9 September 2019

CAEs for 3D-VarDA

MSc Machine Learning

Author: Julian Mack
Supervisor: Dr Arcucci

Project Summary

1. Used an Attention-based Convolutional Autoencoder (CAE)
2. ...to reduce the space of Background Covariance Matrix in 3D Variational Data Assimilation (3D-VarDA).
3. ...extending the work in [Arcucci et al.2019] which used TSVD for the same task.

Project Summary

In comparison with [Arcucci et al.2019] approach:

1. DA error reduced by 37%.
2. Up to x30 faster
3. Compressed representation is $\mathcal{O}(10^3)$ smaller.

Imperial College
London

Performance-Time Tradeoff

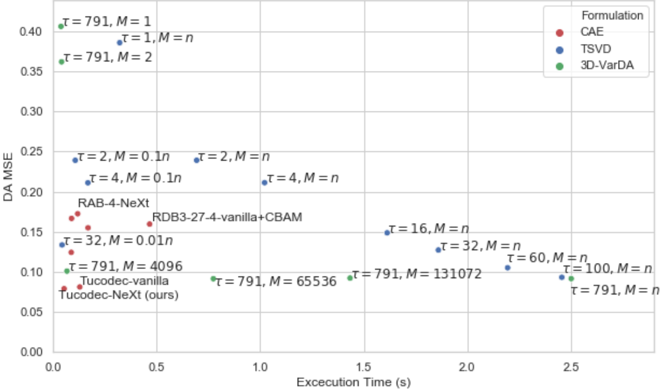

Figure 1: Performance-speed tradeoff for a range of systems.

Imperial College London

Results are *consistently* improved

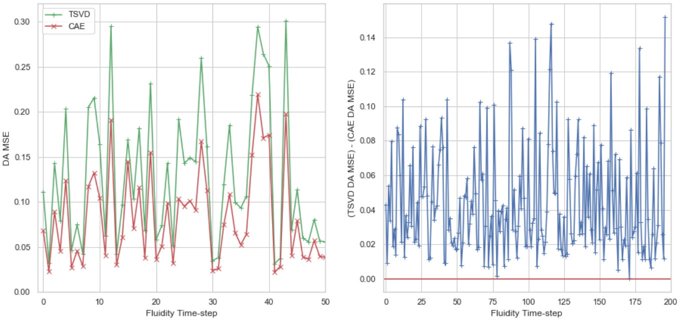

Figure 2: Comparison of TSVD ($\tau = 32$, $M = n$) and CAE data assimilation performance on sequential test-set time-steps.

Imperial College London

Presentation Structure

Imperial College London

Presentation Structure

Imperial College London

MAGIC Project

- 'Managing Air for Green Inner Cities', or MAGIC project.
- An international collaboration to produce models to monitor and control pollution in urban areas.
- One of the project's aims is to produce: "*reduced order models [of inner city fluid flow] that allow rapid calculations for real time analysis and emergency response*".

MAGIC project (continued)

- [Arcucci et al.2019] use TSVD in 3D Variational data assimilation to tackle this problem.
- ...on a MAGIC test-site location in South London
- ...with synthetic data generated by the open-source finite-element fluid dynamic software Fluidity.

We use the same domain and data

...to enable a clear comparison between the approaches

MAGIC project (continued)

We use the same domain and data

...to enable a clear comparison between the approaches

But the new method is **non-intrusive** (required no information about the underlying forecasting model) so it is more widely applicable.

