This section will mainly discuss and analysis the results from different dimensionality reduction approaches and predictions from PredGAN and DA-PredGAN, then the problems and discoveries of the models based on considerably trial and error will be demonstrated.

5.1 Results

It is exciting to see that the single point predictions of CO_2 concentration are almost indistinguishable from the Fluidity data at sensor locations based on the novel and non-linear compression method SFC-CAE as shown in Figure 14. For further comparison, the velocity data is also reconstructed back to physical space in the x,y and z axis (refer x and z axis of velocity fields in appendices 6.6). As shown in Figure 7, the predictions from PredGAN are promising to predict the concentration of CO_2 at sensor locations.

For the DA-PredGAN simulations, it can be seen that both PredGAN and DA-PredGAN could predict the level of CO_2 accurately in time and DA-PredGAN slightly improved the accuracy of predictions which is shown in Figure 6 as the results are slightly closer to the data from Fluidity comparing with results from PredGAN. However, there are some fluctuations both on the predictions from PredGAN and DA-PredGAN which could be seen in Figures 5 and 7, the overall trends are still matching with the data from Fluidity. We assume that the accuracy of the predictions could be further improved once better settings or models are discovered.

5.2 Dimension reduction

Compression could reduce the computational cost and extract features from data [38]. Different linear and non-linear approaches of dimensionality reduction are investigated based on the given data.

PCA is the most common method used for reducing dimension and it is easy to compute and provide accurate representation in low dimension, which means that it generates accurate low-rank embedding. For SVD, it could yield the optimal rank-k (low-rank) approximation with small reconstruction error and numerically stable for any matrix. However, vectors formed in SVD may lack any meaning which seems to make AE harder to learn the representations of the data when it is combined with neural networks, and it is expensive to compute. Based on trial observations, this approach (SVD-AE) is also easy to be overfitting once the data is insufficient. NMF is also evaluated, as it theoretically provides parts based representation and remains more localised patterns. However, it is time-consuming and computationally intensive for processing if the matrix is large, and the algorithm itself ensures that basis vectors include no negative entries which mean it critically requires strict scaling techniques.

SFC creates a topological mapping that is spatially non-disruptive, maintaining local data correlations [31]. Although the SFCs-CAE approach is computationally intensive (4.37 hours), it is promising to provide better representations in lower dimensions comparing with the other methods. We provide latent variables of 16 in shared google link. The representations of latent variables are visualised by using t-distributed Stochastic Neighbor Embedding (t-SNE) [50] (refer to appendices 6.7). As it shows in the Figure, the pattern of representations in SFCs-CAE latent variables are monotonously comparing with the others.

For the SVD-AE part, it seems that it is difficult for SVD-AE to learn the representations of data (after applying SVD) by training the given amount of data (includes 455 time levels), in particular, the training data set is even smaller after splitting it to validation and test data sets as the ratio 8:1:1. The training loss for the model in test mode based on the different number of training data sets have been evaluated in the code, we found that the model could be beneficial from sufficient data set by having capability of learning the representative of the problem. As proposed in papers [40, 23], not only the performance of the models will gain great benefit from enough data, but also

more measurements e.g. more test and validation data set could be added into the design without the concern of peaking in its performance. The proposed methods are promising to reduce the dimension based on flow past cylinder data which has 1000 time levels with fewer nodes (20550) and we believe if more data is provided, this approach could also achieve better results on dimensionality reduction.

5.3 Thoughts on SFCs-CAE and hyper-parameters

In order to find the optimised models, there are few interesting points that had been found based on trial observations. Intuitively, we thought the monotonous representations in the latent variables from SFC-CAE (refer to appendices 6.7) are less challenging for neural networks to learn. Hypothetically, we think GAN may find it is difficult to extract the meaningful patterns from the latent variables and the discriminator seems that it could easily 'trick' generator to produce unrealistic results as the generator will also easily 'satisfy' for the results produced, which brings the difficulty of choosing the reasonable models and the ratio of training parameters between generator and discriminator since the model is relatively harder to train comparing with the model based on POD. Digging into more explanations on this, we think the data distribution of latent variables are mainly distributed in higher values, that could dominate the calculation of the distance and slow down the gradient descent in the network.

For the side of hyper-parameters tuning, it has been observed in practice that increasing the batch size will decrease the fluctuation and lead to more stable training no matter in SVD-AE or SFC-CAE models, however, the training loss will also increase. It seems that it is because the larger batch sizes will lead to larger gradient steps comparing with the smaller batch sizes as the training function is tended to converge to sharp minimisers. The losing accuracy when the larger batch size is used also could be compensated by increasing learning rate, which means that the models are capable of finding faraway solutions during the training. It seems that this effect has generalisation in AE-based models.

5.4 PredGAN and DA-PredGAN

Based on the considerably trial and error, there are problems that have been discovered and optimisations that may benefit the models. The batch normalisation (BN) is added into the GAN training for reducing the computational cost and stabilising the learning process by standardising the inputs to the layer for each (mini)-batch. It is worth noting that BN is transformed immediately before the non-linear activation function as suggested in [22]. However, the recent code written by Christian applies *Relu* [32] before BN. We acknowledge that it might not have generality for every model, but based on the trial observations, the model putting BN before activation function outperforms the after. This occurs because the input data is distributed evenly over the domain of activation function after BN, resulting in a more balanced utilisation of all areas.

The PredGAN model is data dependency and its architecture needs to be changed if compressed variables are different. Generally, there is no rule of thumb of choosing the ratio of training parameters in generator and discriminator, while the performance of generator that is used to predict the latent variables are hard to evaluate as the optimisation process for PredGAN model is unstable, not well-understood, and it requires a very careful balance between the generator and discriminator. In additionally, loss value of the generator and discriminator is not promising to be the indicator of the performance of G. For example, G and D are not capable of generating output and distinguishing at first as G evolves with D, after a large number of epochs, G could also be promising of producing output even if G 'fools' D just 10% of the time, while the graph of loss D and G might not look reasonable intuitively. The inception scores [5] could be implemented in future work as the indicator of the performance of PredGAN.

For the DA-PredGAN part, the advantages of DA-PredGAN are that it provides the possibility of

converging faster, although it is only with a limited number of forward-backward iterations as shown in the code. And it achieves slightly better results comparing with PredGAN (refer to Figure 6 in section 4) with no further simulations in high-fidelity model. However, the main disadvantage of DA-PredGAN is that it requires careful tuning on the parameters ζ_{obs} and Λ_u^k in the loss functions in equation 7, while the value of Λ_u^k is hard to interpret. We believe that the chosen settings of both values could generally provide promising results.

