Imperial College
London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Attention-based Convolutional Autoencoders for 3D-Variational Data Assimilation

Author:
Julian Mack

Supervisor:
Dr. Rossella Arcucci

Submitted in partial fulfillment of the requirements for the MSc degree in Computing (Machine Learning) of Imperial College London

September 2019

Abstract

We propose a 'Bi-Reduced Space' approach to solving 3D Variational Data Assimilation (3D-VarDA) using Convolutional Autoencoders (CAEs). We extend the approach of Parrish et al., 1992 in which a Control Variable Transform (CVT) is used to reduce the space of the background covariance matrix. We use a CAE to introduce a second reduced space: a latent representation of this CVT. We find that we can decrease the size of our representation by $\mathcal{O}(10^3)$ compared with the Parrish et al., method and, at the same time, **increase** our data assimilation performance. This method has considerably lower online computational cost than the traditional 3D-VarDA CVT approach and we demonstrate this experimentally. In this work, we use data from the MAGIC project on pollution modelling and the open-source fluid dynamic software Fluidity.

Contents

Chapter 1

Introduction

Data Assimilation (DA) is an uncertainty quantification technique that is used to insert observation data into a forecasting model. Typically this produces predictions that are more accurate than those that would be attainable if the model or the observations were used independently. For most real-world applications, the size of the systems make DA very computationally costly. For example, the Met Office's DA procedure runs online in one of the world's largest supercomputers (1) and is still only able to assimilate observations once every six hours (2). This results in a steady increase in prediction error on a six hour time-frame despite the fact that the Met Office has a network of sensors providing new observations on sub-second intervals.

Depending on the system state size and the compute available, it is often necessary to solve data assimilation in a reduced space in order to achieve real-time DA. Convolutional Autoencoders (CAEs) have had huge successes in computer vision (3) (4) including in image compression (5) (6). In this work, we use CAEs to compress a physical domain to a reduced space in which DA can be performed efficiently.
This work utilises the Variational approach to data assimilation – specifically the incremental formulation (7) with a Control Variable Transform (CVT) to factorise the background error covariance $\boldsymbol{B}$ as $\boldsymbol{B} = \boldsymbol{V}\boldsymbol{V}^T$ (8). Many authors use eigenanalysis techniques such as PCA or TSVD to precondition $\boldsymbol{V}$ by reducing its rank (9). We propose swapping these eigenanalysis approaches for convolutional autoencoders.

This research is part of the 'Managing Air for Green Inner Cities', or MAGIC project (10), (11) (9) (12): an international collaboration which aims to produce models to monitor and control pollution in urban areas. One of the project's stated aims is to produce: "*reduced order models [of inner city fluid flow] that allow rapid calculations for real time analysis and emergency response*". At the beginning of 2019, MAGIC authors Arcucci et al. used TSVD in 3D Variational data assimilation (3D-VarDA) as a step towards realising this aim (9)[1]. This project builds on the previous work by replacing TSVD with a CAE. The original authors used a MAGIC test-site location in South London and synthetic data generated by the open-source finite-element fluid dynamic software Fluidity (`http://fluidityproject.github.io/`). We test

[1]The supervisor of this project, Dr Arcucci, was the lead author in this work.

the proposed approach on the same domain and data as used in the original study to enable a clear comparison between the approaches. We find that our method gives considerably more accurate predictions and, in most cases, provides them sooner that the previous approach. In fact, our method is also more accurate (and much faster) than the Parrish et al., CVT formulation (8) in which the rank of $\boldsymbol{V}$ is not reduced.

Although our work is specific to the MAGIC project, the proposed approach is non-intrusive and is therefore applicable to any data assimilation problem in which a reduced order system is used.

In this paper we make the following contributions:

1. We propose a new 'Bi-reduced space' 3D-VarDA formulation that has an online complexity that is independent of the number of assimilated observations meaning that it can be used with arbitrarily dense sensor networks. We show that our approach has lower online complexity than that (9) while also proving that it gives approximately equivalent forecasts.

2. We create and evaluate 3D extensions of a range of state-of-the-art 2D Convolutional Autoencoders for image compression. To our knowledge, we are the first to extend the image compression network of (12) and image restoration GRDN of (13) to three-dimensions. We find that Zhou et al.'s attention-based model (12) performs best, and make some small changes to this system including the replacement of vanilla residual blocks (14) with 'NeXt' residual blocks (15) to reduce decoder inference time.

3. This adapted Zhou et al. CAE, in combination with our proposed DA formulation, achieves a substantial relative reduction in DA error of 37% compared with the Arcucci et al. TSVD approach (9). Depending on the number of assimilated observations, the proposed method is up x30 faster. We discuss the speed-accuracy tradeoff at length in section 5.2.

4. We release a well tested open-source Python module `VarDACAE` that enables users to easily replicate our experiments, use our model implementations, and train CAEs for any Variational data assimilation problem. The repository can be found at `https://github.com/julianmack/Data_Assimilation`.

The structure of this paper is as follows: in the following chapters 2 and 3 we will cover the necessary background and related work in the data assimilation and CAE literature. In the latter chapter we focus on the image compression application as we believe this is a similar use-case to ours. In chapter 4 we will describe the contributions of this work which we evaluate in chapter 5 and discuss in chapter 6 before concluding with chapter 7.

Chapter 2

Background: Data Assimilation

The canonical application of Data Assimilation (DA) (16) (17) is Numerical Weather Prediction (NWP) (7), (18), (19) but the technique is applicable in any scenario where both a forecasting model and observational data is available. As such, it has now been utilised in contexts as diverse as oceanic modelling (20), (21), solar wind prediction (22) and inner city pollution modelling (9), the latter of which is the test-case for this work.

Forecasting models introduce uncertainty from numerous sources. These include, but are not limited to, uncertainty in initial conditions, imperfect representations of the underlying physical processes and numerical errors. As a result, a model without access to real-time data will accumulate errors until its predictions no longer correspond to reality (23). Similarly, all observations will have an irreducible uncertainty as a result of imperfect measuring devices. The key idea in DA is that the overall uncertainty in a forecast can be reduced by producing a weighed average of model forecasts and observations. In DA, quantities with lower uncertainty are given a larger importance and spatial and temporal correlations between data is taken into account.

Since we approximately know how the system develops (as we have a forecasting model), DA is an 'inverse' problem which can be summarised as:

> *"what set of initial conditions will seed the models to best predict the known observations?"*(24)

The problem ill-posed so we must introduce *a priori* information in the form of historical data and knowledge of the underlying physics. Broadly speaking, there are three methods to approximately solve the data assimilation problem (25):

i Ensemble Kalman Filters (EnKFs). KFs (26) are a Bayesian method of incorporating multiple sources of Gaussian uncertainty. In DA, they are typically used in ensembles (EnKF) (20) to generate error statistics.

ii Variational Data Assimilation (VarDA) methods (18), (19). VarDA approaches involve minimising a cost function to obtain a single maximum likelihood es-

timate. In VarDA, errors are also assumed to be Gaussian, but unlike EnKF, VarDA does not produce uncertainty estimates.

iii Monte-Carlo methods allow assimilation of data from sources with non-Gaussian uncertainties. The most common type of method is the Particle Filter (PF) (27).

PFs are a generalisation of KFs in which non-Gaussian distributions are represented with an ensemble of models. While theoretically appealing, PFs suffer from 'collapse' in which a single model trajectory is assigned all the weight in the sample (28). As a result, they require infeasibly large ensembles to give useful predictions although recent work by Graham et al. may have found a way to scale them by interpolating between a patchwork of small domains (29). As PFs are not currently in operational use (25) (30) we will not cover them here but the interested reader should see the the review of these and other Monte-Carlo methods in (27).

In this work, we use the VarDA approach in order to test our hypothesis rather than KFs. However, as we are making Gaussian assumptions, we believe the proposed method is extendable to reduced space KFs.
In the following sections we will briefly summarise the theoretical work relevant to this project starting with data assimilation definitions in section (2.1). In section (2.2) we will detail the VarDA variant that we evaluate and extend in this work while in 2.3 we briefly describe TSVD to enable a comparison between the Arcucci et al. approach and the proposed method.

2.1 Definitions

We will mostly follow the data assimilation notation given in the review paper (25).

- Let $\boldsymbol{x}_t$ represent the state of the model at time t such that:

$$\boldsymbol{x}_t \in \mathbb{R}^n \tag{2.1}$$

where n is the number of elements in the state vector. The state for T total time-steps can be given in a single matrix:

$$\boldsymbol{X} = [\boldsymbol{x}_0, \boldsymbol{x}_1, ..., \boldsymbol{x}_T] \quad \in \quad \mathbb{R}^{n \times T} \tag{2.2}$$

In most practical problems, n is large and of and order $\geq \mathcal{O}(10^6)$.

- Let $\boldsymbol{y}_t$ represent the observation space of the system where:

$$\boldsymbol{y}_t \in \mathbb{R}^M \tag{2.3}$$

where typically $M << n$. The Met Office uses $M = 0.01n$ (2).

- Let $\boldsymbol{\mathcal{H}}_t$ be an observation operator such that:

$$\boldsymbol{\mathcal{H}}_t[\boldsymbol{x}_t] = \boldsymbol{y}_t + \boldsymbol{\epsilon}_t \tag{2.4}$$

where $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{R}_t)$ is the observation error. Often, observations are assumed to be uncorrelated meaning that $\boldsymbol{R}_t$ is diagonal. When all observations are of the same type and made with the same device we have:

$$\boldsymbol{R}_t = \sigma_0^2 \boldsymbol{I} \tag{2.5}$$

- Let $\boldsymbol{\mathcal{M}}_{t-1,t}$ be the forecast model that propagates the system forward from time-step $t-1$ to t such that:

$$\boldsymbol{x}_t = \boldsymbol{\mathcal{M}}_{t-1,t}[\boldsymbol{x}_{t-1}] + \boldsymbol{\eta}_t \tag{2.6}$$

where $\boldsymbol{\eta}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q}_t)$ is the model error introduced over this interval.

- Let $\boldsymbol{x}_t^b \in \mathbb{R}^n$ be the background state at time-step t. All *a priori* information about the system is introduced in the first background state $\boldsymbol{x}_0^b$ and the model $\boldsymbol{\mathcal{M}}_{t-1,t}$. Future background estimates are then defined according to the free-running model in which $\boldsymbol{\eta}_t$ is assumed to be zero and therefore:

$$\boldsymbol{x}_t^b = \boldsymbol{\mathcal{M}}_{t-1,t}[\boldsymbol{x}_{t-1}^b] \tag{2.7}$$

- Let $\boldsymbol{B}_t$ represent the background state $\boldsymbol{x}_t^b$ covariance matrix. In theory, it is found by evaluating:

$$\boldsymbol{B}_t = (\boldsymbol{x}_t^b - \boldsymbol{x}_t^*)(\boldsymbol{x}_t^b - \boldsymbol{x}_t^*)^T \tag{2.8}$$

where $\boldsymbol{x}_t^*$ is the unknown true state of the atmosphere.