3D-VarDA Definitions

1. $\boldsymbol{x}$: state of model. Vector in $\in \mathbb{R}^n$
2. $\boldsymbol{y}$: observation space of model. Vector in $\in \mathbb{R}^M$
3. $\boldsymbol{x}^b$: background state. Our prior on the model state before seeing any observations. Vector in $\in \mathbb{R}^n$
4. $\boldsymbol{H}$: Observation operator. $\boldsymbol{H} : \mathbb{R}^n \to \mathbb{R}^M$
5. $\boldsymbol{R}$: Observation error covariance matrix in $\in \mathbb{R}^{M \times M}$
6. $\boldsymbol{B}$: Background error covariance matrix in $\in \mathbb{R}^{n \times n}$

Incremental 3D-VarDA

Consider perturbations $\delta \boldsymbol{x} := \boldsymbol{x} - \boldsymbol{x}^b$ to background state.
Cost function minimisation [Courtier et al.1994]:

$$\begin{aligned} \delta \boldsymbol{x}^{DA} &= \arg\min_{\delta \boldsymbol{x}} J(\delta \boldsymbol{x}) \\ J(\delta \boldsymbol{x}) &= \frac{1}{2}\delta \boldsymbol{x}^T \boldsymbol{B}^{-1}\delta \boldsymbol{x} + \frac{1}{2}\left\|\boldsymbol{d} - \boldsymbol{H}\delta \boldsymbol{x}\right\|^2_{\boldsymbol{R}^{-1}} \end{aligned} \tag{1}$$

Misfit

$\boldsymbol{d} = \boldsymbol{y} - \boldsymbol{H}\boldsymbol{x}^b$ is the ‘misfit’ between observations $\boldsymbol{y}$ and prior expectation on observations $\boldsymbol{H}\boldsymbol{x}^b$

CVT

$$J(\delta \boldsymbol{x}) = \frac{1}{2}\delta \boldsymbol{x}^T \boldsymbol{B}^{-1}\delta \boldsymbol{x} + \frac{1}{2}\left\|\boldsymbol{d} - \boldsymbol{H}\delta \boldsymbol{x}\right\|^2_{\boldsymbol{R}^{-1}}$$

[Parrish and Derber1992] use the following Control Variable Transform(CVT):

$$\delta \boldsymbol{x} = \boldsymbol{V}\boldsymbol{w}$$
$$\boldsymbol{B} = \boldsymbol{V}\boldsymbol{V}^T$$

CVT

$$\delta \boldsymbol{x} = \boldsymbol{V}\mathbf{w}$$
$$\boldsymbol{B} = \boldsymbol{V}\boldsymbol{V}^T$$

where $\boldsymbol{V}$ is constructed from mean-centred model outputs $\boldsymbol{X}^b$: which are set aside as "background":

$$\boldsymbol{X}^b = [\boldsymbol{x}_0^b, \boldsymbol{x}_1^b, ..., \boldsymbol{x}_S^b] \in \mathbb{R}^{n \times S}$$
$$\boldsymbol{V} = (\boldsymbol{X}^b - \bar{\boldsymbol{x}}^b) \ \in \mathbb{R}^{n \times S}$$

S, sample size

S is number of samples used to create background prior.

Reduced Space

$$\boldsymbol{V} = \in \mathbb{R}^{n \times S}$$

This introduces a **reduced space** of dimension S:

$$\mathbf{w} \in \mathbb{R}^{S}$$

We are implicitly representing background matrix of size $\mathcal{O}(n^2)$ with matrix with $\mathcal{O}(nS)$ parameters.

CVT continued

This gives cost function:

$$\begin{aligned} \mathbf{w}^{DA} &= \arg\min_{\mathbf{w}} J(\mathbf{w}) \\ J(\mathbf{w}) &= \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{1}{2}\left\|\boldsymbol{d} - \boldsymbol{H}\boldsymbol{V}\mathbf{w}\right\|^2_{\boldsymbol{R}^{-1}} \end{aligned} \tag{2}$$

After finding $\mathbf{w}^{DA}$ by minimisation we can return to the full space with:

$$\boldsymbol{V}\mathbf{w} = \delta\mathbf{x}$$

Imperial College London

Presentation Structure

Latent space

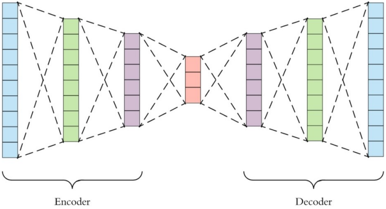

Figure 3: AE framework. An encoder compresses the input (blue) to a smaller latent representation(red).