5.5 Conclusion and future work

In the work presented within the project, the proposed loss function seems to show the great promise to make training more stable and decrease the training loss. And the success of applying SFC-CAE brings the chance of using non-linear dimensionality reduction method combining with novel PredGAN with enhanced training in the large data set. Furthermore, the developed generative adversarial networks are capable of making predictions spatio-temporally of CO_2 concentrations in an enclosed room and it shows the effectiveness of assimilating the data based on the PredGAN. Its superior capabilities of predicting and observing data at sensor locations brings the opportunity to model viral infection fluid flow without further numerical simulation.

Due to the sensitivity of hyper-parameters in DCGANs, one of the future work will find the better models of PredGAN (e.g. WGAN [4]) that combines with SFC-CAE as dimensionality reduction approach. Also, the dual twin experiments based on the experimental data for validation will also be interesting to see in the future. Once the potential better model is selected, the measurement for performance of generator like inception scores [5] could be implemented, instead of analysing the performance in the prediction which generally requires more computational load.

References

[1] 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.

[2] Xiaohu Guo Adam Candy, Christian T. Jacobs and Jon Hill. fluidity github, url: http://fluidityproject.github.io/.

[3] Maddalena Amendola, Rossella Arcucci, Laetitia Mottet, César Quilodrán Casas, Shiwei Fan, Christopher C. Pain, Paul Linden, and Yi-Ke Guo. Data assimilation in the latent space of a neural network. *CoRR*, abs/2012.12056, 2020.

[4] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

[5] Shane Barratt and Rishi Sharma. A note on the inception score, 2018.

[6] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[7] Jason Brownlee. A gentle introduction to generative adversarial networks (gans), 2019.

[8] César Quilodrán Casas, Vinicius Santos Silva, Rossella Arcucci, Claire E. Heaney, Yike Guo, and Christopher C. Pain. Digital twins based on bidirectional LSTM and GAN for modelling COVID-19. *CoRR*, abs/2102.02664, 2021.

[9] Andrzej Cichocki and Anh-Huy Phan. Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE Transactions*, 92-A:708–721, 03 2009.

[10] Andrew Collette. *Python and HDF5*. O'Reilly, 2013.

[11] Zhiwei Deng, Rajitha Navarathna, Peter Carr, Stephan Mandt, Yisong Yue, Iain Matthews, and Greg Mori. Factorized variational autoencoders for modeling audience reactions to movies, 2017.

[12] Timothy Dozat. Incorporating nesterov momentum into adam. 2016.

[13] Hamidreza Eivazi, Hadi Veisi, Mohammad Hossein Naderi, and Vahid Esfahanian. Deep neural networks for nonlinear model order reduction of unsteady flows. *Physics of Fluids*, 32(10):105104, Oct 2020.

[14] G.Buonanno, L.Stabile, and L.Morawska. Estimation of airborne viral emission: Quanta emission rate of SARS-CoV-2 for infection risk assessment, 2020.

[15] Francisco J. Gonzalez and Maciej Balajewicz. Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems, 2018.

[16] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[17] Xiaoxiao Guo, Wei Li, and Francesco Iorio. Convolutional Neural Networks for Steady Flow Approximation. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[18] Claire E. Heaney, Yuling Li, Omar K. Matar, and Christopher C. Pain. Applying convolutional neural networks to data on unstructured meshes with space-filling curves, 2021.

[19] Jan S. Hesthaven and Anthony T. Patera. Reduced basis approximation and a posteriori error estimation for parametrized partial differential equations, 2010.

[20] I. Higgins, Loïc Matthey, A. Pal, Christopher P. Burgess, Xavier Glorot, M. Botvinick, S. Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework, 2017.

[21] G. E. Hinton. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[22] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[23] A.K. Jain and B. Chandrasekaran. Dimensionality and sample size considerations in pattern recognition practice. In *Classification Pattern Recognition and Reduction of Dimensionality*, volume 2 of *Handbook of Statistics*, pages 835–855. Elsevier, 1982.

[24] Johns Hopkins University (JHU). Covid 19 dashboard. `https://gisanddata.maps.arcgis.com/apps/dashboards/bda7594740fd40299423467b48e9ecf6`, 2021. Accessed: 2021-06-20.

[25] Song Jiyun, S. Fan, W. Lin, L. Mottet, H. Woodward, M. Davies Wykes, R. Arcucci, D. Xiao, J.-E. Debay, H. ApSimon, E. Aristodemou, D. Birch, M. Carpentieri, F. Fang, M. Herzog, G. R. Hunt, R. L. Jones, C. Pain, D. Pavlidis, A. G. Robins, C. A. Short, and P. F. Linden. Natural ventilation in cities: the implications of fluid mechanics. *Building Research & Information*, 46(8):809–828, 2018.

[26] I.T. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

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

[28] S. Kullback and R. A. Leibler. On Information and Sufficiency, 1951.

[29] Y. Li, G. M. Leung, J. W. Tang, X. Yang, C.Y. H. Chao, J. Z. Lin, J. W. Lu, P. V. Nielsen, J. Niu, H. Qian, A. C. Sleigh, H-J. J. Su, J. Sundell, T. W. Wong, and P. L. Yuen. Role of ventilation in airborne transmission of infectious agents in the built environment - a multidisciplinary systematic review. *Pubmed.gov*, 17(1):2–18, 2007.

[30] Ariana Mendible, Steven L. Brunton, Aleksandr Y. Aravkin, Wes Lowrie, and J. Nathan Kutz. Dimensionality reduction and reduced-order modeling for traveling wave physics. *Theoretical and Computational Fluid Dynamics*, 34(4):385–400, May 2020.

[31] Baback Moghaddam, Kenneth J. Hintz, and Clayton V. Stewart. Space-filling curves for image compression. In Firooz A. Sadjadi, editor, *Automatic Object Recognition*, volume 1471 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pages 414–421, August 1991.

[32] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In Johannes Fürnkranz and Thorsten Joachims, editors, *ICML*, pages 807–814. Omnipress, 2010.

[33] World Health Organization. Coronavirus. `https://www.who.int/health-topics/coronavirus#tab=tab_1`, 2020. Accessed: 2021-06-20.

[34] Christopher Pain. an overview of inhale, magic consortia and other our work on indoor flows by laetitia.

[35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[36] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[37] Zhe Peng and Jose L. Jimenez. Exhaled co2 as a covid-19 infection risk proxy for different indoor environments and activities. *Environmental Science & Technology Letters*, 8(5):392–397, 2021.

[38] Toby R. F. Phillips, Claire E. Heaney, Paul N. Smith, and Christopher C. Pain. An autoencoder-based reduced-order model for eigenvalue problems with application to neutron diffusion. *International Journal for Numerical Methods in Engineering*, 2021.

[39] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016.