In practice, even if the true state were known, this matrix is too large to fit in memory as it is in n^2 which is $\geq \mathcal{O}(10^{12})$ for most practical problems. There are many ways of representing $\boldsymbol{B}_t$ with fewer than n^2 pieces of information, including localisation (31), Control Variable Transforms (CVTs) (8) (32) and CVTs with TSVD (9). In this work we present a novel method of implicitly representing this matrix using CVTs with Autoencoders.

2.2 Variational DA, VarDA

VarDA involves minimising a cost function in order to find the most likely state values $\boldsymbol{X}^{DA}$ given the observations $\boldsymbol{y}_t$, model $\boldsymbol{\mathcal{M}}_{t-1,t}$ predictions and their uncertainties. The problem is to find the initial state $\boldsymbol{x}_0^{DA}$ that satisfies:

$$\boldsymbol{x}_0^{DA} = \arg\min_{\boldsymbol{x}_0} J(\boldsymbol{x}_0) \tag{2.9}$$

where the cost function $J(\boldsymbol{x}_0)$ is:

$$\begin{aligned} J(\boldsymbol{x}_0) = \frac{1}{2}\left\|\boldsymbol{x}_0 - \boldsymbol{x}_0^b\right\|^2_{\boldsymbol{B}_0^{-1}} + \frac{1}{2}\sum_{t=0}^{T}\left\|\boldsymbol{y}_t - \boldsymbol{\mathcal{H}}_t[\boldsymbol{x}_t]\right\|^2_{\boldsymbol{R}_t^{-1}} \\ + \frac{1}{2}\sum_{t=0}^{T}\left\|\boldsymbol{x}_t - \boldsymbol{\mathcal{M}}_{t-1,t}[\boldsymbol{x}_{t-1}]\right\|^2_{\boldsymbol{Q}_t^{-1}} \end{aligned} \tag{2.10}$$

In VarDA, this cost-function is explicitly differentiated and then approximately solved by first-order optimisation routines (see section 2.2.3). This formulation has an equivalent solution to the KF approach. For a proof of this, see Appendix A.

The terms in (2.10) can be interpreted as follows:

- The first term is the background term J^b that measures the difference between our *a priori* expectation and the initial model state after DA has been performed.
- The second term is the observation term J^o which measures the deviation between the observations and the model predictions.
- The third term J^Q measures the size of the model error.

Note that we are assuming all errors are Gaussian by using this least-squares formulation. This problem specification is known as the 'weak constraint 4D-Var' (33) where:

- '4D' refers to the fact that we are considering three spatial dimensions as well as one temporal dimension. It is contrasted with 3D-Var in which a single time-step is assimilated.
- 'Weak' refers to the fact that model errors are non-zero and, therefore, we are not requiring the system to follow the model trajectory exactly. This is contrasted with *strong constraint* VarDA in which $J^Q = 0$.

In the current work we are considering the 3D case. As we are only only assimilating a single time-step, we can drop the t subscripts and the cost function is given as:

$$J(\boldsymbol{x}) = \frac{1}{2}\left\|\boldsymbol{x} - \boldsymbol{x}^b\right\|^2_{\boldsymbol{B}^{-1}} + \frac{1}{2}\left\|\boldsymbol{y} - \boldsymbol{\mathcal{H}}[\boldsymbol{x}]\right\|^2_{\boldsymbol{R}^{-1}} \tag{2.11}$$

In the 3D case, the distinction between strong and weak VarDA disappears as we do not use the model to project forward to future time-steps.

2.2.1 Incremental VarDA

The cost functions 2.11 (and 2.10) will be convex if $\boldsymbol{\mathcal{H}}_t$ (and $\boldsymbol{\mathcal{M}}_{t-1,t}$) are linear which they are not in general. We can approximately linearize these operators about the background state $\boldsymbol{x}^b$ by formulating the problem in terms of perturbations to this state in a method known as the incremental formulation (7)[1]:

$$\delta\boldsymbol{x} \coloneqq \boldsymbol{x} - \boldsymbol{x}^b \tag{2.12}$$

[1]Note: in theory, the perturbations can be made with respect to a general state $\boldsymbol{x}^g$ instead of $\boldsymbol{x}^b$ but this is rarely useful in practice.

The problem statement then becomes:

$$\begin{aligned} \delta \boldsymbol{x}^{DA} &= \arg\min_{\delta \boldsymbol{x}} J(\delta \boldsymbol{x}) \\ J(\delta \boldsymbol{x}) &= \frac{1}{2}\delta \boldsymbol{x}^T \boldsymbol{B}^{-1} \delta \boldsymbol{x} + \frac{1}{2} \left\| \boldsymbol{d} - \boldsymbol{H}\delta \boldsymbol{x} \right\|^2_{\boldsymbol{R}^{-1}} \end{aligned} \tag{2.13}$$

where $\boldsymbol{H}$ is the observation operator linearized about the background state and the 'misfit' between observation and expected observation is:

$$\boldsymbol{d} = \boldsymbol{y} - \boldsymbol{H}\boldsymbol{x}^b \tag{2.14}$$

The more general 4D-Var weak constraint incremental formulation is given in (7) and reviewed in a modern context in (25).

2.2.2 Control Variable Transform

As discussed, to deal with the fact that $\boldsymbol{B}$ is in $\mathcal{O}(n^2)$ we must represent it implicitly. A common way of doing this is by using the formulation proposed in (8) and reviewed in (32) which states that:

$$\delta \boldsymbol{x} = \boldsymbol{V}\mathbf{w} \tag{2.15}$$

$$\boldsymbol{B} = \boldsymbol{V}\boldsymbol{V}^T \tag{2.16}$$

In this case the problem can be written as:

$$\begin{aligned} \mathbf{w}^{DA} &= \arg\min_{\mathbf{w}} J(\mathbf{w}) \\ J(\mathbf{w}) &= \frac{1}{2}\mathbf{w}^T \mathbf{w} + \frac{1}{2} \left\| \boldsymbol{d} - \boldsymbol{H}\boldsymbol{V}\mathbf{w} \right\|^2_{\boldsymbol{R}^{-1}} \end{aligned} \tag{2.17}$$

Following equation 2.8, we can see that $\boldsymbol{V}$ is theoretically found by stacking a series of background states $\boldsymbol{X}^b$ and subtracting the true state, $\boldsymbol{x}^*$:

$$\boldsymbol{V}_t = (\boldsymbol{X}^b_t - \boldsymbol{x}^*_t) \tag{2.18}$$

In reality, we do not know $\boldsymbol{x}^*$ but in the 3D case we can estimate $\boldsymbol{V}$ with a sample of S model state forecasts $\boldsymbol{X}^b$ that we set aside as 'background' such that:

$$\boldsymbol{X}^b = [\boldsymbol{x}^b_0, \boldsymbol{x}^b_1, ..., \boldsymbol{x}^b_S] \in \mathbb{R}^{n \times S}$$

$$\boldsymbol{V} = (\boldsymbol{X}^b - \boldsymbol{x}^b) \ \in \mathbb{R}^{n \times S} \tag{2.19}$$

where $\boldsymbol{x}^b$ is the mean of the sample of background states. In the incremental VarDA formulation this gives:

$$\boldsymbol{V} = [\delta \boldsymbol{x}^b_0, \ \delta \boldsymbol{x}^b_1, \ ..., \ \delta \boldsymbol{x}^b_S] \in \mathbb{R}^{n \times S} \tag{2.20}$$

In 4D-Var, if we require flow-dependency in our representation of $\boldsymbol{B}$, we could produce $\boldsymbol{V}_t$ from an ensemble of models in the Ensemble Kalman Filter approach. From equations 2.15 and 2.20 it follows that:

$$\mathbf{w} \in \mathbb{R}^{\mathbf{S}} \tag{2.21}$$

and since $S << n$ in all practical cases, $\mathbf{w}$ is referred to as the 'reduced space'. $\boldsymbol{V}$ is therefore an affine transform from the reduced space of size S to the full space of size n. Hence, the generalised inverse of $\boldsymbol{V}$ (denoted $\boldsymbol{V}^{+}$) gives the reverse transformation from the full to the reduced space:

$$\boldsymbol{V}^{+}\delta\boldsymbol{x} = \mathbf{w} \tag{2.22}$$

Note that as $S < n$, the implicit matrix $\boldsymbol{B}$ is not full rank and so, as a result of sampling errors, this method will overestimate the covariance between states that are spatially distant. This problem is reduced by using localisation methods (31) in which covariances are attenuated by the distance between states. Localisation approaches are not used in this work but would be expected to improve the results found herein.

2.2.3 CVT VarDA Optimisation

There is a large amount of research on VarDA optimisation strategies which we do not consider deeply here. However, it is necessary to highlight a few findings which inform our approach:

- The problem in (2.17) is ill-conditioned because, in most practical contexts, the matrix $\boldsymbol{V}$ has a large condition number[2]. For an example see (9).
- Conjugate Gradient (CG) methods are successful in minimising poorly conditioned functions (34) and a CG methods 'L-BFGS' has been show by a number of authors (35), (36), (9) to give faster convergence than its counterparts for problems of the size encountered in DA. However, the rate of convergence is still dependent on the condition number of the Hessian (36).
- The CVT approach (see equation 2.17), while poorly conditioned, still confers an optimisation advantage over the standard incremental formulation (see equation 2.13) as the former has the Hessian:

  $$(\boldsymbol{I} + \boldsymbol{V}^{T}\boldsymbol{H}^{T}\boldsymbol{R}^{-1}\boldsymbol{H}\boldsymbol{V}) \tag{2.23}$$

  which has a minimum eigenvalue of 1 and is dominated by the condition number of $\boldsymbol{V}$ while the latter has the Hessian:

  $$(\boldsymbol{B}^{-1} + \boldsymbol{H}^{T}\boldsymbol{R}^{-1}\boldsymbol{H}) \tag{2.24}$$

  which has a minimum eigenvalue of 0 and is dominated by the condition number of $\boldsymbol{B}$. This difference in condition-number and subsequent convergence rate has been extensively observed in the literature including in (37) and (9).

[2]The condition number is the ratio of the largest to the smallest eigenvalues.

We follow Arcucci et al. in using L-BFGS in the implementation of our approach but, as our memory requirements are considerably less, there is a sound argument for investigating other minimisation routines in future work.

2.3 Truncated SVD

One way to precondition $\boldsymbol{V}$ and thereby increase the speed of convergence is to use an eigenanalysis technique such as SVD (38) (39) or PCA to generate Empirical Orthogonal Functions (EOFs) (40)[3] and remove low-variance modes from $\boldsymbol{V}$. We have implemented 3D-VarDA with TSVD as described in (9) and we evaluate the success of our proposed approach against this scheme in section 5.2. In order to draw out the theoretical differences between the methods we briefly summarise TSVD here.