- We train an encoder $f(\boldsymbol{x})$ to create a **latent space** representation of the inputs.
- This is restored to the full space with decoder $g(\cdot)$.
- This is a lossy process so $g(f(\boldsymbol{x})) \neq \boldsymbol{x}$.

Latent space $\boldsymbol{V}$

$$\boldsymbol{V} = \left[\delta \boldsymbol{x}_0^b,\ \delta \boldsymbol{x}_1^b,\ \dots,\ \delta \boldsymbol{x}_S^b\right] \in \mathbb{R}^{n \times S}$$

We define latent space $\boldsymbol{V}_l$ such that:

$$\begin{aligned} \boldsymbol{V}_l &= f(\boldsymbol{V}) \\ \boldsymbol{V}_l &= \left[f(\delta \boldsymbol{x}_0^b),\ f(\delta \boldsymbol{x}_1^b),\ \dots,\ f(\delta \boldsymbol{x}_S^b)\right] \in \mathbb{R}^{m \times S} \end{aligned}$$

Size of representation

We are representing $\boldsymbol{B}$ with $\boldsymbol{V}_l$ which has $\mathcal{O}(mS)$ parameters.

In our implementation $m \sim 0.001n$ so this is an $\mathcal{O}(10^3)$ reduction in comparison with previous approach.

Proposed Formulation

$$
\begin{aligned}
\mathbf{w}_l^{DA} &= \arg\min_{\mathbf{w}_l} J(\mathbf{w}_l) \\
J(\mathbf{w}_l) &= \frac{1}{2}\mathbf{w}_l^T\mathbf{w}_l + \frac{1}{2}\left\|\boldsymbol{d}_l - \boldsymbol{V}_l\mathbf{w}_l\right\|^2_{\boldsymbol{R}_l^{-1}}
\end{aligned}
\tag{3}
$$

where a subscript l implies the matrix or vector has been replaced by its latent-space equivalent. See report for definitions of $\boldsymbol{d}_l$ and $\boldsymbol{R}_l$.

Bi-reduced Space

Note that there are two 'reduced' spaces in this case:

1. The reduced space of size S introduced by the CVT. The **'reduced space'**.
2. The reduced space of size m introduced by the encoder-decoder framework. The **'latent space'**.

Hence the 'Bi-reduced space' formulation.

Return to full space

$\mathbf{w}_l^{DA}$ can be restored to the full space in a two-stage transformation:

1. Multiplication by $\boldsymbol{V_l}$ to move from the reduced space representation to the latent space.
2. Using the decoder $g(\cdot)$ to move from the latent space to the full space $\in \mathbb{R}^n$.

Return to full space (continued)

Overall this gives:

$$\delta \boldsymbol{x}^{DA} = g(\boldsymbol{V_l} \mathbf{w}_l^{DA})$$

Complexity

The online complexities of the [Parrish and Derber1992] reduced space approach R_{on} is:

$$R_{\text{on}} = \mathcal{O}(I_1 M^2 + nS)$$

where I_1 is the number of iterations in the reduced space VarDA minimisation routine.
And the bi-reduced space approach B_{on} is:

$$B_{\text{on}} = \mathcal{O}(nm)$$

Derivation and discussion in report.

Approaches are [almost] equivalent

- i.e. we show that:

$$\mathbf{w}^{DA} = \mathbf{w}_l^{DA}$$

- ...under three assumptions including that the compression *is* lossless $g(f(\boldsymbol{x})) = \boldsymbol{x}$.
- See report for more details.