[40] S.J. Raudys and A.K. Jain. Small sample size effects in statistical pattern recognition: recommendations for practitioners. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):252–264, 1991.

[41] S. Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction, 2011.

[42] Ian Ross. Nonlinear dimensionality reduction methods in climate data analysis, 2009.

[43] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning Internal Representations by Error Propagation, 1985.

[44] Nico Schlömer. meshio.

[45] Vinicius.L.S Silva. Gan for time series prediction, data assimilation and uncertainty quantification. `https://github.com/viluiz/gan`, 2021.

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

[47] P Smolensky. Reflections on cognition and parallel distributed processing. *Parallel Distributed Processing*, 1:194–281, 1986.

[48] Gabor Tajnafoi, Rossella Arcucci, Laetitia Mottet, Carolanne Vouriot, Miguel Molina-Solana, Christopher Pain, and Yi-Ke Guo. Variational gaussian process for optimal sensor placement. *Applications of Mathematics*, 66(2):287–317, 2021.

[49] Jonathan Tompson, Kristofer Schlachter, Pablo Sprechmann, and Ken Perlin. Accelerating eulerian fluid simulation with convolutional networks, 2017.

[50] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[51] Pascal Vincent, H. Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders, 2008.

[52] W F Wells. Airborne contagion and air hygiene: An ecological study of droplet infections. *Journal of the American Medical Association*, 159(1):90, 1955.

[53] Shengjia Zhao, Jiaming Song, and Stefano Ermon. Infovae: Information maximizing variational autoencoders, 2018.

6 Appendices

6.1 Quanta

It is worth mentioning that the information that could be utilised and benefited by engineers to model the dispersion of diseases through the air in indoor environments. Wells proposed the statistical concept of 'quanta' in 1955, which is defined as the number of pathogens that cause a person to reach the least amount of disease, and he stated that the dose of airborne droplet nuclei required to cause infection in Poisson distribution 63.2% ($1-e^{-1}$) [52]. Based on the concept aforementioned, quanta emission rate (QR_E) is frequently introduced in [14], to adopt in infection risk models demonstrating its function by evaluating how many people infected [14]. The suggested method in [14] is important since it is a vital tool for use in enclosed spaces, and it may assist experts in the management of indoor environments during epidemics simply by knowing the viral load, rather than waiting for the end of the outbreak.

6.2 AE

Before knowing autoencoder, the Kullback-Leibler divergence (also known as relative entropy) has to be introduced. It is introduced by Kullback et al [28] in 1951, to measure of how one probability distribution is different from the other one. It could determine which distribution retains more information about the original data distribution.

AE has an encoder $f(x)$ and decoder $g(w)$:

$$f(x) = w \tag{8}$$
$$g(w) = x' \tag{9}$$

where $x, x' \in R^n$, $w \in R^t$ and $t < n$ in $f(\cdot)$ and $g(\cdot)$ are non-linear neural networks.

It also means that AE find a latent representation z that is of lower dimensionality by compressing the input data x to find.

6.3 GAN

The loss function introduced by Goodfellow et al.[16], could be derived from the formula of binary of cross-entropy loss, which could be formulated as:

$$L(\hat{y}, y) = y \log \hat{y} + (1-y)\, \log(1-\hat{y}) \tag{10}$$

where y and $\hat{y}$ are defined as original data and reconstructed data.

While the discriminator D is trained, the label of data coming from original data distribution $P_{data}(x)$ is $y = 1$ (real) and $\hat{y} = D(x)$, on the other hand, the label is $y = 0$ (fake) and $\hat{y} = D(G\,(z)$, and therefore:

$$L(D(x), 1) = \log(D(x)) L(D(G(z)), 0) = \log(1 - D(G(z))) \tag{11}$$

Now, it is straightforward to minimise and maximised the Equation mentioned above for the discriminator D and generator G respectively:

$$L^D = \max[\log\,(D\,(x)) + \log\,(1 - D\,(G(z)))] \tag{12}$$

$$L^G = \min[\log\,(D\,(x)) + \log\,(1 - D\,(G(z)))] \tag{13}$$

Combining Eqs (12) and (13), the equation as proposed in paper [16] could be derived:

$$\min_G \max_D V(D,G) = \min_G \max_D (E_{x\sim P_{data(x)}}[\log(D(x))] + E_{z\sim P_{z(z)}} \log(1 - D(G(z)))) \tag{14}$$

It is worth noting that the expectation of the equation above should be taken if the entire dataset is considered.

6.4 Room

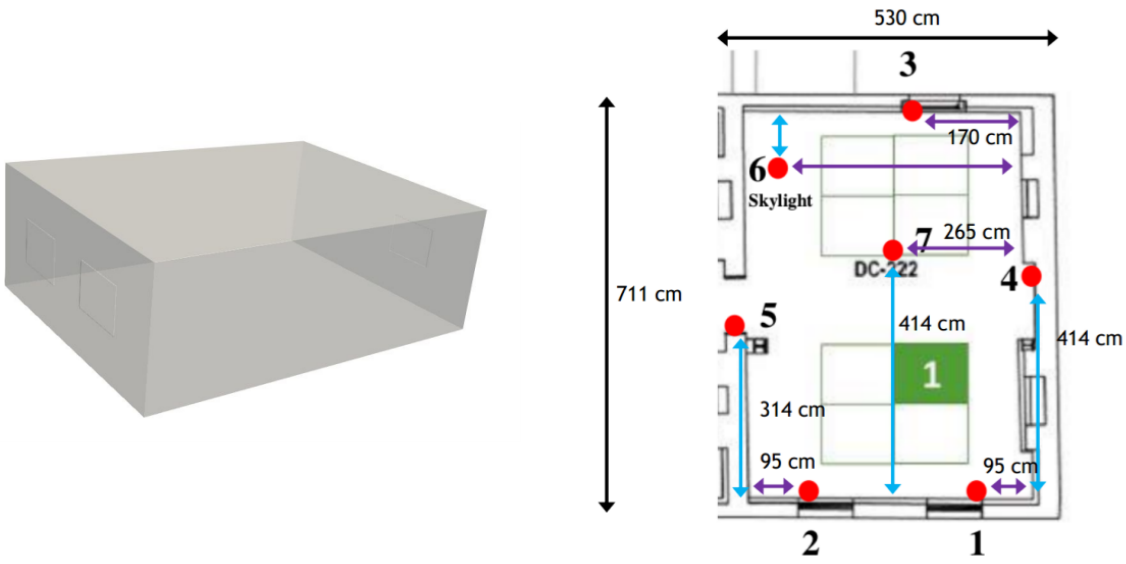


Figure 8: Room of the data has been collected.