The singular value decomposition of matrix $\boldsymbol{V}$ is as follows:

$$\boldsymbol{V} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{W}^T \tag{2.25}$$

where $\boldsymbol{U} \in \mathbb{R}^{n\times S}$, $\boldsymbol{\Sigma} \in \mathbb{R}^{S\times S}$ are both orthogonal and $\boldsymbol{W} \in \mathbb{R}^{n\times S}$ is diagonal and contains $\boldsymbol{V}$'s eigenvalues σ_i such that:

$$\Sigma = \text{diag}[\sigma_1, \sigma_2, ..., \sigma_S] \tag{2.26}$$

where the eigenvalues appear in decreasing order:

$$\sigma_1 > \sigma_2 > ... > \sigma_S > 0 \tag{2.27}$$

To perform TSVD, τ of the modes are retained where $0 < \tau < S$ and the matrix $\boldsymbol{V}_\tau$ is reconstructed such that:

$$\Sigma_\tau = \text{diag}[\sigma_1, ..., \sigma_\tau, 0, ..., 0] \tag{2.28}$$

$$\boldsymbol{V}_\tau = \boldsymbol{U}\boldsymbol{\Sigma}_\tau\boldsymbol{W}^T \tag{2.29}$$

This has generalised inverse:

$$\boldsymbol{V}_\tau^+ = \boldsymbol{W}\boldsymbol{\Sigma}_\tau^+\boldsymbol{U}^T \tag{2.30}$$

where $\boldsymbol{\Sigma}_\tau^+$ is the generalised inverse of $\boldsymbol{\Sigma}_\tau$ such that:

$$\boldsymbol{\Sigma}_\tau = \text{diag}\left[\frac{1}{\sigma_1}, ..., \frac{1}{\sigma_\tau}, 0, ..., 0\right] \tag{2.31}$$

The authors of (9) provide a method for choosing τ which states that the optimal truncation occurs where $\sigma_\tau = \sqrt{\sigma_1}$. The authors made an algebraic mistake that undermines this conclusion[4] but the heuristic appears to work well in practice so we have used it as one of a number of options when evaluating TSVD in section 5.2.

[3]The term EOF is used widely in the DA literature but is absent from the machine learning literature.

[4]Specifically, in equation (43), the paper states $f(\sigma_\tau) = \frac{\sigma_{\tau-1}}{\sigma_1} + \frac{\sigma_1}{\sigma_\tau} = \frac{\sigma_\tau\sigma_{\tau-1}+\sigma_1}{\sigma_1\sigma_\tau}$. This should read $f(\sigma_\tau) = \frac{\sigma_{\tau-1}}{\sigma_1} + \frac{\sigma_1}{\sigma_\tau} = \frac{\sigma_\tau\sigma_{\tau-1}+\sigma_1^2}{\sigma_1\sigma_\tau}$. This has downstream consequences on their theoretical argument for the optimal truncation parameter.

Chapter 3

Background: Autoencoders

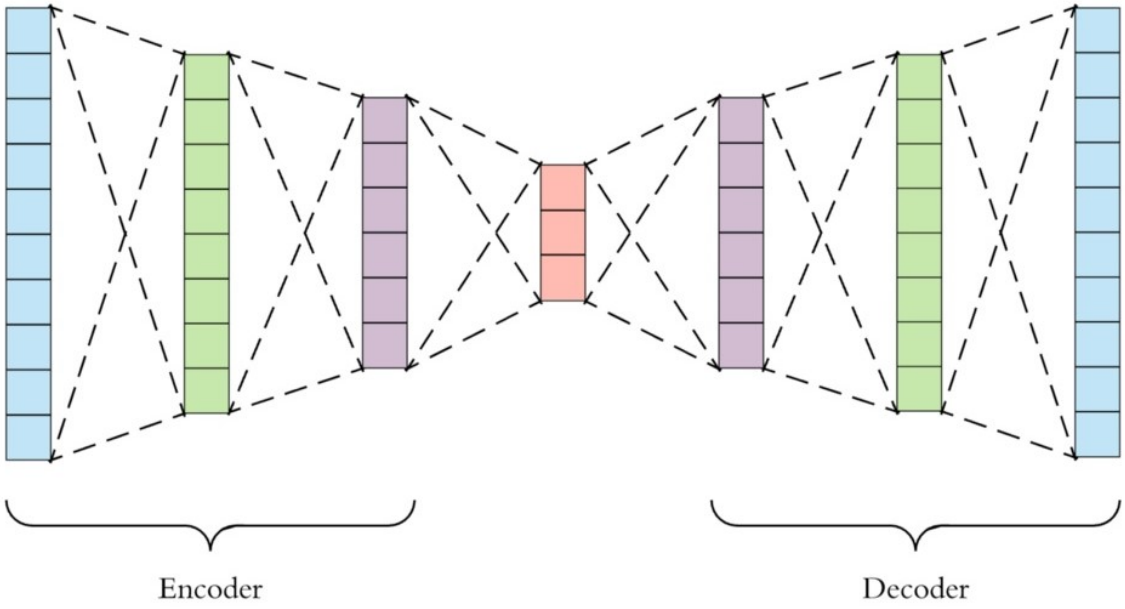

Figure 3.1: An encoder compresses the input (blue) to a smaller latent representation (red). The decoder reconstructs the input in a lossy process. Image adapted from (41).

Autoencoders are an unsupervised machine learning method first proposed in (42). AEs are composed of two elements: an encoder which compresses the input to a small latent representation, and a decoder which reconstructs the input (see Figure 3.1). This is achieved by introducing an 'information bottleneck' that forces the AE to find and exploit redundancies in the training data in order to *implicitly model the data distribution*. Although they are rarely thought of in this way, AEs are theoretically equivalent to a clustering algorithm (43) in the sense that the encoder learns to map commonly co-occurring inputs to a single internal representation.

As an aside, **Variational AutoEncoders, VAEs** are a tool for *explicitly* modelling the underlying data distribution. They can be used to generate new samples or to simply impose priors on the data distribution (44). VAEs have had successes in a range of generative modelling tasks, particularly NLP (45) (46) where this imposed structure

is helpful. However, as a result of the distribution constraints, VAEs tend to produce poorer reconstructions than standard AEs as discussed in chapter 20 of (47). As such, we do not use them in this work nor cover them in detail in this background chapter but they are used by many authors referenced herein and are a possible avenue for extensions of this work. The interested reader should refer to the VAE tutorial in (48) for a comprehensive overview.

Vanilla AEs are useful in any problem in which a latent data representation is needed for a downstream task but they have also had success as a standalone solution in two specific applications:

Anomaly detection (49) in which an AE is trained on a set of 'normal' examples. When this system encounters an anomaly, it will reconstruct it poorly as the compression is optimised for the training data-distribution, thereby allowing the user to identify the unusual sample. For example, as shown in Figure 3.2, an AE trained on images of healthy brains can be used to locate tumours.

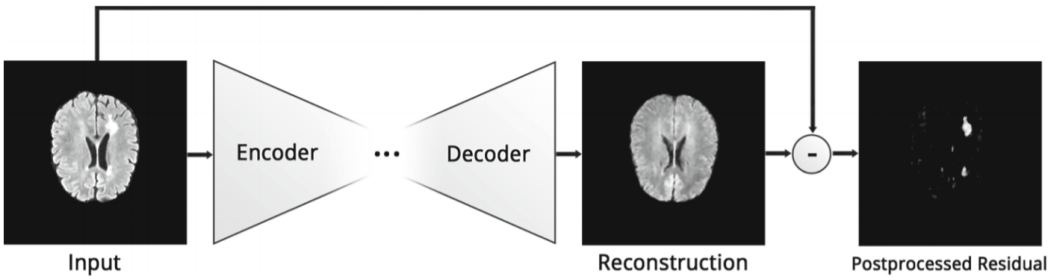

Figure 3.2: The 'anomaly detection concept at a glance. A simple subtraction of the reconstructed image from the input reveals lesions in the brain.' Image and caption from (50).

Denoising in which the AE is trained on 'clean' data and is then able to remove noise from data at inference (3) (4). The majority of this work has been in imaging, but Hayou et al. (51) used this approach to remove noise from covariance matrices in work that is closely related to ours.

When designing our AE architecture, we, like others, are faced with a challenge: with infinite design permutations available, which regions of the architecture space should be explored? With limited compute, the only rational answer is to be guided by similar work. However, while there are many examples of AE uses in DA, we did not find any in which the AE a) utilises state-of-the-art developments in machine learning and b) is similar enough to our use-case to be informative. As such, we have chosen to consider architectures in the image compression literature as, for reasons discussed in 3.3.2, we felt that this was the most appropriate parallel application.

To explain why we came to this conclusion, it is necessary to provide both a formal definition of AEs and a design specification which we give in sections 3.1 and 3.2

respectively. In section 3.3.1 we will briefly discuss AEs used in the data-assimilation literature, and why they are not applicable to our problem before moving onto to the reasons why image compression AEs *are* applicable in section 3.3.2. Having motivated our approach, we will cover the building blocks of AEs for image compression in section 3.4. Finally, in section 3.5 we will end the chapter with a discussion of AE architectures that directly influence those proposed and evaluated in later chapters.

3.1 Definitions

AEs consist of an encoder $f(\boldsymbol{x})$ and decoder $g(\mathbf{z})$ such that:

$$\begin{aligned} f(\boldsymbol{x}) &= \mathbf{z} \\ g(\mathbf{z}) &= \hat{\boldsymbol{x}} \end{aligned} \tag{3.1}$$

where $f(\cdot)$ and $g(\cdot)$ are non-linear neural networks and:

$$\boldsymbol{x}, \hat{\boldsymbol{x}} \in \mathbb{R}^n \tag{3.2}$$
$$\mathbf{z} \in \mathbb{R}^m \tag{3.3}$$
$$m < n \tag{3.4}$$

In other words, AEs compress the input data $\boldsymbol{x}$ to find a smaller latent representation $\mathbf{z}$. Baldi refers to the AE training process as an attempt to find 'a low rank approximation to the identity function' (43). Typically the AEs are trained with the L1 or L2 reconstruction error which are respectively:

$$J_1(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \sum_{i=1}^{n} |x_i - \hat{x}_i| \tag{3.5}$$
$$J_2(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \sum_{i=1}^{n} (x_i - \hat{x}_i)^2 \tag{3.6}$$

In this work, we used the L2 error $J_2(\boldsymbol{x}, \hat{\boldsymbol{x}})$ to train our AEs as we found that the networks produced in this manner performed consistently better for data assimilation than those trained with $J_1(\boldsymbol{x}, \hat{\boldsymbol{x}})$ (see Section 4.2). We experimented with fine-tuning our models with the $J_1(\boldsymbol{x}, \hat{\boldsymbol{x}})$ loss as suggested in (52) but, as described in section 5.1.3, this did not provide us with a consistent improvement.

3.2 Specification

Our AE has a number of requirements. It must:

i Compress a large state of size n to a latent representation with m components where $m << n$, with as small a loss of information loss as possible.

ii Only work well for inputs of a fixed size and scale since in data assimilation, the state space size is fixed.

iii Utilise the 3D spatial location of points in the input state.

Specification point ii) means that, all other quantities being equal, an AE that is successful for DA will need to be considerably less powerful than its equivalent for image compression[1]. Image compression networks will see a selection of, say cats, at many different distances from the camera and must be adept at compressing all of them while in our data assimilation formulation, the AE will always see the same domain at the same scale. In addition, if the image compression network is trained on the ImageNet (53) data-set it will also have to contend with images of varying sizes which our system will not.

