A CONVOLUTIONAL AUTO-ENCODER FOR REDUCED ORDER MODELS ON UNSTRUCTED MESHES

PROJECT PLAN FOR MSC COURSE ACSE-9

Author : Jiaye Mao
Supervisors : Christopher Pain, Claire Heaney

June 28, 2019

1 Introduction and objective

Due to large dimension size and complexity, it is challenging to solve real-world problems which require large scale dynamic simulations on today's computers. Reduced Order Models (ROMs) offer the potential to increase substantially computational efficiency while maintaining reasonable accuracy [11]. This potential is achieved by reducing the model's associated state space dimension or degrees of freedom.

Reduced order models can be divided into two types: projection-based ROMs and non-intrusive ROMs (NIROMs). Projection-based ROMs project the governing equations onto a subspace spanned by basis functions obtained from the compression of a dataset of solution snapshots [6]. However, for specific applications, this method has been reported to suffer from instability and non-linearity efficiency issues [23]. NIROMs have become popular recently because, as well as addressing these issue, they avoid cumbersome and troublesome modification [26]of the source code for complex dynamical systems. [23]

The dimensionality reduction can lead to a computational acceleration of many orders of magnitude resulting in a model that can be efficient in multi-query problems (such as optimisation) and real-time calculations (such as predictive control). Proper orthogonal decomposition (POD) is one way to reduce the dimensionality, which has been broadly applied in engineering fields, including image processing, signal analysis [14]. However, this method cannot easily be used to construct a non-linear manifold. [6] Deep learning technology has flourished in recent years because of the advancement in computational power, and it is capable of finding more hidden correlation within the data than traditional methods [23]. A further advantage of auto-encoders (one such deep learning method) is their ability to produce non-linear representations that are position invariant.

In this project, a NIROM based on auto-encoders will be implemented to learn an optimal low-dimensional representation. The low dimensional representation is in terms of coordinates on the expressive nonlinear data-supporting manifold within an unstructured mesh finite element fluid model. Auto-encoders to be implemented in this project include a fully-connected auto-encoder and a convolutional auto-encoder, and we will test them on a range of problems making comparison with traditional approaches such as POD and singular value decomposition.

2 Literature Review

2.1 NIROM

The offline stage of the NIROM can be described in 3 steps:

1. solve the high-fidelity model to generate snapshots
2. calculate the POD basis functions and coefficients
3. train the machine learning method on hypersurface prediction using the inputs and outputs

The first step of NIROM is to generate snapshots matrix for the variable of interest (e.g. velocity) from the high-fidelity model (HFM). For the examples studied in this project we will use a finite element code with the ability to solve on

unstructured meshes (Fluidity and IC-FERST)[26]. The columns of the matrix are snapshots of the solution to the HFM taken at certain times.

The second step of NIROM is to calculate the POD basis functions and coefficients. POD, also referred to principal component analysis (PCA)[17], is a method of compressing the snapshots matrix into a small number of basis functions which keep the principle behaviour of the system.[24] It first apply the SVD to the snapshots matrix $S = U\Sigma V^T$ and the POD basis functions are the first N columns of U. If some singular values are relatively small compared to a given tolerance, the size of basis functions can be reduced.

The third step is to train a neural network to generate hypersurfaces of the model that should have similar predictive power as the high-fidelity model. The neural network is able to map the set of POD coefficients from one-time level to the next time level, and so, predict the variables. There are several methods to generate hypersurfaces, for example, Gaussian Process Regression (GPR) and Long Short-Term Memory (LSTM).[26]

Once the offline stage is complete, the online prediction is straightforward. A low-dimensional representation is constructed by the encoder network, and then this data is then fed into the neural network to predict the results. After a certain number of steps, the decoder network allows the user to reconstruct the full-dimensional state.