Imperial College London

Comparison Summary

Reduced Space 3D-VarDA

$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{1}{2}\left\|\boldsymbol{d} - \boldsymbol{H}\boldsymbol{V}\mathbf{w}\right\|^2_{\boldsymbol{R}^{-1}}$

Return: $\delta\boldsymbol{x}^{DA} = \boldsymbol{V}\mathbf{w}^{DA}$

$\mathrm{R}_{\mathrm{on}} = \mathcal{O}(I_1 M^2 + nS)$

Bi-reduced Space 3D-VarDA

$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{1}{2}\left\|\boldsymbol{d}_l - \boldsymbol{V}_l\mathbf{w}\right\|^2_{\boldsymbol{R}_l^{-1}}$

Return: $\delta\boldsymbol{x}^{DA} = g(\boldsymbol{V}_{\boldsymbol{l}}\mathbf{w}^{DA})$

$\mathrm{B}_{\mathrm{on}} = \mathcal{O}(nm)$

Theory: TSVD vs CAE

A good CAE should produce higher quality compression of $\boldsymbol{V}$ since:

1. Redundancies in the training data distribution can be stored 'for free' by the decoder.
2. CAEs take location data into account – can use properties like local smoothness to compress more efficiently.
3. CAEs use non-linear combinations of the inputs – likely to be of greater expressive quality than those in SVD.
4. In *Truncated* SVD, some of the information is intentionally discarded. This is not true in the CAE framework.

Presentation Structure

Imperial College London

CAE Design

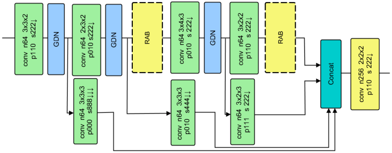

Figure 4: The Tucodec encoder of [Zhou et al.2019].

- We only see benefits of proposed approach with a **high-quality** CAE.
- We defined CAE specification and argue that the **Image Compression literature** is most relevant parallel use-case.

CLIC

- CLIC: 'Challenges on Learned Image Compression'. A workshop to find SOTA in lossy image compression (runs annually as part of CVPR).
- CLIC-2019 was in June so we had an up-to-date comparison of available approaches.
- I reviewed $\sim 65+$ ML papers using CLIC as a starting point.

I implemented and evaluated every **top-5** CLIC-2018 and CLIC-2019 entry

...as well as number of related architectures.

Implementation challenge

- The MAGIC problem domain is 3D but CLIC focuses on 2D image compression.
- It was non-trivial to extend many architectures to three spatial input dimensions as default channel and kernel sizes gave unreasonably large feature maps or had very high computational cost.
- To mitigate this, we made a number of alterations to architectures including replacing all large convolutional kernels with 3x3 convolutions.

CAE Building Blocks

Most CLIC architectures had some of the following components:

- Parallel convolutional filters.
- GDN activation functions.
- Multi-scale learning.
- Attention.
- Complex residual blocks: e.g. CBAMs or RABs.

Parallel Convolutional Filters

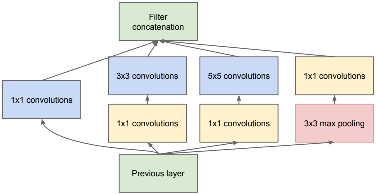

Figure 5: The Inception module [Szegedy et al.2015]. Three convolutional and one pooling operation are performed in parallel.

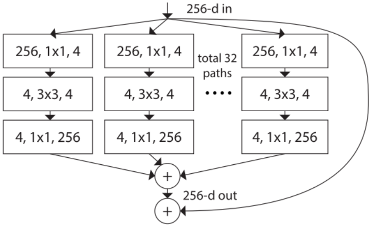

Figure 6: ResNeXt residual block in which the 'cardinality', or width can be increased to boost the capacity of the network [Xie et al.2017].

GDNs

Generalised Divisive Normalisation transformations or GDNs [Ballé et al.2015] normalise and Gaussianize the inputs on the assumption that they are drawn from a very general probability distribution. See report for definition.

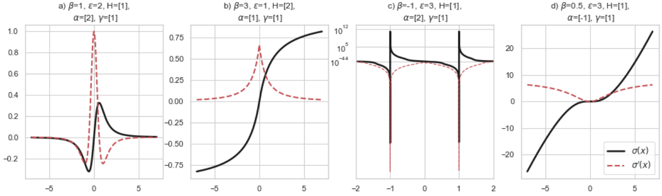

Figure 7: A range of GDN transforms.