3.3 Comparable Applications

3.3.1 AEs for DA

There are many uses of AEs in DA to reduce the space of the system but the majority of this work is for the task of Reduced Order Modelling (ROM). ROM is a method of embedding the knowledge of the expensive forecast model $\boldsymbol{\mathcal{M}}_{t-1,t}, \forall t$ in a lower dimensional latent space. The first generation of ROMs used PCA, or in the DA literature 'POD': Proper Orthogonal Decomposition to derive the latent space modes. There are many examples of POD-based ROMs but Cordier et al.'s work from 2013 is typical (54). The next generation of systems are hybrid models in which a network is trained to find approximate versions of these PODs (11) (55). This can reduce inference latency by many orders of magnitude. However, as these models are still POD-based, they dispose of a large amount of the forecast model's knowledge since POD can only creates modes that are linear combinations of the inputs[2].

In comparison, state-of-the-art ROMs obtain their latent representation with an AE that is trained 'end-to-end' to reconstruct the forecast model's state outputs. This should achieve a performance boost as well as an operational speed-up. There are many examples, but the work by Wang et al. and Loh et al. are representative (56)

[1]Note that 'all other things' are not equal here as the input data in our system is three-dimensional whereas most images are two-dimensional.

[2]There is a clear parallel in the comparison between the use of AE and POD based ROMs here and the comparison between CAEs and TSVD to precondition $\boldsymbol{V}$ in section 4.1.2.

(57). ROMs are only applicable in 4D-VarDA when an online method of generating model forecasts is required. Their architecture reflects this: as this is a sequence-to-sequence problem (58) most modern ROMs use LSTMs or other RNN variants to make their predictions. Unfortunately, this means these architectures are not directly relevant in this case. However, if our approach is extended to 4D, the methods proposed in this paper may have non-trivial interactions with ROMs which we cover briefly in an aside below.

Separate from ROMs, there are many authors who use deep learning in the DA literature but these works typically fall into one of two categories:

i The intended use of the neural network is orthogonal to ours. For example, MAGIC authors Zhu et al. teach a fully connected network to perform DA in the full space (59). This is used as a method of removing the minimisation routine but we found that, in our proposed approach, the minimisation accounts for a small fraction of the execution time (see section 4.1.3).

ii The neural network design does not reflect the state-of-the-art in machine learning. For example, early this year, Liu et al. and Quilodran et al. used AEs to perform EnKF in a reduced space for oceanic and seismic modelling respectively (60) (61). The former used a relatively basic CAE with just three convolutional layers and no residual connections while the latter neglected the spatial location of the data entirely and opted to use a fully connected network. We give a detailed review of convolution, residual blocks and other more complex CAE architectural components in section 3.4.

ROM Aside

There are two ways in which utilising a ROM and our method could potentially interact if our proposed approach is extended to 4D. The first is that, as a result of the speed-up achieved through our method, DA can be performed on shorter intervals. This means that lower quality ROMs will become viable as they will have to accurately project into the future for a smaller number of time-steps. The second potential interaction is that the two AEs could be trained end-to-end as a single system with the same training objective. This may be a fruitful route for future work.

3.3.2 AEs for Image Compression

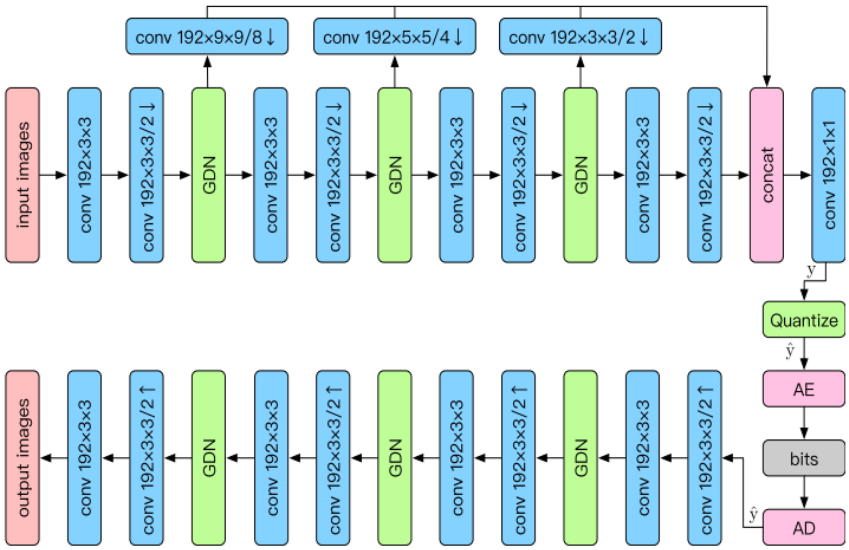

Figure 3.3: An example image compression CAE architecture (62). This model was the winner of CLIC in 2018, and was latterly extended by the authors (show in Figure 3.17) to create the winning entry in CLIC 2019 (12).

Lossy image compression is a process in which an image is contracted to a more efficient representation with a small loss of information. In the last two years convolutional autoencoders have started to outperform traditional image compression systems (5) such as JPEG and JPEG 2000[3]. There are a few key differences between the image compression problem and ours but these systems have CAE variants that hit all three of our specification criteria.

In addition, unlike DA covariance reduction, this problem has received a great deal of attention from the machine learning community. For example, there is a competition called CLIC or 'Challenges on Learned Image Compression' that runs as part of CVPR every year to find the state of the art in lossy image compression. CLIC 2019 was held in June so we have an up-to-date quantitative comparison of the available approaches. It is primarily as a result of this competition that we decided to focus on the image compression literature in order to locate a subset of AE architectures that are likely to be useful for our problem.

We used the papers of the CLIC winners and runners up from 2018-19 as a starting point and then followed their citations to give a total of approximately 40 relevant papers. This led us to other domains, particularly image segmentation[4] which

[3]In fact, a new *lossless* CAE-based image compression format L3C was proposed in May this year that also outperforms traditional lossless methods (63).

[4]In fact, we considered using segmentation instead of compression for structuring our architecture review but decided that the compression problem had more obvious parallels with our AE requirements.

heavily influenced image compression with the proposal of multi-scale learning approaches (64) as discussed in section 3.4.5. We found that there were a number of key papers that were notably absent from the CLIC citation chains so, in addition, we reviewed a further 25 machine learning papers.

The CLIC winner and runners-up this year (12), (65), (66), (67), (52) and (68) used a variety of architectural components including attention-based models (69), complex residual blocks (14) and multi-scale learning in order to improve their compression quality. In the subsequent sections, we will describe the basic elements of these systems and their precursors. We have implemented 3D extensions of the majority of these models which are detailed in section 4.2 and evaluated in chapter 5. First of all, we make two remarks about the image compression solutions.

1) Bitstream Generation

The encoder output in an image compression system is a bitstream, while in our case it is simply a vector of floats (see Figure 3.3). This adds complexity in comparison with our system in a number of ways:

i Firstly, there is a tradeoff in image compression between the compressed size and quality so most of these systems are trained with a multi-task loss of bitstream entropy and reconstruction error. In our case the latent size is specified by the user so we can simply train with the reconstruction loss.

ii Secondly, the quantization operation is non-differentiable and therefore the systems cannot be trained by backpropagation directly. Some authors deal with this by using an approximate derivative of the quantization operation (70). Others follow (71) and (6) in realising that the quantization operation is equivalent to the addition of uniform noise to the latent system (5),(62) (12) and this addition avoid the need to actually perform quantization during training. This second approach means that the AEs are in fact VAEs with a Gaussian likelihood.

iii Finally, in order to achieve an efficient compression, these systems use 'importance maps' (72) which are a form of attention that determine how many bits should be allocated to each region of the image. This is useful because smooth areas of an image require fewer encoding bits than areas with large gradients.

These considerations are largely ignored in this paper although there is definitely scope for future work that uses quantization to to create a more efficient latent representation.

2) Alternative Approaches

It is worth noting that there are whole classes of image compression networks that are *not* applicable here. LSTM compression networks iteratively reduce the image

size by successive down-sampling the input (70), (73) but these networks violate our second specification point of using just a fixed input size and are also comparatively slow. More recently, Huang et al. proposed a GAN-based AEs to generate visually plausible reconstructions (74). There is a lot of work to be done in proving the 'correctness' of GAN-generated samples and we were not confident that 'visually plausible' in the image domain would translate to scientifically correct in the 3D field domain. Nevertheless, this is certainly another route for future work.

3.4 Building Blocks

3.4.1 Convolutional Neural Networks

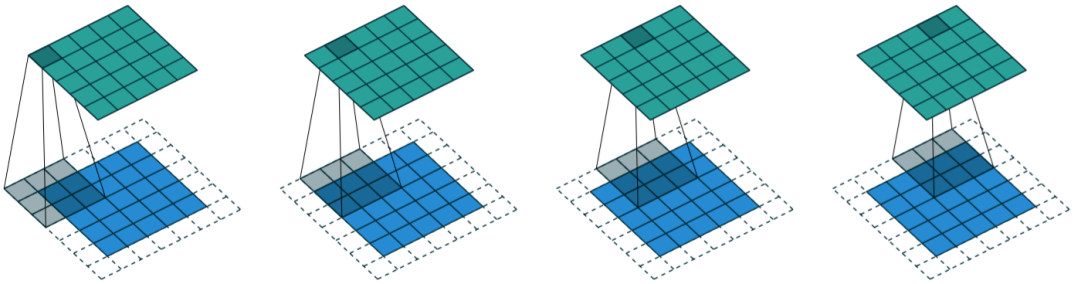

Figure 3.4: Example convolution for 2D system. 5x5 input (blue) with (white) padding=1 is convolved with a 3x3 kernel to produce the output feature map (green). Image and (adapted) caption from (75).

The simplest type of autoencoder is the linear AE in which fully connected layers are used to down-sample the input to produce a small latent representation (42). Linear AEs, unlike 'Convolutional Autoencoders', CAEs, do not account for the spatial location of the data so tend to perform poorly for image processing tasks. The key insight of neural networks that utilise convolution[5], collectively known as CNNs, is that many features are invariant to their position in the input. A cat is a cat regardless of whether is is viewed in the top right or bottom left corner of an image. This can also be thought of as a prior that input data is locally self-similar and was first encoded in network in Fukushima's neocognitron in 1980 (76). The modern CNN was introduced and used by LeCun et. al in their seminal paper on MNIST handwritten digit recognition in 1998 (77) but it was not until after the successes of Alex Krizhevsky et al. on ImageNet (53) classification in 2012 (78) that CNNs became dominant in computer vision applications.

[5]Note: As described below, modern CNNs actually utilise the cross-correlation operator rather than the convolution operator.