The room where the data has been collected is in the Clarence Centre building at London South Bank University (LSBU) in Elephant and Castle in London, UK. Figure above shows three windows in the room, one is facing onto the courtyard, and the other two on the opposite facing London Road. In January 2018, the MAGIC project (http: /www.magic.org) used this place to perform a one-day field investigation [48].

6.5 SFC-CAE

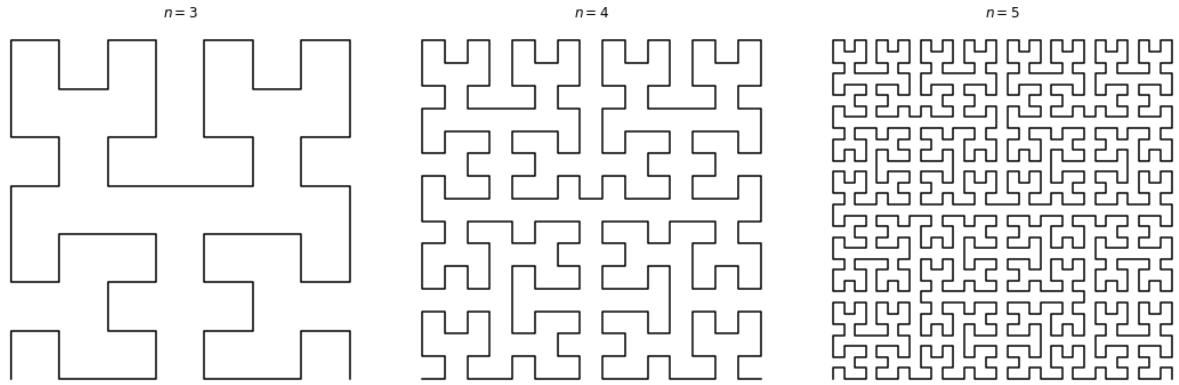


Figure 9: 3 iterations of the Hilbert curve construction

The Figure above is made by the code in the *SFC-CAE compression.ipynb* on GitHub.

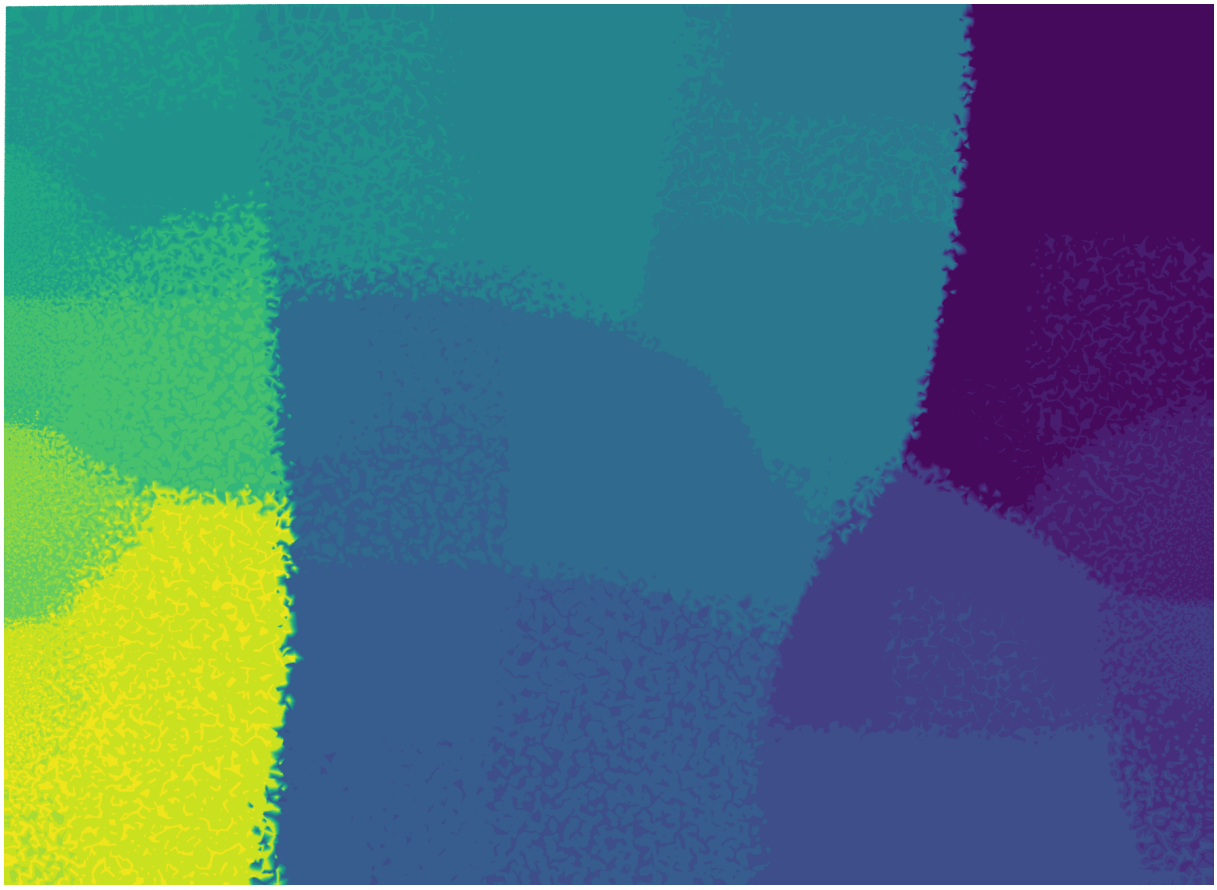
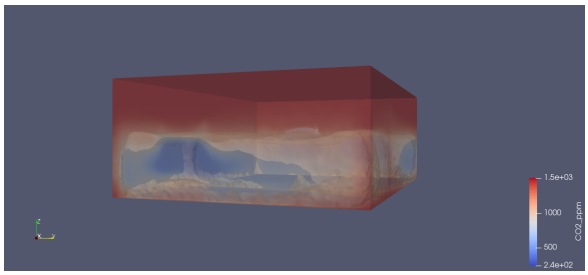


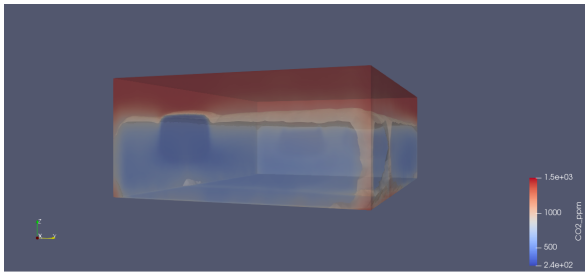
Figure 10: contour plot of 2D unstructured mesh based on a coordinate sequence at 20 levels of colormap.

Figure 10 shows that the contour of a 2D unstructured mesh based on a coordinate sequence. It has 20 levels of colormap for the plot.