Wang et al. for the first time used LSTM to construct a non-intrusive reduced order model. The computation time outcome of ocean gyre simulation and flow past a cylinder simulation shows that the CPU time of DLROM is reduced by three orders of magnitude compared to the full model.[23]

Instead of LSTM method, Xiao et al. presented a ROM based on POD and the GPR method, which is six orders of magnitude faster than the high-fidelity model.[26] Xiao et al. also combined domain decomposition and NIROM, improving the capability of the NIROM for large-scale problems.[25]

2.2 Auto-encoder

The main limitation of POD is its linear nature[2] since many data sets include essential nonlinear structures that are invisible to POD[19]. The use of machine learning methods has increased dramatically in the last 15 years due to the increased capability of computers. One method, the auto-encoder, has shown great promise for image compression. For example, using efficient convolutional architecture and other techniques, auto-encoder has achieved better performance than JPEG 2000[20].

Recently, the auto-encoder has been incorporated into a ROM framework as depicted in Figure 1. An auto-encoder is an unsupervised learning algorithm setting the target values to be equal to the inputs that can be trained by applying back-propagation. The network is able to discover correlations that exist inside the input features. It learns a compressed representation of the input and tries to reconstruct the input[16]. The auto-encoder includes adaptive "encoder" network to transform the high-dimensional data into a low-dimensional representation and a similar "decoder" network to recover the data from the low-dimensional representation[7]. Auto-encoders have been used in multiple applications, such as dimensionality reduction, image denoising[22] and image retrieval[10]

An auto-encoder can also be described as a nonlinear generalisation of POD[7], and in fact, POD can be constructed as two-layer auto-encoder, with linear activation functions for its linearly weighted input[1].

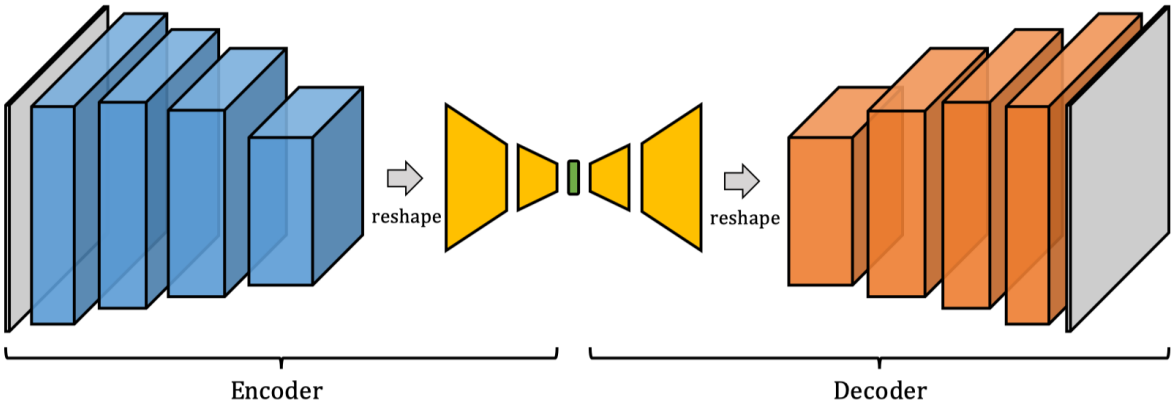


Figure 1: A convolutional auto-encoder example, taken from Gonzalez et al. 2018

Carlberg et al. combine local compression using auto-encoders followed by global compression using POD. It is supposed that global compression is necessary because even if the number of degrees of freedom in each element is reduced, the number of elements in the mesh may still be large. The result showed that the auto-encoder reduced the dimensionality of the element-local state from 320 to 24 without incurring significant errors.[4]

D'Agostino et al. use five hidden layers composed of 160-50-N-50-160 neurons and exponential linear unit (ELU) activation functions for each hidden layer, except for the output layer where a linear activation function is used. For the original (M = 27) design space, a reconstruction error achieved by this deep convolutional auto-encoder is smaller than 5% with 12 variables.[5]

2.3 Convolutional Neural Network

Fully-connected auto-encoders have two drawbacks: (1) The input data must be assembled into a 1D array, and the local spatial relations between values are thus eliminated[6] (2) It becomes computationally intractable for input data of sizes larger than 10^6 since the parameters for the neuron network will be extremely large.