The primary component of a CNN is the discrete cross-correlation operator $\star$ which for filter F and input I, defined over an infinite one-dimensional domain, is given as:

$$(F \star I)(i) = \sum_{u=-\infty}^{\infty} F(u)I(i+u) \tag{3.7}$$

In CNNs, this operation is performed over *finite* input domains. A filter's 'kernel' is strided across the input and the dot-product is taken between the input patch and kernel at all stationary locations. This process produces an output feature map as shown in Figure 3.4. In modern CNNs, a bank of N filters operate in parallel on the input to each layer (see Figure 3.5). In our application, equation 3.7 can be generalised to N four-dimensional (three spatial and one channel dimension) filters operating in parallel (see Figure 3.5) over spatial input dimensions of size (H, W, D):

$$(F \star I)(i,j,k) = \sum_{n=1}^{N} \sum_{u=0}^{H} \sum_{v=0}^{W} \sum_{w=0}^{D} F_n(u,v,w)I_n(i+u, j+v, k+w) \tag{3.8}$$

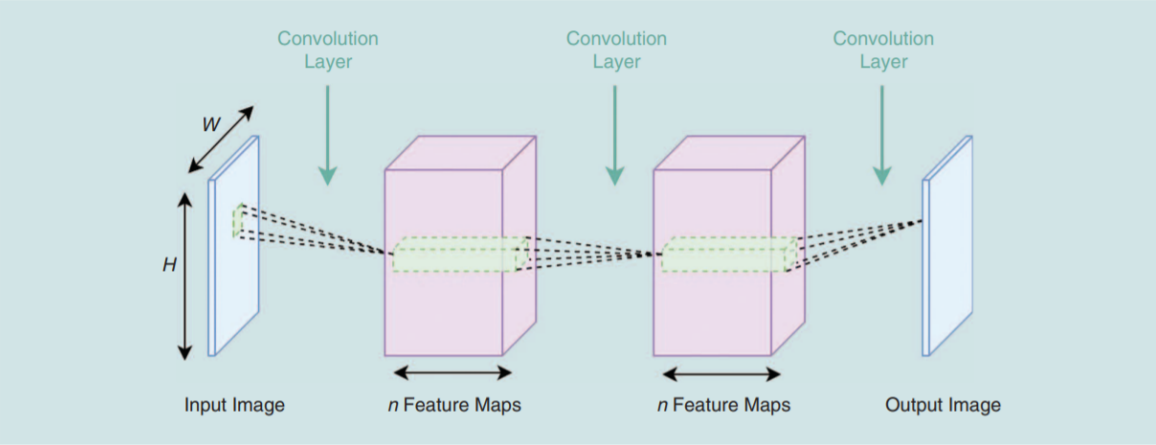

Figure 3.5: Banks of n convolutional filters operate in parallel on their inputs so that each 'channel' is able to extract a different set of features. Image from (79).

In the context of the CAE framework, a key design decision regards the method of down-sampling. Historically, down-sampling was achieved with pooling operations, such as local max pooling see (78) but current best-practice in image compression is to use strided-convolution so that the network can learn its own down-sampling routine (80). There are a number of other salient considerations in terms of how padding, stride and dilation affect output size but these will not be covered here. The interested reader should consult (75) for a comprehensive discussion.

Another important insight from image compression CAEs, is that the majority of systems use largely symmetric encoder and decoders with equal number of layers

in each. Typically, the encoder's strided convolutions are replaced by transposed convolutions for up-sampling by the decoder. This means there are similar numbers of parameters in $f(\boldsymbol{x})$ and $g(\mathbf{z})$ so that, when properly trained, neither the encoder nor decoder is worse at their respective task. The exception to this rule, noted in (5), is when just one of the encoding or decoding processes is time sensitive at inference. In this case, the constituent that has tighter latency constraints should be created with fewer layers and parameters.

3.4.2 Activation Functions

Convolutional and fully connected layers are matrix operations and therefore only produce affine transformations of their inputs. Activation functions are non-linear transforms $\sigma(\cdot)$ that are applied between layers to enable the network to produce arbitrary non-linear combinations of input features. The choice of activation function can have a large impact on training time and model performance and we have investigated a range of them in section 5.1.2. In this section we will briefly summarise the advantages and shortcomings of five activation functions that are used in image compression networks: sigmoids, ReLUs, Leaky ReLUs, their extension PReLUs and GDNs.

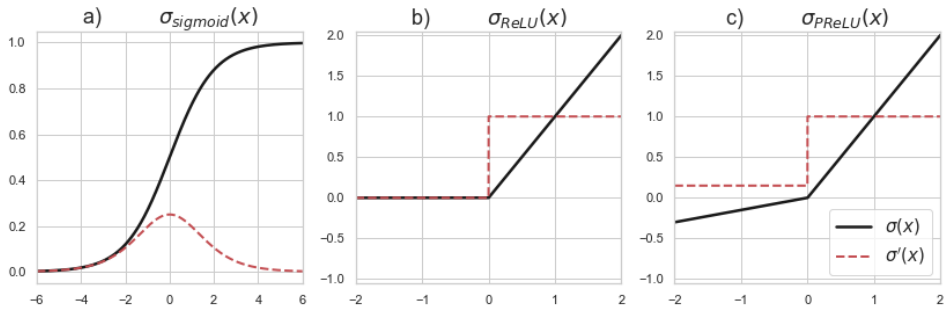

Figure 3.6: Three common activation functions (black) and their derivatives (red).
a) $\sigma_{\text{sigmoid}}(\boldsymbol{x})$, b) $\sigma_{\text{ReLU}}(\boldsymbol{x})$, c) $\sigma_{\text{PReLU/LReLU}}(\boldsymbol{x}; \alpha = 0.15)$

Sigmoid

Sigmoid activation functions $\sigma_{\text{sigmoid}}(\cdot)$ (see Figure 3.6a) were suggested as a method of imitating the response of a neuron in the brain and were in widespread use in neural networks before 2012:

$$\sigma_{\text{sigmoid}}(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{x}}} \tag{3.9}$$

A large disadvantage of $\sigma_{\text{sigmoid}}(\cdot)$ activations is that they suffer from the 'vanishing gradients problem' as the derivative of the sigmoid is very close to zero for all values where $|\boldsymbol{x}| > 5$. In fully connected layers this means that large magnitude activations through a neuron will result in zero backpropagated gradient and no weight updates. This neuron is 'dead' because it will no longer learn and hence the model capacity

has decreased. The same phenomena can be seen in CNNs, except, in this case, a whole channel may die.

ReLU

In 2010, Hinton et. al proposed Rectified Linear Units $\sigma_{\text{ReLU}}(\cdot)$ (81), or ReLUs (see Figure 3.6b) that are considerably faster to train than their sigmoid counterparts and yet they typically lead to equivalent or superior performance. The ReLU activation is:

$$\sigma_{\text{ReLU}}(\boldsymbol{x}) = \max(0, \boldsymbol{x}) \tag{3.10}$$

The use of ReLUs was a key component to Krizhevsky and Hinton's success on ImageNet in 2012 (78). Their training speed-up over sigmoid activations is a result of:

i The high computational complexity of the exponential function.

ii The low complexity of the ReLU gradient calculation since: $\sigma'_{\text{ReLU}}(\boldsymbol{x}) = \begin{Bmatrix} 1 & \boldsymbol{x} > 0 \\ 0 & \boldsymbol{x} < 0 \end{Bmatrix}$

iii The fact that a smaller proportion of the gradients vanish.

ReLU is state of the art for many computer vision tasks (82) but can also cause some neurons or channels to die as negative activations will result in zero gradient.

PReLU and LReLU

To tackle this issue, the Leaking ReLU (83), LReLU and parametized ReLU, PReLU (see Figure 3.6c) were proposed (84):

$$\sigma_{\text{PReLU}}(\boldsymbol{x}) = \begin{Bmatrix} \boldsymbol{x} & \boldsymbol{x} > 0 \\ \alpha\boldsymbol{x} & \boldsymbol{x} \leq 0 \end{Bmatrix} \tag{3.11}$$

where $\alpha \in (0, 1)$. In the Leaking ReLU variant, α is fixed at a small value whilst in the PReLU system, the parameter α is learnt through backpropagation. α can be learnt, globally, layer-wise or channel-wise. The replacement of ReLUs with PReLUs by He et al. was the first occasion on which human-level performance was surpassed on ImageNet classification (84).

GDN Transforms

Finally, Generalised Divisive Normalisation transformations or GDNs were proposed by Ballé et al. in 2015 (85) specifically in the context of natural image denoising and compression. The GDN transform normalises and Gaussianizes the data on the assumption that it is drawn from a very general, implicitly defined probability distribution. The authors used GDNs as activation function for the first time in 2018 in an image compression CAE (6). This design was used as back the backbone architecture

of Zhou et al.'s system which won CLIC 2018 (62) and 2019 (12).

They are very versatile functions defined as:

$$\sigma_{\text{GDN}}(x) = g(\boldsymbol{x};\boldsymbol{\theta}) \qquad \text{s.t.} \qquad g(z_i) = \frac{z_i}{\left(\beta_i + \sum_j \gamma_{ij}|z_j|^{\alpha_{ij}}\right)^{\varepsilon_i}}$$
$$\text{and} \qquad \boldsymbol{z} = \boldsymbol{H}\boldsymbol{x}. \tag{3.12}$$

The overall parameter vector is $\boldsymbol{\theta} = \{\boldsymbol{\beta}, \boldsymbol{\varepsilon}, \boldsymbol{H}, \boldsymbol{\alpha}, \boldsymbol{\gamma}\}$ which, for C input channels, has a total of $2C + 3C^2$ parameters. GDNs are a multivariate generalisation of the sigmoid function (see the similarity between 3.6a and 3.7b) and have an inverse 'IGDN' which must be used in the decoder.

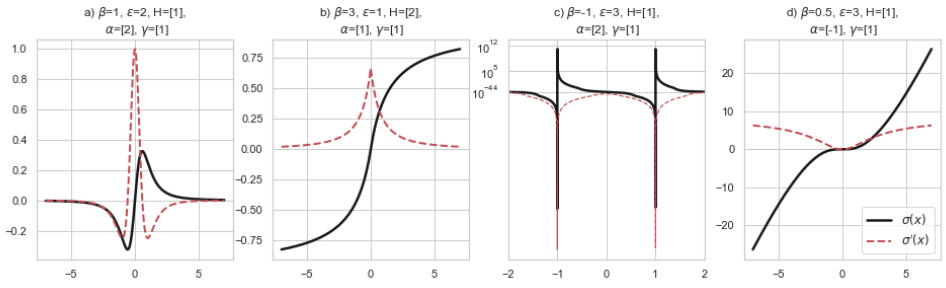

Figure 3.7: GDN transforms for a range of hyperparameters $\boldsymbol{\theta}$ in one dimension. As $\boldsymbol{H}, \boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ are all matrices, a higher dimensional GDN will exhibit even more variation than the above. Figure c) is displayed with a symmetric log scale for illustrative purposes.

Discussion

As of 2019, sigmoid functions are almost never used between layers but find uses within more complex blocks including LSTMs and CBAMs (see section 3.4.7). ReLUs are used in the majority of computer vision applications. In fact, there is a curious bias of modern machine learning practitioners, noted by many including (86), to use ReLUs without experimenting with PReLUs (or GDNs) despite the fact that the latter activations are frequently more successful (87) (88). This trend continues: state of the art on ImageNet classification as of September 2019 (Facebook's FixResNeXt-101) uses ReLU activations (82) and yet their paper makes no mention of activation function tuning[6].