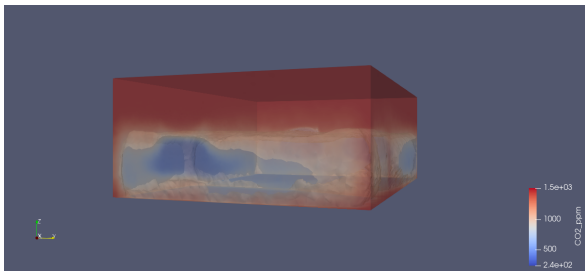
6.6 Results



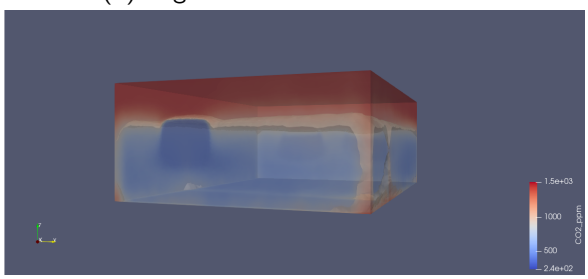
(a) original data at time level 200.



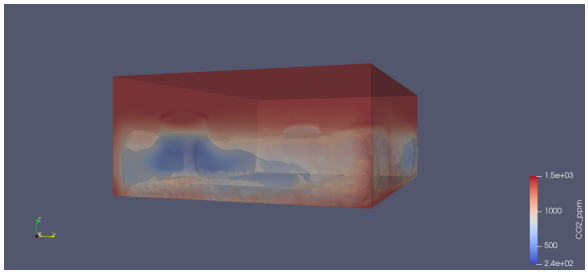
(b) original data at time level 400.



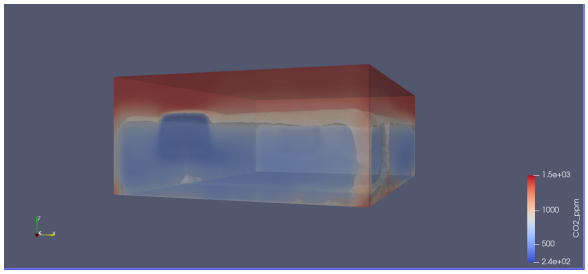
(c) SFCs-CAE at time level 200.



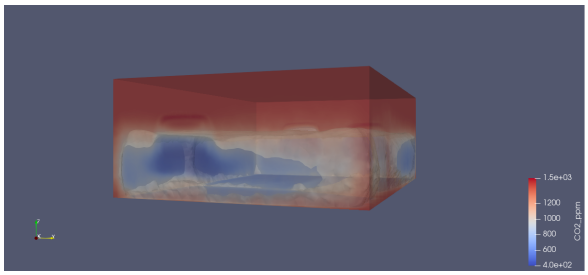
(d) PCA at time level 400.



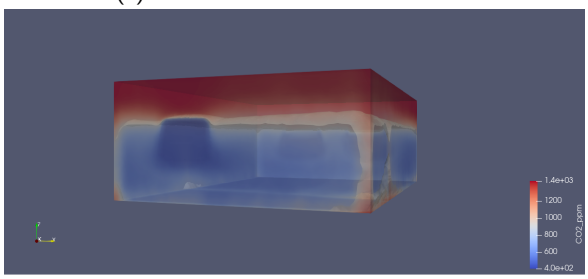
(e) PCA at time level 200.



(f) SFCs-CAE at time level 400.

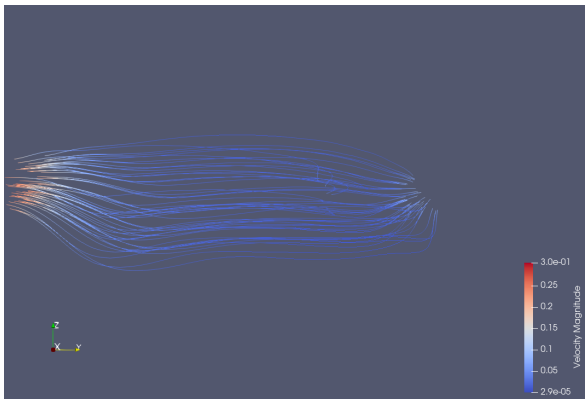


(g) NMF at time level 200.

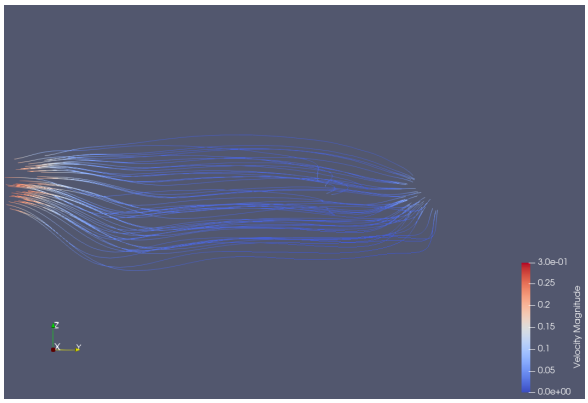


(h) NMF at time level 400.

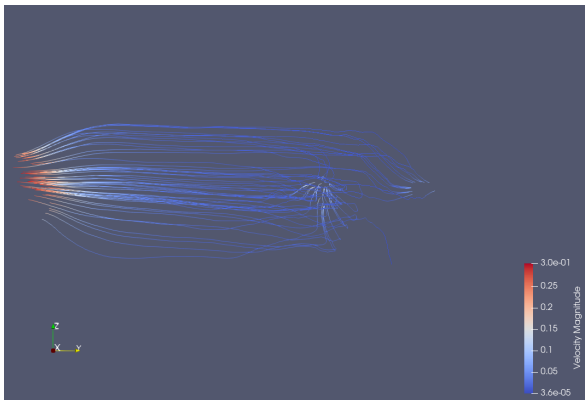
Figure 11: original and reconstruction data sets from SFCs-CAE, PCA and NMF at time level 200 and 400 (1 time step ∼2seconds): (a) original data at time level 200 (b) Original data at time level 200 (c) SFCs-CAE at time level 200 (d) SFCs-CAE at time level 200 (e) PCA at time level 200 (f) SFCs-CAE at time level 400 (g) NMF at time level 200 (h) NMF at time level 400.