Multi-scale Learning

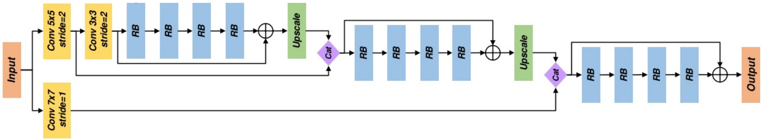

Figure 8: The decoder of Lu et al.'s fourth place entry to CLIC 2019 [Lu et al.2019] in which features are available at multiple resolutions simultaneous. ‘RB’ stands for residual block.

Attention

Figure 9: The effect of attention for classification on ImageNet using the Grad-CAM method for visualisation [Selvaraju et al.2017][1]. Image adapted from the CBAM paper [Woo et al.].

Residual Blocks

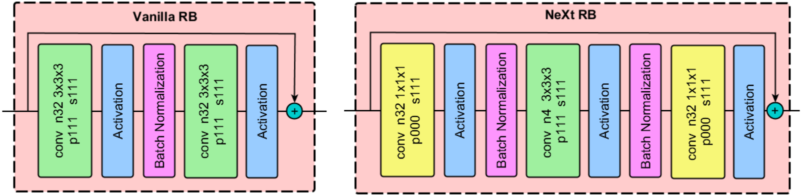

Figure 10: The two basic RBs we evaluated.

NeXt RBs use 1x1 convolutions to bottleneck the inputs

...and therefore have fewer parameters. In the example above:
NeXt: 2k params
vanilla: 60k params.

CBAMs

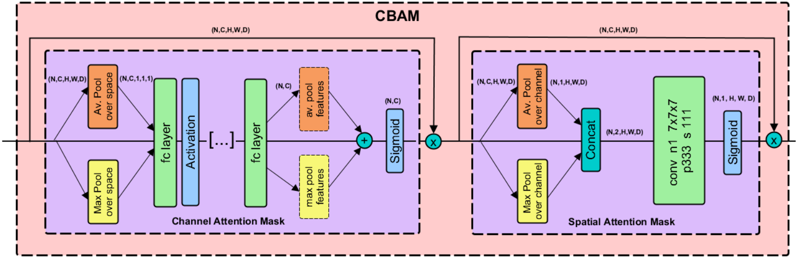

Figure 11: Convolutional Block Attention Modules, or CBAMs, [Woo et al.] are residual blocks that add channel-wise and spatial attention sequentially.

Imperial College
London

RABs

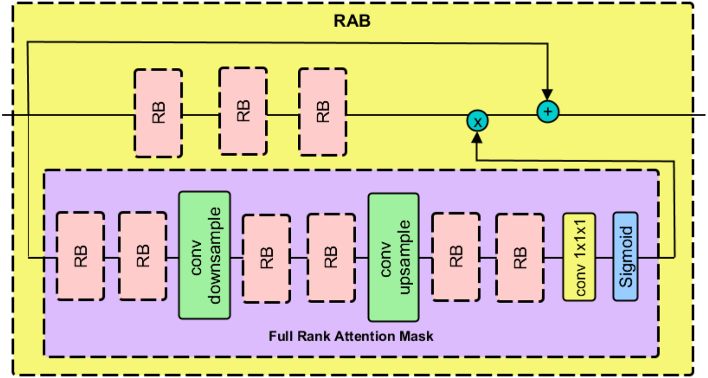

Figure 12: Residual Attention Blocks proposed by [Zhang et al.2019] and utilised by [Zhou et al.2019].

Architecture Search

We investigated our pool of CLIC models with two [main] changes:

1. Activation Functions (x3):
   - GDNs
   - PReLUs
   - ReLUs
2. Residual Blocks (x4):
   - Vanilla RBs
   - NeXt RBs
   - Vanilla + CBAM
   - NeXt + CBAM

Imperial College London

Architecture Summary

Model	DA MSE	Execution Time (s)	Number of Parameters
Backbone	0.1665	0.0897	0.3M
ResNeXt3-27-1-vanilla+CBAM	0.1548	0.1693	3.5M
RDB3-27-4-vanilla+CBAM	0.1594	0.4666	25.6M
RAB-4-NeXt	0.1723	0.1192	1.3M
GRDN-NeXt+CBAM	0.1241	0.0983	4.7M
Tucodec-NeXt	**0.0787**	**0.0537**	2.5M
Tucodec-vanilla	0.0809	0.1294	10.6M

Table 1: A comparison of the DA MSE and inference speeds of a selection of our best models.