Thus, we will investigate an alternative to fully-connected auto-encoders in which data is structured as multiple arrays. First introduced by LeCun et at. in 1989[13], convolutional auto-encoders are able to address the drawbacks mentioned above, and they have achieved excellent performance in fields such as hand-writing classification or image detection.[27]

The two critical properties of convolutional neural networks are:[12] 1) Local connections: it computes local low-level features in a small subset[15]. 2) Shared weights: the shared nature of each filter bank allows the identification of similar features in the overall input, which leads to a reduction in the trainable parameters.

Feeding in two input signals (one of which is flipped), convolution is the procedure that multiplies and adds together the instantaneous values of the overlapping samples. This procedure will be regarded as 2D convolution if the convolution is performed on two signals having two perpendicular dimensions[3]. The convolution for a 3×3 filter is depicted graphically in Figure 2, in which padding is a margin of zero values which are placed around the image.

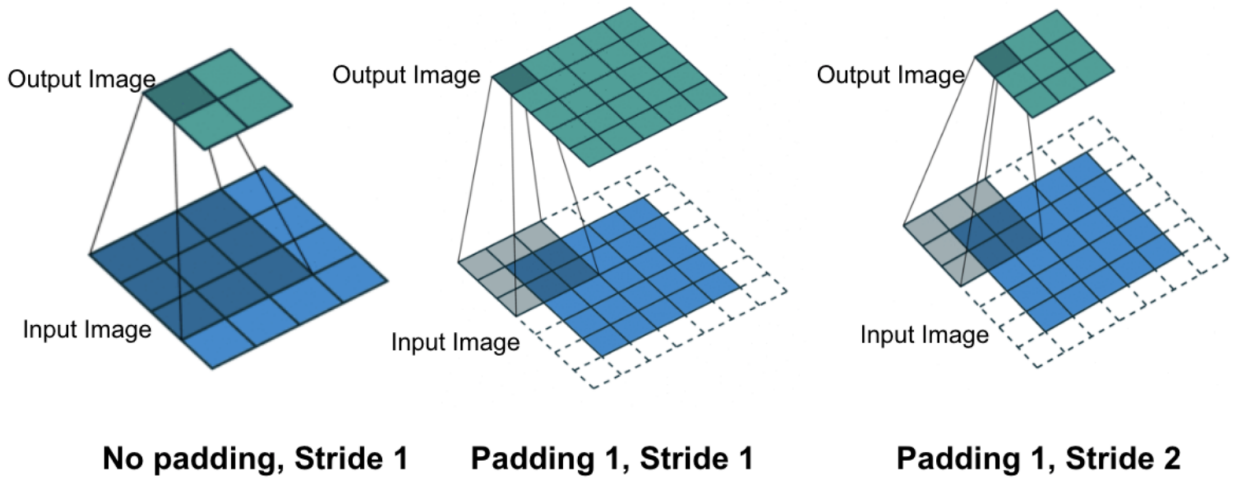


Figure 2: 2D convolution procedure, taken from Murphy 2016.

In image processing, a few filters are set and used to perform a few tasks. For example, if a 3×3 filter has the same weight on each element, then it can be used to blur images. Each pixel's new value is set to a weighted average of that pixel's neighbourhood. Other filters, such as Sobel filters, can give us an activation map that indicates where the edges are in the input.