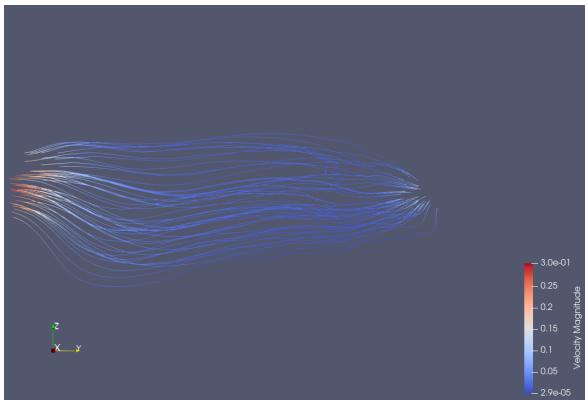
(a) original data at 200



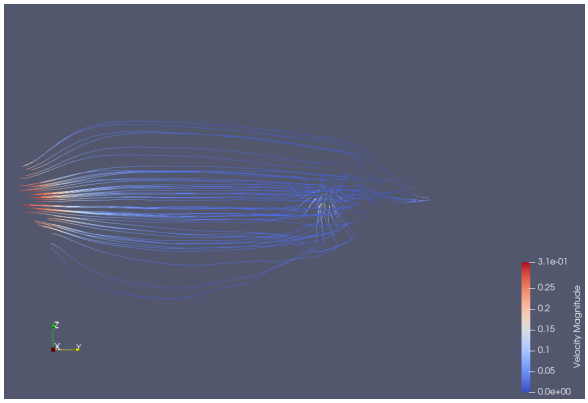
(b) original data at 400



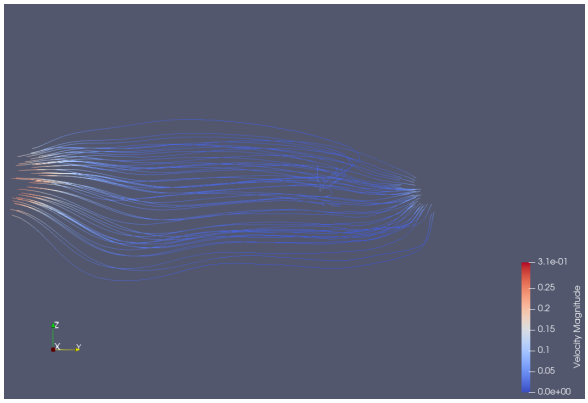
(c) PCA at 200



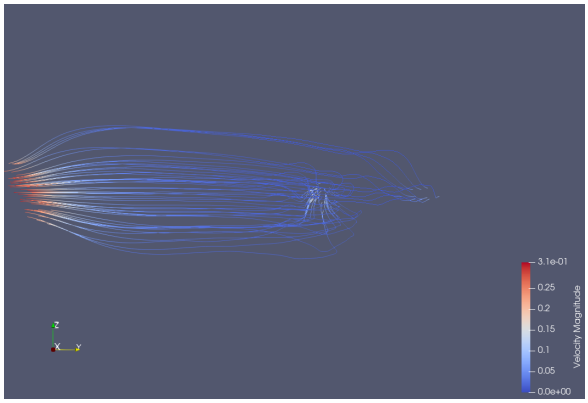
(d) PCA at 400



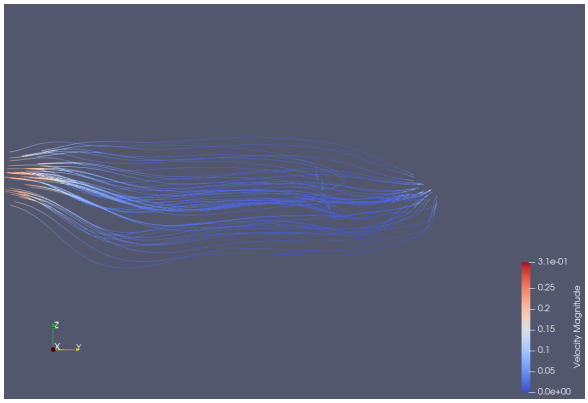
(e) SFCs-CAE at 200



(f) SFCs-CAE at time level 400



(g) NMF at 200

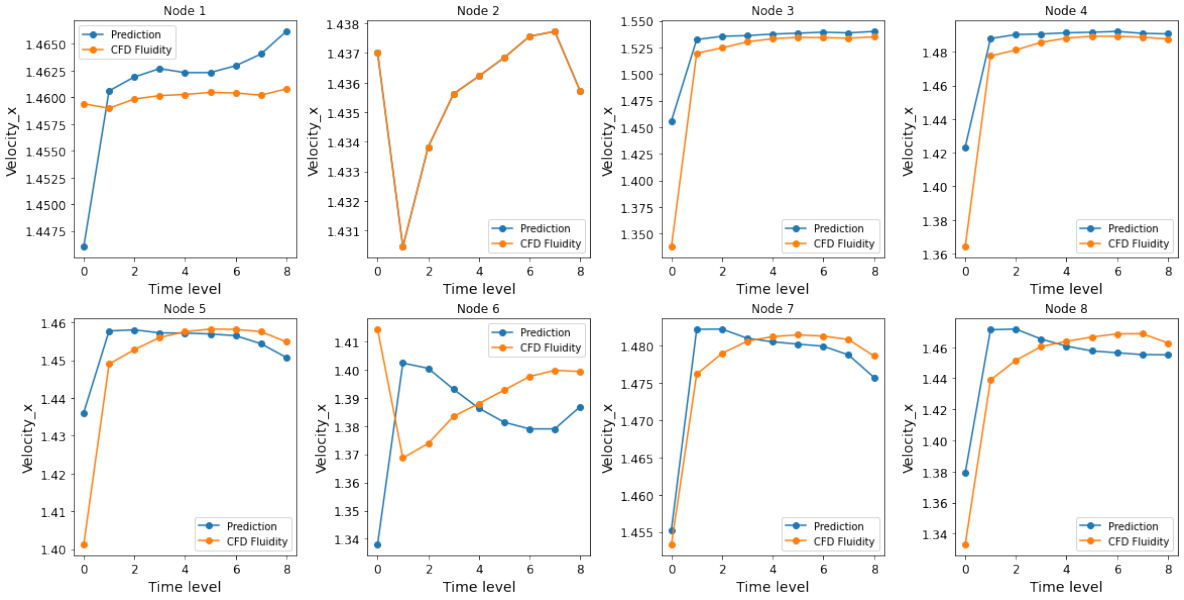


(h) NMF at 400

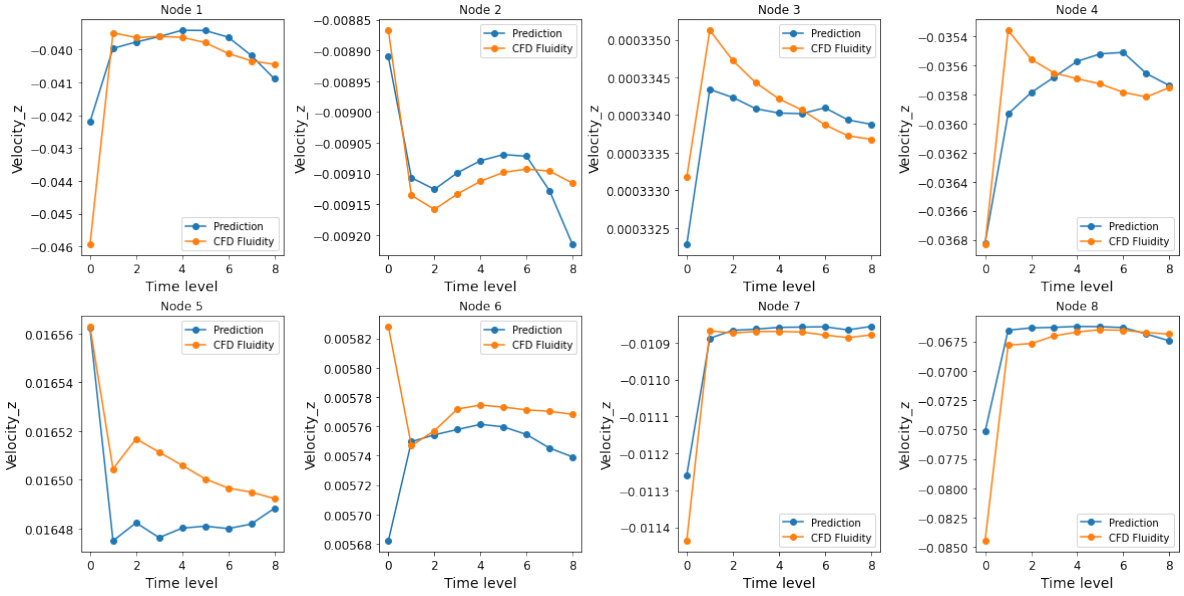
Figure 12: Original and reconstruction data set from SFCs-CAE, PCA and NMF that are visualised in ParaView: (a) Original data at time level 200 (b) Original data at time level 200 (c) SFCs-CAE at time level 200 (d) SFCs-CAE at time level 200 (e) PCA at time level 200 (f) SFCs-CAE at time level 400 (g) NMF at time level 200 (h) NMF at time level 400.