In theory, the activations that perform well in classification may differ to those that are useful in compression. A hypothetical reason for this is that classification requires that weights produce binary 'on or off' activations whilst compression requires fine-grained control of the output amplitude. The former could favour ReLUs whilst the

[6]It is very possible that the authors investigated this hyperparameter and did not judge the results interesting but the wider trend is well documented (82) in any case.

latter might prefer PReLUs or GDNs.

There has been some research into the best activations for image compression: in 2018, Cheng et al. found that PReLU activations gave better results than ReLUs (87) while Ma et al. found that GDNs outperformed ReLUs (88) in the same year but, as far as we are aware, we are the first to compare PReLUs with GDNs for compression across a range of architectures. We found that PReLUs were generally more successful than GDNs although this was not true in all cases. For a detailed discussion of these results see section 5.1.2.

3.4.3 Skip Connections

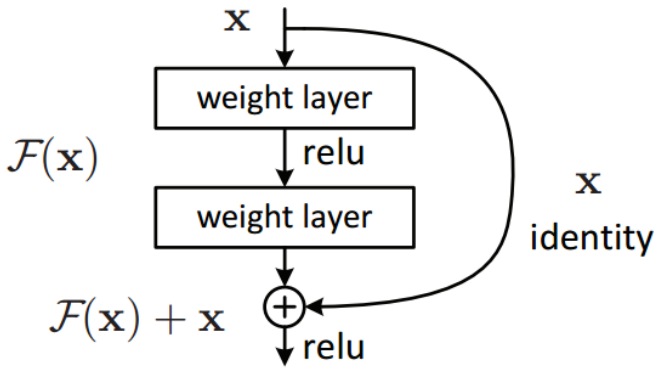

Figure 3.8: The basic residual block proposed by He et al. in their 2015 'ResNet' paper (14). The skip connection from the top to the bottom of the image ensures gradient is able to flow to earlier layers.

As a network gets deeper, the theoretical range of functions it is able to approximate increases. This comes with a tradeoff: the network will be more difficult to train as backpropagated gradients may struggle to reach early layers with sufficient magnitude for the system to learn. With this problem in mind, He et al. proposed the skip connection in 2015 in their ResNet paper (14). Skip connections place an identity shortcut around a matrix operation (or set of matrix operations) $\mathcal{F}(\mathbf{x})$ ensuring that, regardless of what takes place in $\mathcal{F}(\mathbf{x})$, gradient will be able to flow to earlier layers during training (see Figure 3.8). This enabled the training of *considerably* deeper networks. Before this paper was published in 2015, the state of the art on ImageNet classification was 'GoogleLeNet' (89), which features the 'Inception' module discussed in section 3.4.4. This network has 22 layers while the best performing ResNet variant chained together 76 of the blocks in Figure 3.8 to create a network with 152 layers.

Mathematically, the residual block in Figure 3.8 computes:

$$g(\mathbf{x})_{\mathbf{res}} = \mathcal{F}(\mathbf{x}) + \mathbf{x}$$

Note that this requires that $\mathcal{F}(\mathbf{x})$ has the same dimensions as input $\mathbf{x}$. There are many variants on this simple system. For example, the result of $\mathcal{F}(\mathbf{x})$ can be concatenated with $\mathbf{x}$ to give:

$$g(\mathbf{x})_{\mathbf{res_2}} = [\mathcal{F}(\mathbf{x}), \mathbf{x}]$$

In this case, there is no constraint on the output size of $\mathcal{F}(\mathbf{x})$.

Residual blocks (RBs) are exceptionally powerful for image compression and modern systems almost always use them in one way or another through a range of variations on the basic $g(\mathbf{x})_{\mathbf{res}}$ and $g(\mathbf{x})_{\mathbf{res_2}}$ building blocks (5) (70) (90) (91) (62) (12) (52) (65) (66) (67) (68). In fact, in recent years, these blocks are the unit on which architectural innovation occurs as repeatable blocks can be more easily transferred between problems than a macro network structure. In section 3.4.7 we will describe some of the more exotic variants that have had successes in image compression but first, it is necessary to describe other components of these systems including the use of parallel filters, multi-scale resolution and the attention mechanism.

3.4.4 Parallel Filters

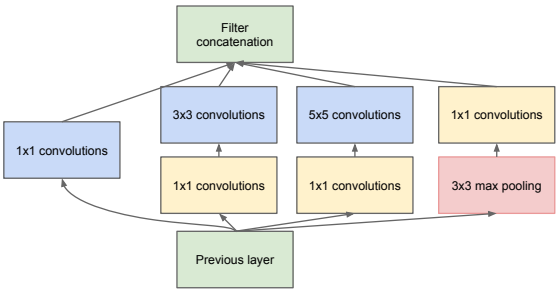

Figure 3.9: The Inception module proposed in (89). Three convolutional and one pooling operation, are performed in parallel and 1x1 filters bottleneck the inputs so that the features map size and number of parameters are reduced.

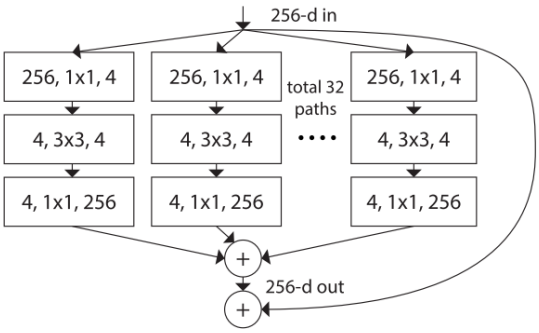

Figure 3.10: ResNeXt residual block in which the 'cardinality', or width can be increased to boost the capacity of the network (15). Unlike the Inception module, the ResNeXt block uses 3x3 filters only.

The problem with the neural network design space is that, choosing whether to employ even a binary attribute of the form 'use or do not use x' *doubles* the compute required for a hyperparameter grid-search. As more researches produce more architecture variants it becomes computationally unfeasible to search them all, even if you have Google's compute. So with this in mind, in 2015, Google researcher proposed the 'Inception' module (89) shown in Figure 3.9 as a method of reducing the decisions available to researchers. In the Inception module, 1x1, 3x3 and 5x5 convolutions are placed in parallel so that the network can 'choose' which combination of filters is relevant for the task in hand without the need for user input. This is referred to as the *split-transform-merge* paradigm. These modules are then stacked repeatedly in a deep network so that only the number of layers must be tuned.

1x1 convolutions

Another innovation of the Inception module was the use of 1x1 convolutions to bottleneck the inputs. This drastically decreases the computational cost by reducing both the input feature map size and the number of parameters. The ideas here were extended by others including notably the 'SqueezeNet' authors with their proposal of the Fire module (92) and the 'ResNeXt' authors who proposed the block of variable width shown in Figure 3.10 (15). In section 5.1 we evaluate a number of CAEs from the ResNeXt family and show that we can reduce the inference latency of our best network by a factor of x2.5 by using ResNext blocks instead of the vanilla residual blocks in Figure 3.8.

3x3 convolutions

The work in the Inception paper (89) was part of a wider trend at the time to reduce the convolutional kernel width K as computational cost scales in K^2 in two-dimensions and K^3 for three-dimensions. The key insight is that two consecutive 3x3 convolutions have the same receptive field as a 5x5 convolution but require just two thirds of the number of parameters and FLOPS. In three dimensions, two 3x3x3 filters requires just a third of the parameters and FLOPS compared with a single 5x5x5 filter[7]. In fact, it was shown in the VGG paper (93) that results on ImageNet were actually improved by replacing larger kernels with stacked 3x3 filters (most likely because of the additional activation functions) and this was imitated by the inception authors in their next publication of 2015 (94). In current state of the art systems in computer vision, kernels of size 3x3 (or lower) are almost always used.

3.4.5 Multi-scale resolution

Convolutional networks with small filters have had many successes but they have a key limitation: a single filter is only able to extract local patterns within a 3x3 pixel range. If there are two pixels, or groups of pixels, on opposite side of the image that have some shared meaning, there will not be a feature map with a receptive field containing both pixels until very deep in the network and by this point the feature map is (typically) very low resolution. Therefore, it is almost impossible for the network to make fine-grained decisions about pairs of spatially distant pixels.

In order to tackle this issue in the domain of medical image segmentation, the U-Net authors proposed[8] concatenating high and low-resolution features so that the network has access to multiple feature-map resolutions simultaneously[9]. In the U-Net design, shown in Figure 3.11, high resolution data from the encoder network

[7]These ratios are approximate as they are dependent on the size of the feature map which can change between layers.

[8]To our knowledge, the U-Net (64) authors were the first to propose using multi-scale features in a machine learning context but there had been a long history of using multiple resolutions in computer vision and segmentation in particular. The following paper from 2001 is a typical example (95).

[9]This sits within the residual framework as we are concatenating features after a skip connection ($g(\mathbf{x})_{\mathbf{res_2}}$ in section 3.4.3).

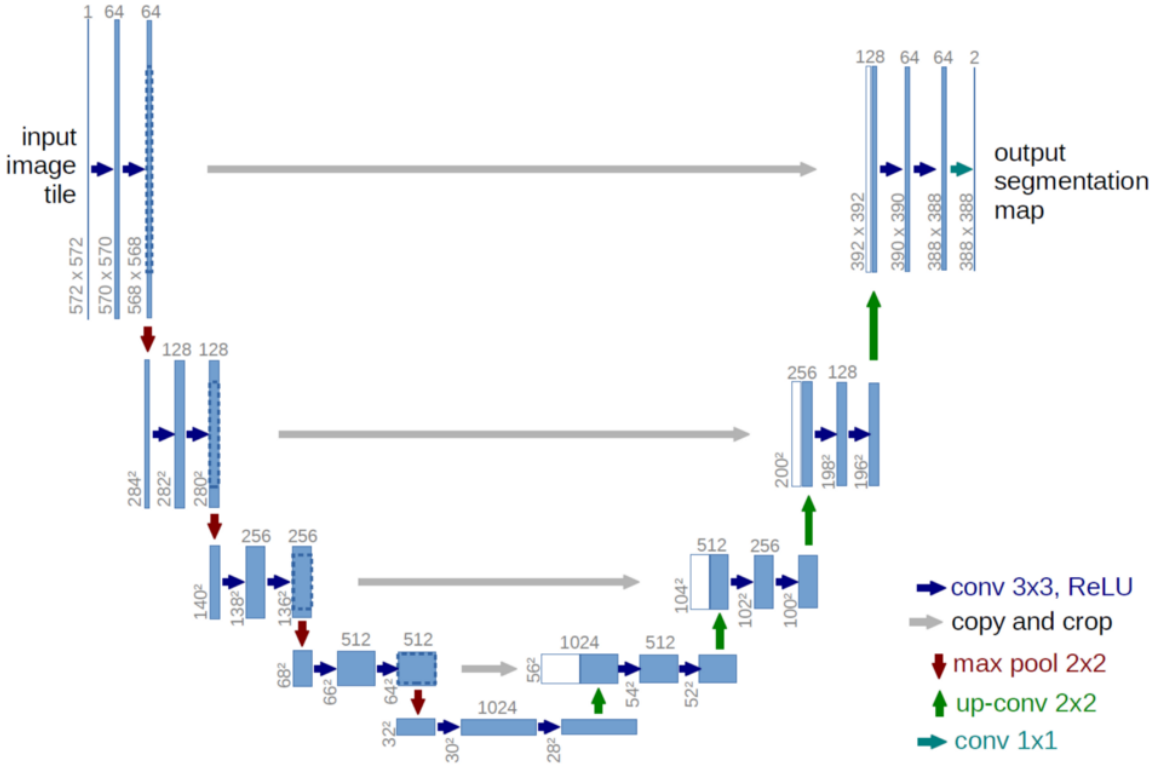

Figure 3.11: The U-Net architecture for image segmentation proposed in (64). The feature maps are successively down-sampled (red arrows) in the encoder to a latent dimension of 1024 and then up-sampled (green arrows) by the decoder to the original size. The decoder is provided with high resolution data from the encoder network.

is provided to the decoder network. We note that this design would be 'cheating' in a compression setting as higher resolution information cannot be made available during decoding because the cost of storing this data would negate any gains made by the compression system. However, the approach can be applied to compression by making multi-scale paths self-contained within either the encoder or decoder.