Tucodec Model

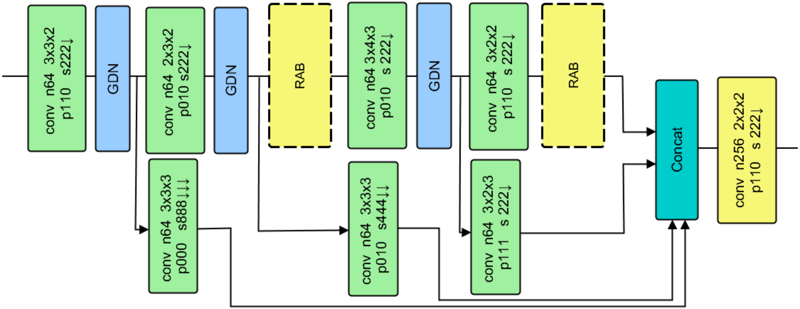

Figure 13: The Tucodec encoder of [Zhou et al.2019]. The decoder did not have the multi-scale path but was otherwise symmetric.

Imperial College London

Presentation Structure

Imperial College London

Effect of M

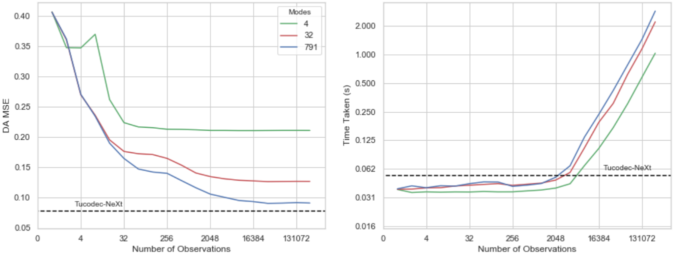

Figure 14: Effect of number of observations M on a) DA MSE and b) execution time. Tucodec values are marked with dashed black lines.

Comparison with [Arcucci et al.2019]

Model	DA MSE	Excecution Time (s)
Ref MSE	1.0001	-
TSVD, $\tau = 32$, $M = n$	0.1270	1.8597
TSVD, $\tau = 32$, $M = 0.1n$	0.1270	0.2627
TSVD, $\tau = 32$, $M = 0.01n$	0.1334	0.0443
TSVD, $\tau = 32$, $M = 0.001n$	0.1680	**0.0390**
Tucodec-NeXt	**0.0787**	0.0537

Table 2: Comparison of our best Tucodec model with the [Arcucci et al.2019] approach which sets $\sigma_\tau = \sqrt{\sigma_1} = 32$. Our DA MSE is 37% lower than the best performing Arcucci et al. system.

Comparison with [Parrish and Derber1992]

Model	DA MSE	Excecution Time (s)
Ref MSE	1.0001	-
TSVD, $\tau = 791$, $M = n$	0.09132	2.5009
TSVD, $\tau = 791$, $M = 0.1n$	0.0923	0.3182
TSVD, $\tau = 791$, $M = 0.01n$	0.1046	0.0481
TSVD, $\tau = 791$, $M = 0.001n$	0.1403	**0.0410**
Tucodec-NeXt	**0.0787**	0.0537

Table 3: Comparison of our best Tucodec model with the [Parrish and Derber1992] approach in which there is no truncation of $\boldsymbol{V}$. Our DA MSE is 14% lower than the best performing system.