The reconstruction to physical space above in velocity fields were under following settings in ParaView:

seed type is point Cloud without showing sphere, maximum streamline length is set to 5 with 50 number of of points under StreamTracer.



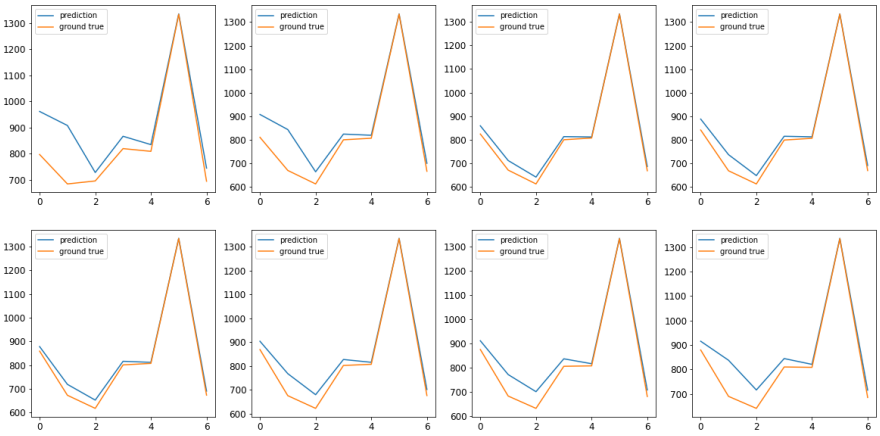
(a) Velocity in x axis



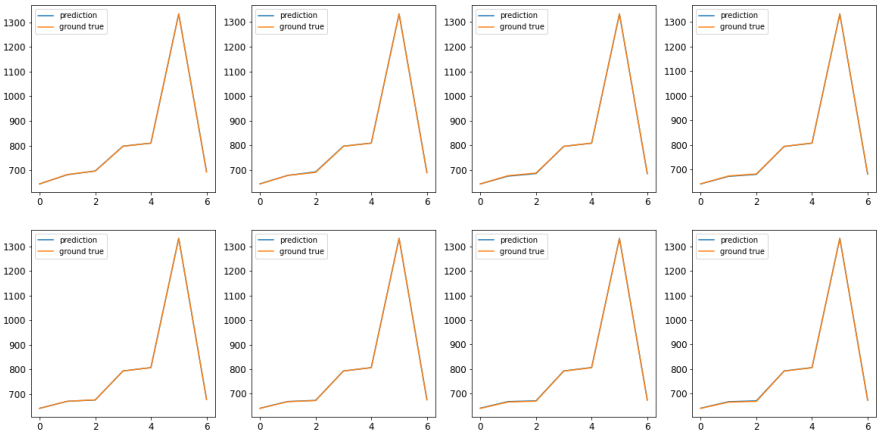
(b) Velocity in z axis

Figure 13: Single point prediction: (a) Forward prediction in one time level of velocity field in x-axis.(b) Forward prediction in one time level of velocity in z-axis on the same nodes as the level of CO2 (refer to Figure 4).

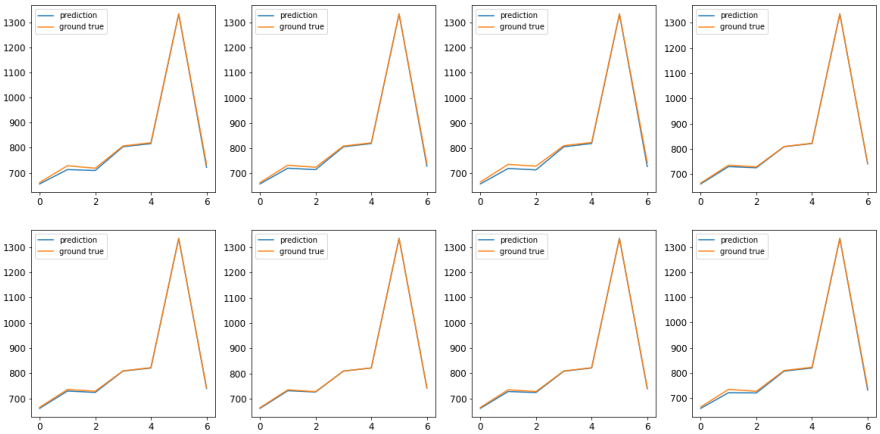
In Figure 14, it shows that the single point predictions at sensor locations, the yellow and blue lines represent the data from Fluidity and predictions from trained PredGAN.