One example of CNNs called LeNet-5 is depicted in Figure 3 . CNNs are neural networks in which the activation functions are expressed in terms of convolutional kernels or filters[9]. These filters are just small patches that represent visual feature; these are the "weights" and "biases" of a CNN. We take each filter and convolve it over the input volume to get a single activation map. In convolutional neural networks, a layer consists of feature maps, which is calculated by filters in the previous layer. For the 2D output X, a convolutional layer generates feature maps Y by applying 2D convolutions:[6]

$$Y^f_{i,j} = \sum_{k=0}^{a-1} \sum_{l=0}^{b-1} K^f_{a-k,b-l} X_{1+s(i-1)-k,1+s(j-1)-l}$$

where s is integer value called stride, indicating the interval size of filter sliding.

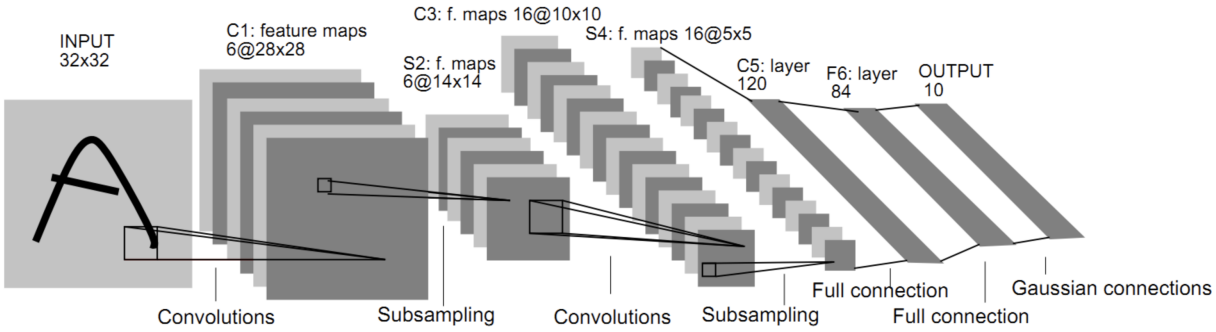


Figure 3: Stucture of LeNet-5, taken from LeCun et al., 1998

Similar to detecting an instance of an object anywhere in an image, a convolutional layer in an autoencoder should be able to capture the large-parameter variations implicitly defined in the initial condition. This feature is called location invariant, and the two properties of CNNs above work to detect this feature. sih[6]

Gonzalez et al. presented a parameter-varying flow in a periodic box to indicate the location-invariance capabilities. The result showed that compared to POD, the recurrent convolutional autoencoder(8 latent vectors) performs well in predicting new solutions using fewer features than for POD, illustrating the effectiveness of deep autoencoder-based approaches to nonlinear model reduction[6].

As a result of location invariance, the components in the low-dimensional representation have a direct meaning, for example, x position or y position. This feature enables the compressed output to be linearly computed. In this project, we hope to reduce the dimension of building matrix to a rank-4 feature map, including two-dimensional size, location and rotation.

Kashima examined the projection error on the reduction of a singularly perturbed system for an auto-encoder and POD. The error for POD is 200 times greater than for the auto-encoder. This result shows when approximating the noise response data, the accuracy of a two-dimensional nonlinear manifold cannot be realised by any two-dimensional linear manifold.[8]

3 Proposed Approach and Future Plan

3.1 Approach

In this project, a NIROM employs auto-encoders will be implemented within an unstructured mesh finite element fluid model. The unstructured mesh is identified by irregular connectivity. If the number of elements in the mesh is N, then it can be expressed as an $N \times N$ sparse matrix, each line of which saves the indices of its connected neighbour. This mesh matrix is stored as a compressed sparse row (CSR) matrix.

There are two steps in performing a convolution on such a mesh.