Multi-scale approaches have had substantial successes in image compression in recent years (66) (62) including for four of five top finishers in CLIC 2019 (12), (67), (52), (68). For example, Lu et al. employed the hierarchical decoder shown in Figure 3.12.

3.4.6 Attention

A shortcoming of the convolutional networks discussed so far is that they assign the same importance to a particular location or channel of the feature map regardless of the input. This can make it very difficult to learn complex features. To see why imagine the possible (but unlikely) case in which an ImageNet classification network included a channel in which the feature map encoded information about the type of animal fur in an image as a method of distinguishing between cats and dogs. In a

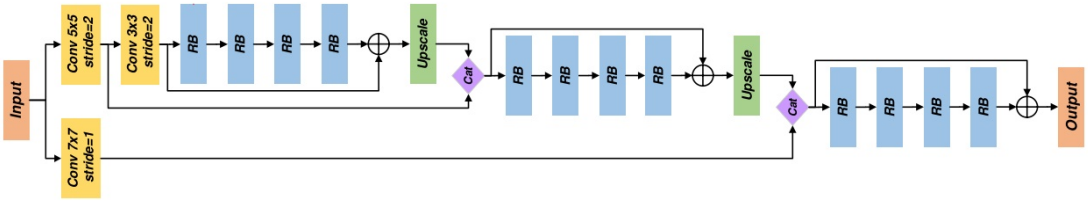

Figure 3.12: The decoder of Lu et al.'s fourth place entry to CLIC 2019 (52) in which features are available at multiple resolutions simultaneous.

network without an attention mechanism, this channel will be given equal importance in the final classification when the input is a cat as when it is an aeroplane. Clearly this channel is simply noise in the second case so the model will down-weight this sometime-useful 'fur' feature or, more likely, never learn it in the first place.

The attention mechanism solves this problem by focusing the network's 'attention' on relevant elements of the feature map (69). It was first proposed and used in NLP to increase the importance of different words in a sentence and has had a raft of successes in this field, including in Transformers where is is suggested that 'Attention is *All* You Need' (96). In imaging, there are two types of attention, *channel* attention which focuses on relevant channels as in the example above, and *spatial* attention which focuses on relevant locations in the image.

There are various types of attention $g(\boldsymbol{x})_{Att}$ but in all variants there is a 'trunk'[10] $T(\boldsymbol{x})$ of the same size as the tensor input $\boldsymbol{x}$ and an attention mask $M(\boldsymbol{x})$ such that:

$$g(\boldsymbol{x})_{Att} = T(\boldsymbol{x}) * M(\boldsymbol{x}) \tag{3.13}$$

where $*$ denotes element-wise multiplication. In many cases, $T(\boldsymbol{x}) = \boldsymbol{x}$ meaning that the attention mask is applied directly to the input. The mask elements $M_i(\boldsymbol{x})$ are soft assignments in the range $[0, 1]$ to indicate how much attention should be given to the corresponding trunk element $T_i(\boldsymbol{x})$. In theory the mask elements could also be hard assignments in $\{0, 1\}$ although, as far as we are aware, this is not used in practice. The granularity of $M(\boldsymbol{x})$ is usually one of the following:

i $M(\boldsymbol{x})$ is a full rank tensor in which each element of $T_i(\boldsymbol{x})$ has its own mask value $M_i(\boldsymbol{x})$.

ii For C_{in} channel inputs, $M(\boldsymbol{x})$ has C_{in} distinct values that are shared on a per-channel basis.

iii For an input with p pixels $M(\boldsymbol{x})$ has up to p distinct values but can also have $int\left(\frac{p}{k}\right)$ distinct values that are shared between neighbouring pixels in an image.

[10] This terminology is not used by all authors.

[11] Note that this heat-map is **not** the spatial attention mask. It is a visualisation of the pixels that

Figure 3.13: A visualisation of the effect of attention with CBAM (97) for classification on ImageNet using the Grad-CAM method (98)[11]. Image adapted from the CBAM paper (97).

3.4.7 Complex Residual Blocks

We are now in a position to describe the complex residual blocks that are regularly used in image compression, with a particular focus on the four blocks that have featured in successful CLIC entries from the last two years. These can be split into densely connected blocks and attention based blocks.

Densely Connected Blocks

A problem in deep residual networks like ResNet-152 (14) is that many of the useful features created by early RBs are not directly available to latter RBs meaning that similar calculations to create these features may occur multiple times in the same network. Densely connected networks (99) avoid this problem by providing the features calculated by each RB to *all* of its successors. This is achieved by concatenating each RB's input and output and providing this tensor as input to the next RB as shown in Figure 3.14a). Note that this results in a consecutively larger input from right to left in Figure 3.14a). **RDBs**, proposed by Zhang et al. place a limit on this growth by using 1x1 convolutions to return outputs of the same size as the original input (100). **GRDBs** or *grouped* RDBs take this one stage further by stacking multiple RDBs in sequence (13) as shown in Figure 3.14b). RDBs were used by the second place challenger in CLIC-2018 (91) and GRDBs were used by the second place finisher this year (65).

Attention-Based Blocks

CBAMs or Convolutional Block Attention Modules are residual blocks that add channel-wise and spatial attention sequentially (97) as shown in Figure 3.15. Despite their relatively complex structure, CBAM blocks have a small number of parameters as a

are most relevant to the classification. In the larger Figure from which this is taken, an equivalent heat-map is given for a vanilla ResNet-50 to demonstrate that it makes less focused use of its input pixel data.

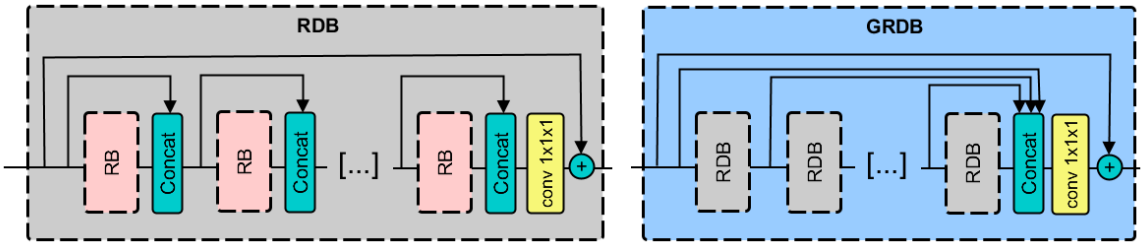

Figure 3.14: a) The Residual Dense Block (99), (101) and its extension b) The Grouped Residual Dense Block (13).

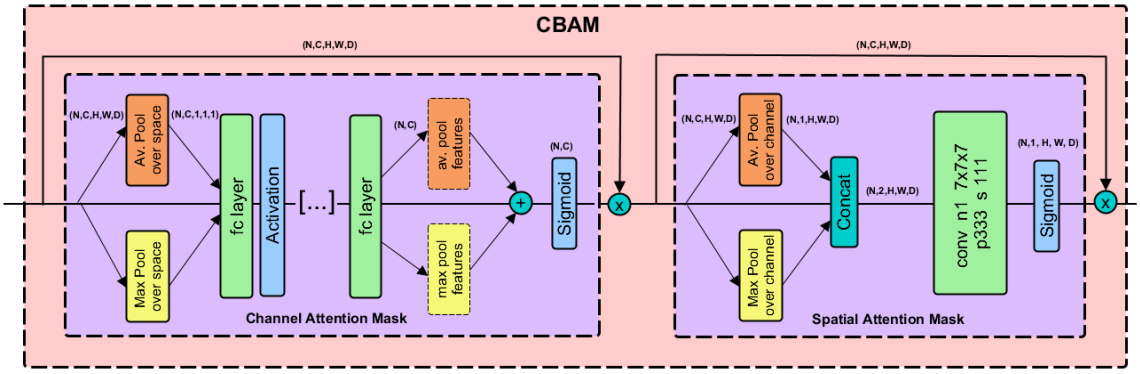

Figure 3.15: The Convolutional Block Attention Module (97). The channel mask $M_C(\boldsymbol{x})$ and spatial mask $M_s(\boldsymbol{x})$ are applied sequentially. These masks are broadcast to full dimensions (N, C, H, W, D) before their element-wise multiplication with the inputs $\boldsymbol{x}$. Note that in $M_C(\boldsymbol{x})$, the features from max pooling and average pooling are fed through the same fully connected network one after the other and the results are then added. 'conv n1 7x7x7 p333 s111' represents a convolutional layer with 1 channel, kernel size = (7, 7, 7), padding = (3,3,3) and stride = (1,1,1) and is specific to our implementation (although in some cases we found that kernel size = (3, 3, 3) was necessary to enable efficient training). Our CBAM has just two fully connected layers in $M_C(\boldsymbol{x})$.

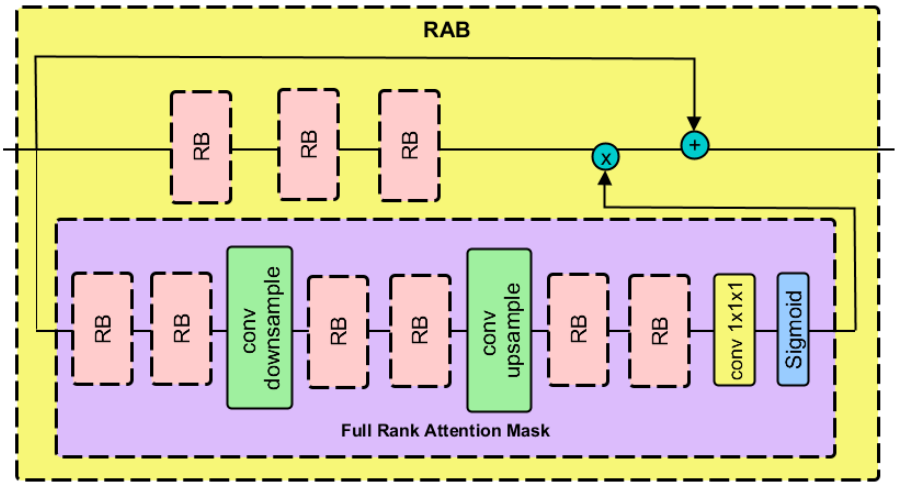

Figure 3.16: The Residual Attention Block proposed by (101) and utilised by (12). Note that unlike CBAMs, the trunk of RABs (yellow background) are not the identity mapping.

result of the pooling operations. The authors recommend placing CBAMs after every traditional RB as a lightweight method of adding attention to a network.