Performance-Speed Tradeoff

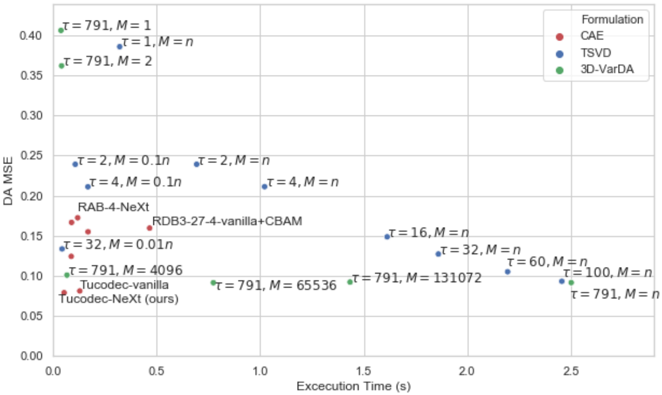

Figure 15: Performance-speed tradeoff for a range of systems.

Acceleration Methods

The quoted figures likely **underestimate** the relative speed of the proposed approach.

This is because the proposed method can be accelerated:

1. With hardware (V100, Graphcore IPU, FPGA implementation).
2. With algorithmic acceleration methods such as:
   - Quantization
   - Pruning (i.e. structured or unstructured sparsity)
   - Thinner Decoder (suggested in [Theis et al.2017])
   - Factorised Convolutions [Wang et al.2018]
   - Pixel Shuffle [Shi et al.2016]

Imperial College London

Importance of Architecture

- The results here demonstrate the central importance of using state-of-the-art CAE architectures.
- This field is moving exceptionally fast: our Backbone network, was state-of-the-art for image compression in 2017 [Theis et al.2017] but gives a DA MSE that is:
    1. Double that of the Tucodec models.
    2. $\sim 30\%$ worse than [Arcucci et al.2019]'s approach with $\tau = 32$ and $M = 0.01$.

Imperial College London

Presentation Structure

Future Work: DA

Apply to approach to:

- 4D-VarDA.
- Other data-sets.

Investigate:

- Integrating with a ROM that learns underlying physics.
- Integrating with data assimilation Localization techniques [Montmerle et al.2018].
- Alternative minimisation routines (used L-BFGS here).

Future Work: ML

Investigate other AE variants including:

- VAEs
- GAN-CAEs

Or investigate acceleration techniques discussed previously:

- Quantization
- Pruning (i.e. structured or unstructured sparsity)
- Thinner Decoder (suggested in [Theis et al.2017])
- Factorised Convolutions [Wang et al.2018]
- Pixel Shuffle [Shi et al.2016]

Imperial College London

Presentation Structure

Contribution Summary 1/3

Proposed a new 'Bi-reduced space' 3D-VarDA formulation that is [approximately] equivalent to [Parrish and Derber1992]'s approach.

- New formulation has lower complexity than [Parrish and Derber1992] or [Arcucci et al.2019] and does not penalise dense sensor networks (i.e large M).
- We evaluate the success of our approach experimentally – it has lower DA MSE in all cases and is faster except for when M is small.

Imperial College London

Contribution Summary 2/3

Implemented and evaluated a range of SOTA CAEs

- We find that [Zhou et al.2019]'s 'Tucodec' attention-based model performs best.
- We perform an extensive architecture search and find that we can reduce Tucodec decoder inference latency by almost x2.5 by replacing vanilla RBs [He et al.2016] with 'NeXt' RBs [Xie et al.2017].
- To our knowledge, we are the first to extend the image compression network of [Zhou et al.2019] and image restoration GRDN of [Kim et al.2019] to three-dimensions.

Contribution Summary 3/3

We release a well tested open-source Python module `VarDACAE`[2] that enables users to:

- Easily replicate our experiments,
- Use our model implementations, and
- Train CAEs for any variational data assimilation problem.

[2] The repository can be found at `https://github.com/julianmack/Data_Assimilation`.

Any Questions?