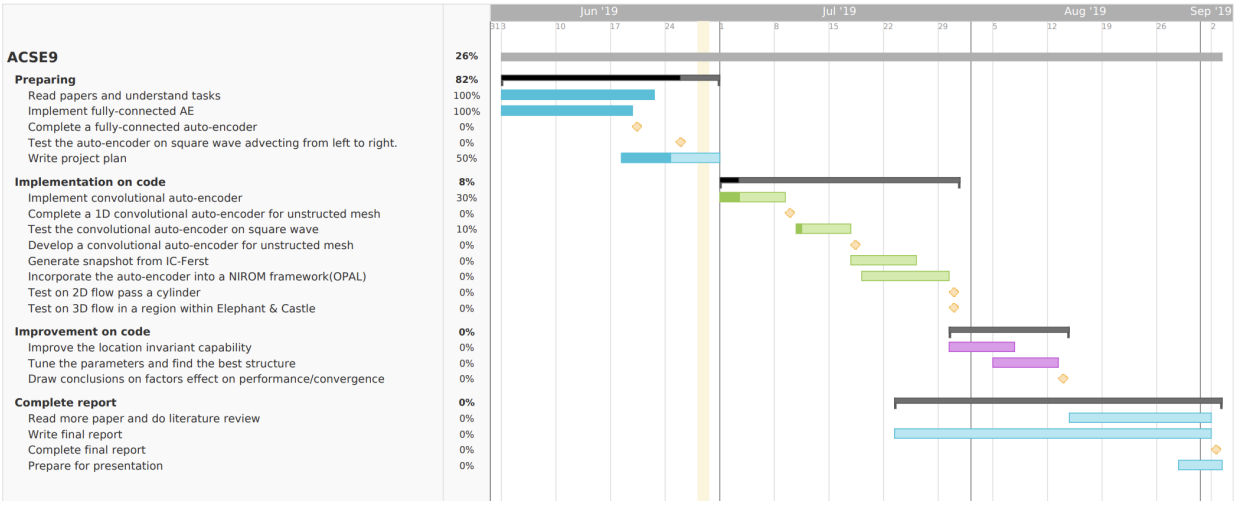
The first step is applying a mesh colouring operation to find the maximum independent set. An independent vertex set of a graph G is the largest subset of the vertices such that no two vertices in the subset represent an edge of G[4]. A mesh colouring operation returns every maximum independent vertex set, and one of the set will be chosen as centres for convolution.

The second step is finding neighbours and convolving them with a filter. Neighbours can be found directly from the sparse matrix, and the filter is a 1D array whose size is the maximum size of neighbours in all centres. There are two key points in the convolution that need to be considered, the connection between centres after the convolution and the sequence of neighbours when doing the convolution. The former problem can be addressed by the number of interval nodes between coarse centres. If the number is equal to 1, then we regard these two coarse centres are close enough to connect them. The latter problem (the sequence of neighbours) is the problem we need to test in this project, and the test case includes clockwise sequence or anticlockwise sequence.

The main difficulty of this action is how to find the best way to do backpropagation. As far as we know, there is no existing library for unstructured mesh propagation, and the unfixed-size of neighbours for the centres will be the hardest point to implement since the program needs to store detailed information for each layer.

Another method to apply convolution on unstructured meshes is interpolation. This method is easier to complete than the former one, which transforms the unstructured matrix into a large-scale sparse matrix. We generate a large-scale matrix overlapping the unstructured mesh and interpolate every element on this matrix [21]. This procedure is included in the library called VTU tools [18]. However, the inverse procedure that transforms the large-scale sparse matrix back to the unstructured matrix is not included in the library. This procedure can be implemented by finding the position in the grid from the values of four corners of this grid.

3.2 Future Plan



Milestone 1 (19th June) : Develop a fully-connected auto-encoder

Milestone 2 (25th June) : Test the auto-encoder on square wave in a 1D advection problem.