(a) Prediction of CO_2 concentration at time level 0.



(b) Prediction of CO_2 concentration at time level 200.



(c) Prediction of CO_2 concentration at time level 400.

Figure 14: Prediction of CO_2 concentration at sensor locations from PredGAN at different time level (1 time step $\sim$ 2 seconds): (a) 0 (b) 200 (c) 400, at nodes 0, 200, 400, 600, 800, 1000, 1200 and 1400. The yellow lines are the Fluidity data that are expected to predict and the blue lines are the predictions.

6.7 data visualisation

This section illustrates that the data visualisation on the data distribution of Clarence data within the scaling range -1 to 1 (Figure 15), and non-linear connections in latent variables extracted from different dimensionality reduction methods by using t-SNE [50] (Figure 16).

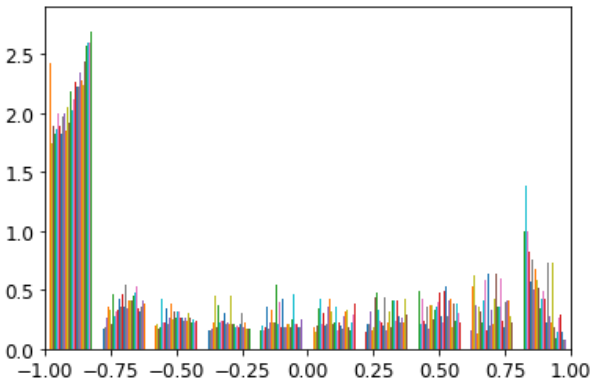
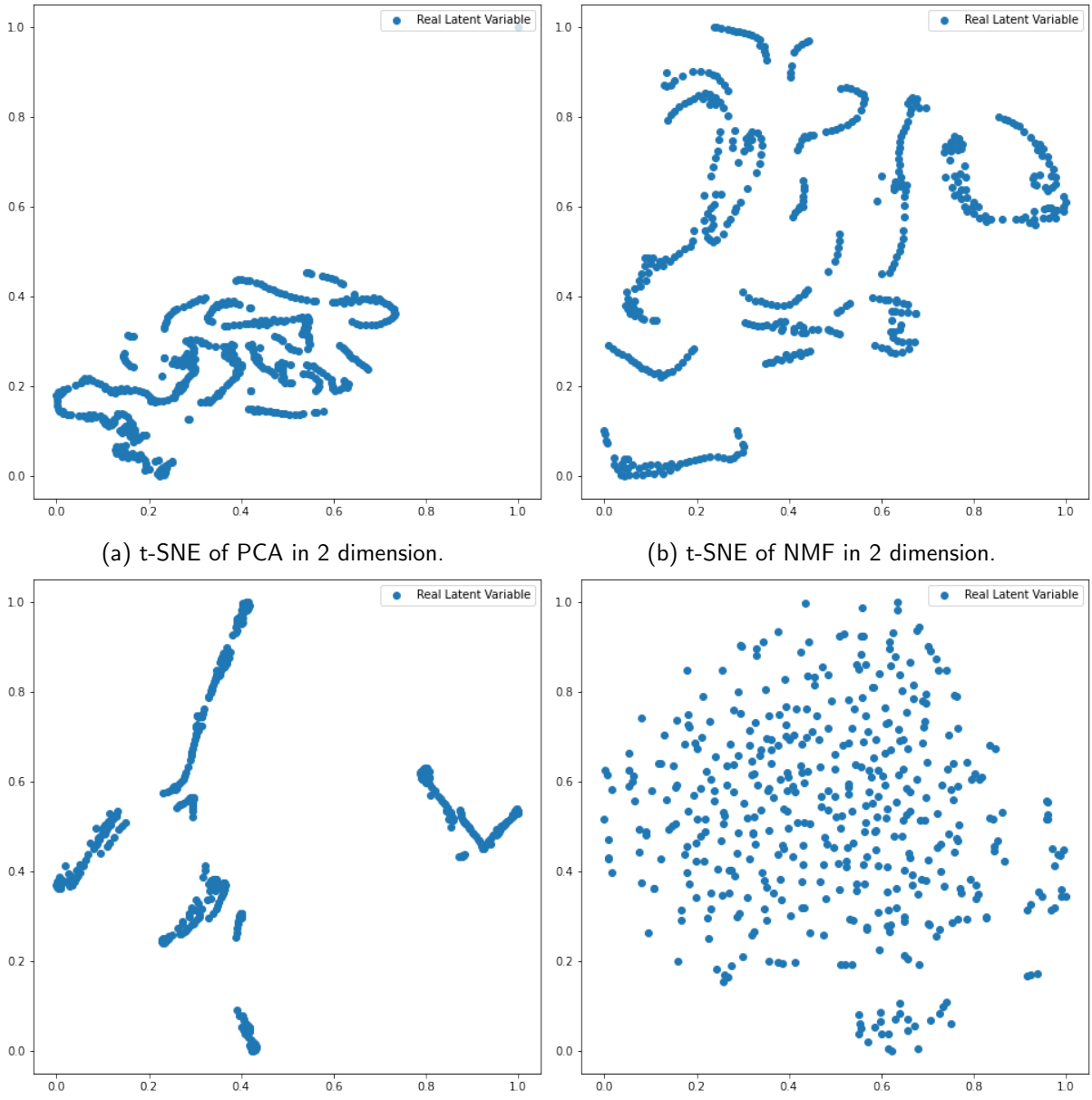


Figure 15: Data distribution of Clarence centre data after scaling, where x and y axis represent the values of scaling within the interval -1 to 1 and the frequency of a particular bin respectively.



(a) t-SNE of PCA in 2 dimension.

(b) t-SNE of NMF in 2 dimension.

(c) t-SNE of SFCs-CAE in 2 dimension.

(d) t-SNE of SFCs-VAE in 2 dimension.

Figure 16: Visualise 43 latent variables from each dimensionality reduction approaches.

As shown in Figure above, it shows that non-linear connections in the latent variables from different compression methods. We also analysis the t-SNE of the Space filling Curves - Variational Autoencoder for future work and implementation [18] in Figure 16(d), which shows beautiful Gaussian shape.

6.8 Mean square error and Conv1d

Mean squared error: `https://scikit-learn.org/stable/modules/model_evaluation.html#mean-squared-error`

Pytorch conv1d layer: `https://pytorch.org/docs/stable/generated/torch.nn.Conv1d.html`

GitHub repository: https://github.com/acse-2020/acse2020-acse9-finalreport-acse-yl2020

6.9 hyper-parameters tuning

The trial observations and hyper-parameters tuning could refer to the the txt files on GitHub repository[1], those files are located at the PredGAN folder. All these models are tuned with different learning rate and batch size.