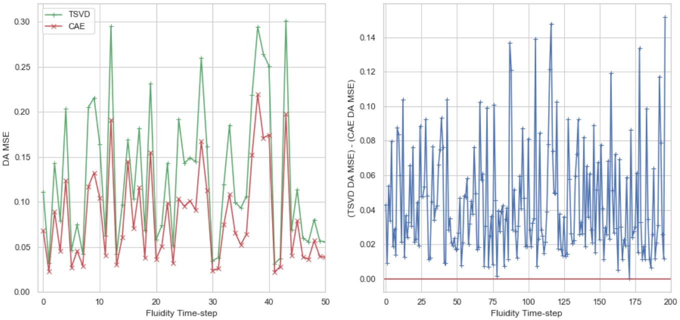

Figure 16: Comparison of TSVD ($\tau = 32$, $M = n$) and CAE data assimilation performance on sequential test-set time-steps.

Imperial College London

Observation encoder

If there is time – observation encoder f^o:

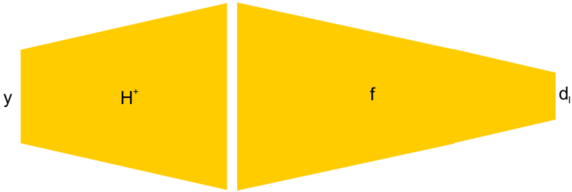

Figure 17: Scheme to create the f^o operator. It should be possible to add the H^+ convolutional network and fine-tune the weights of f to create f^o.

References

Arcucci, R., Mottet, L., Pain, C., and Guo, Y. K. (2019).
Optimal reduced space for Variational Data Assimilation.
Journal of Computational Physics, 379:51–69.

Ballé, J., Laparra, V., and Simoncelli, E. P. (2015).
Density Modeling of Images using a Generalized Normalization Transformation.
pages 1–14.

Courtier, P., Thépaut, J.-N., and Hollingsworth, A. (1994).
A strategy for operational implementation of 4D-Var, using an incremental approach.
Quarterly Journal of the Royal Meteorological Society, 120(519):1367–1387.

He, K., Zhang, X., Ren, S., and Sun, J. (2016).
Deep residual learning for image recognition.
Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016-Decem:770–778.

Kim, D.-W., Chung, J. R., and Jung, S.-W. (2019).
GRDN:Grouped Residual Dense Network for Real Image Denoising and GAN-based Real-world Noise Modeling.

Lu, M., Chen, T., Liu, H., and Ma, Z. (2019).
Learned Image Restoration for VVC Intra Coding.
pages 2–5.

Montmerle, T., Michel, Y., Arbogast, É., Ménétrier, B., and Brousseau, P. (2018).
A 3D ensemble variational data assimilation scheme for the limited-area AROME model: Formulation and preliminary results.
Quarterly Journal of the Royal Meteorological Society, 144(716):2196–2215.

Parrish, D. and Derber, J. (1992).

The National Meteorological Center's Spectral Statistical-Interpolation Analysis System.pdf.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2017).
Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization.
Proceedings of the IEEE International Conference on Computer Vision, 2017-Octob:618–626.

Shi, W., Caballero, J., Huszar, F., Totz, J., Aitken, A. P., Bishop, R., Rueckert, D., and Wang, Z. (2016).
Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network.
Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016-Decem:1874–1883.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015).
Going deeper with convolutions.
Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 07-12-June:1–9.

Theis, L., Shi, W., Cunningham, A., and Huszár, F. (2017).
Lossy Image Compression with Compressive Autoencoders.
pages 1–19.

Wang, M., Liu, B., and Foroosh, H. (2018).
Factorized Convolutional Neural Networks.
Proceedings - 2017 IEEE International Conference on Computer Vision Workshops, ICCVW 2017, 2018-Janua:545–553.

Woo, S., Park, J., Lee, J.-y., and Kweon, I. S.
CBAM: Convolutional Block Attention Module.

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. (2017).
Aggregated residual transformations for deep neural networks.
Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 2017-Janua:5987–5995.

Zhang, Y., Li, K., Li, K., Zhong, B., and Fu, Y. (2019).
Residual Non-local Attention Networks for Image Restoration.
pages 1–18.

Zhou, L., Sun, Z., Wu, X., and Wu, J. (2019).
End-to-end Optimized Image Compression with Attention Mechanism.