Milestone 3 (5th July) : Develop a 1D convolutional auto-encoder for unstructured meshes and test on the square wave

Milestone 4 (15th July) : Develop a convolutional auto-encoder for unstructured meshes and incorporate the auto-encoder into a NIROM framework(using OPAL, a Python library)

Milestone 5 (30th July) : Test on 2D flow past a cylinder and 3D flow in a region within Elephant and Castle

Milestone 6 (13th August): Draw conclusions on how batch size, number of epochs, choice of activation function effect the performance/convergence of the auto-encoder.

Milestone 7 (28th August) : Final report completed

References

[1] Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58, 1989.

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

[3] C Sidney Burrus and TW Parks. *Convolution Algorithms*. Citeseer, 1985.

[4] Kevin T Carlberg, Antony Jameson, Mykel J Kochenderfer, Jeremy Morton, Liqian Peng, and Freddie D Witherden. Recovering missing cfd data for high-order discretizations using deep neural networks and dynamics learning. *Journal of Computational Physics*, 2019.

[5] Danny D'Agostino, Andrea Serani, Emilio F Campana, and Matteo Diez. Deep autoencoder for off-line design-space dimensionality reduction in shape optimization. In *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, page 1648, 2018.

[6] FJ Gonzalez and M Balajewicz. Deep convolutional recurrent autoencoders for learning lowdimensional feature dynamics of fluid systems. *arXiv preprint arXiv:1808.01346*, 2018.

[7] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[8] Kenji Kashima. Nonlinear model reduction by deep autoencoder of noise response data. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 5750–5755. IEEE, 2016.

[9] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[10] Alex Krizhevsky and Geoffrey E Hinton. Using very deep autoencoders for content-based image retrieval. In *ESANN*, 2011.

[11] Toni Lassila, Andrea Manzoni, Alfio Quarteroni, and Gianluigi Rozza. Model order reduction in fluid dynamics: challenges and perspectives. In *Reduced Order Methods for modeling and computational reduction*, pages 235–273. Springer, 2014.

[12] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[13] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[14] YC Liang, HP Lee, SP Lim, WZ Lin, KH Lee, and CG Wu. Proper orthogonal decomposition and its applications—part i: Theory. *Journal of Sound and Vibration*, 252(3):527–544, 2002.

[15] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International Conference on Artificial Neural Networks*, pages 52–59. Springer, 2011.

[16] Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

[17] René Pinnau. Model reduction via proper orthogonal decomposition. In *Model order reduction: theory, research aspects and applications*, pages 95–109. Springer, 2008.

[18] William J Schroeder, Kenneth M Martin, and William E Lorensen. The design and implementation of an object-oriented toolkit for 3d graphics and visualization. In *Proceedings of Seventh Annual IEEE Visualization'96*, pages 93–100. IEEE, 1996.

[19] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[20] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders. *arXiv preprint arXiv:1703.00395*, 2017.

[21] Petr Vanek, Jan Mandel, and Marian Brezina. Algebraic multigrid on unstructured meshes. *University of Colorado at Denver, UCD= CCM Report*, (34), 1994.

[22] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.

[23] Zheng Wang, Dunhui Xiao, Fangxin Fang, Rajesh Govindan, Christopher C Pain, and Yike Guo. Model identification of reduced order fluid dynamics systems using deep learning. *International Journal for Numerical Methods in Fluids*, 86(4):255–268, 2018.

[24] Karen Willcox and Jaime Peraire. Balanced model reduction via the proper orthogonal decomposition. *AIAA Journal*, 40(11):2323–2330, 2002.

[25] D Xiao, CE Heaney, F Fang, L Mottet, R Hu, DA Bistrian, E Aristodemou, IM Navon, and CC Pain. A domain decomposition non-intrusive reduced order model for turbulent flows. *Computers & Fluids*, 182:15–27, 2019.

[26] D Xiao, CE Heaney, L Mottet, F Fang, W Lin, IM Navon, Y Guo, OK Matar, AG Robins, and CC Pain. A reduced order model for turbulent flows in the urban environment using machine learning. *Building and Environment*, 148:323–337, 2019.

[27] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.