In comparison, **RABs** or Residual Attention Blocks, shown in Figure 3.16, compute an individual mask value for every element in the input of size (N, C, H, W, D) (101). This makes them considerably more expressive than CBAMs but, the tradeoff is that RABs add considerable overhead. By way of comparison, in our 3D implementations of these blocks with 64 input channels, CBAMs have just over 2,500 parameters while RABs have just under 2.5 million parameters. The winner of CLIC this year, (and our most successful system) used a number of RABs (12) whilst the second place entry utilised CBAMs (65). In section 5.1 we experiment using both blocks in tandem but find that the network accuracy is significantly reduced by this combination.

As an aside, note that the same authors: Zhang et al., proposed both RABs and RDBs in a four month period.

3.5 State-of-the-art Architectures

System details				Building Blocks							Equivalent model in
Year	CLIC pos.	Authors/Team	Paper	GDNs	Parallel Filters	Multi-scale	Attention	RDB	CBAM	RAB	section 5.1
2017	-	Theis et al.	(5)	×	×	×	×	×	×	×	∼ Backbone
2018	-	Mentzer et al.	(90)	×	×	×	×	×	×	×	ResNeXt3-27-1-vanilla
2018	-	Cheng et al.	(87)	×	✓	×	×	×	×	×	ResNeXt3-3-N-vanilla[§]
2018	-	Balle et al.	(6)	✓	×	×	×	×	×	×	∼ Tucodec (no RAB)
2018	1st	Tucodec	(62)	✓	✓	✓	×	×	×	×	Tucodec (no RAB)
2018	2nd	iipTiramisu	(91)	×	×	×	×	✓	×	×	RBD3NeXt-8-1-vanilla
2018	2nd	AmnesiackLite[*]	-	-	-	-	-	-	-	-	-
2018	3rd	ZTESmartVideo[*]	-	-	-	-	-	-	-	-	-
2018	4th	yfan	(66)	×	✓	✓	×	×	×	×	ResNeXt3-4-32-vanilla[‡]
2019	1st	Tucodec	(12)	✓	✓	✓	✓	×	×	✓	Tuocodec
2019	2nd	ETRI[†]	(65)	×	✓	×	✓	✓	✓	×	∼ Backbone + GRDN[▽]
2019	2nd	Joint	(67)	✓	✓	✓	×	×	×	×	Tucodec (no RAB)
2019	3rd	NJUVisionPSNR	(52)	×	✓	✓	×	×	×	×	∼ Tucodec (no RAB)
2019	4th	Vimicro	(68)	✓	✓	✓	×	×	×	×	Tucodec (no RAB)

Table 3.1: Successful CLIC entries and their precursors for 2018 and 2019. CLIC uses multiple compression metrics (PSNR, MS-SIMM etc) and while success on one metric generally implies success on another, this is not always true. When there is ambiguity, we have given the highest positioning a system achieved. In the final column we have provided the name of the system within the architecture search framework we use in section 5.1. A ∼ here implies our implemented system is similar but not equivalent.

[*] To our knowledge, these teams did not produce a publication detailing their approach.

[†] This system was also the fastest in 2019.

[§] The original model did not have skip-connections but was otherwise identical.

[‡] The authors use 'wide-activated residual blocks' in which the channel size is increased by a factor of 4 internally within the block. The ResNeXt authors (15) have shown that this is equivalent to their system with a cardinality of 4.

[▽] This system was not an end-to-end AE but used the VVC compression standard (102).

A summary of state of the art image compression CAE architectures[12] and an overview of their constituent elements is given in Table 3.1. We have implemented a variant on all of these systems according to a framework described in section 4.2. In the rest this section we will briefly describe the CLIC-2019 winner (12) and the runner up (65).

Tucodec

The 'Tucodec' team placed first in CLIC-2019 and 2018. Their 2018 design given in Figure 3.3 was an adaptation of a VAE proposed by Ballé et al. that employed GDN activations (6). The Tucodec team added a multi-scale path to the Ballé et al. architecture. In 2019, two of the top-five placing teams, 'Joint' (67) and 'Vimicro' (72), used a network that was virtually identical to this 2018 entry but the Tucodec team improved their design with the addition of RABs to aid encoding and decoding. A three-dimensional version of their encoder is given in Figure 3.17.

We note that the Tucodec authors assert incorrectly throughout their paper that they are using *non-local* attention blocks or 'RNABs' instead of the simpler local RABs (proposed in the same paper by (101)). We have used RABs in our implementation of their system.

[12] Specifically, the architectures that comply with our specification in section 3.2.

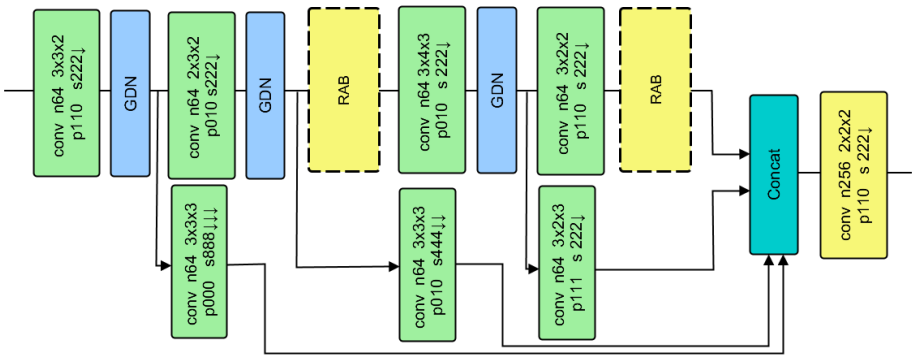

Figure 3.17: The Tucodec encoder (12). All convolutional values are specific to our 3D implementation. The decoder is identical except that, like their 2018 entry in Figure 3.3, it does not have the multi-scale path.

GRDNs

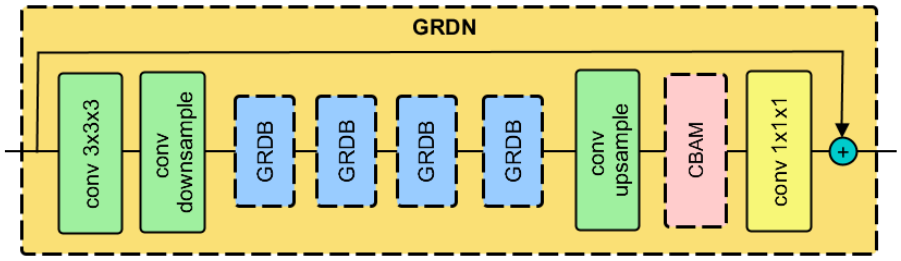

Figure 3.18: The GRDN block (100) in a series of GRDBs (see in Figure 3.14) are used with a CBAM module.

The authors who proposed GDRBs, suggested that they be used within a Grouped Residual Dense Network or GRDN, which also includes a CBAM attention module as shown in Figure 3.18. GRDNs have been used to give state of the art performance on image denoising (100) and were used by Cho et al. in their second place CLIC submission this year to remove compression artefacts from the reconstructed image (65). The remainder of Cho et. al's compression pipeline was not an AE but rather a compression standard 'Versatile Video Compression' VVC that is being developed to replace HEVC (102). As such, Cho et al.'s end-to-end solution was not appropriate in this work but we investigate the effectiveness of the GRDN in section 5.1.

Chapter 4

Contribution

In this chapter we will detail the contributions of this paper: in the following section 4.1, we describe our proposed data assimilation formulation including a discussion of its computational cost. In section 4.2 we describe the framework within which we searched for a suitable CAE design while in section 4.3 we give a brief overview of our Python module `VarDACAE`'s API and design. Finally, in 4.4 we give the details of our CAE training configuration.

4.1 'Bi-reduced space' formulation of DA

Our data assimilation formulation with AEs involves non-trivial changes to the incremental formulation in equations 2.17. We give its derivation below. A subtlety to note is that there are two 'reduced' spaces in this case: the reduced space of size S introduced by the CVT and the reduced space of size m introduced by the encoder-decoder framework. To avoid confusion, we will refer to the AE space as the 'latent space' (denoted with $\mathbf{z}$) and to the CVT space as the 'reduced space' (denoted with $\mathbf{w}$). As our method utilises both of these spaces we refer to it as the 'Bi-reduced space' formulation.

Firstly, our system requires that all data is mean-centred with respect to the historical mean $\bar{\boldsymbol{x}}^b$ which sets $\boldsymbol{x}^b = \mathbf{0}$ for all equations in section 2.2. The AE formulation from equation 3.1 then has many equivalent representations:

$$\begin{aligned} f(\delta\boldsymbol{x}_i) &= f(\boldsymbol{x}_i - \boldsymbol{x}^b) = f(\boldsymbol{x}_i) = \mathbf{z}_i \\ g(\delta\boldsymbol{z}_i) &= g(\mathbf{z}_i) = \boldsymbol{x}_i - \boldsymbol{x}^b = \boldsymbol{x}_i \end{aligned} \tag{4.1}$$

Mean-centering the data is necessary to ensure that our formulation is approximately equivalent to the Parish et al. CVT formulation (8) but has the added benefit of ensuring that encoder inputs are symmetrically distributed about $\mathbf{0}$. The true state can be reconstructed by adding $\bar{\boldsymbol{x}}^b$. In this paper, whenever the success of DA is evaluated, the true state is used.

In order to give our formulation we will first define the quantities f^o, $\boldsymbol{V}_l$, $\boldsymbol{d}_l$ and $\boldsymbol{R}_l$ where a subscript l implies the matrix or vector has been replaced by its latent-space equivalent.

Observation Encoder: f^o

f^o is an operator which maps from the full observation space of size M to the reduced observation space of size M_l such that:

$$f^o \coloneqq \boldsymbol{H}_l f \boldsymbol{H}^+ \in \mathbb{R}^{M_l \times M} \tag{4.2}$$

$$f^o = \quad f\boldsymbol{H}^+ \quad \in \mathbb{R}^{m \times M} \tag{4.3}$$

where $\boldsymbol{H}^+ \in \mathbb{R}^{n \times M}$ is the generalised inverse of $\boldsymbol{H}$ which maps from the observation space to the full space. This is an under-determined problem so there will be many equivalent $\boldsymbol{H}^+$ operators.
where $\boldsymbol{H}_l \in \mathbb{R}^{M_l \times m}$ is the latent space observation operator which maps from the latent space of size m to the latent observation space of size M_l. We have implicitly defined $\boldsymbol{H}_l \coloneqq \boldsymbol{I}$ which implies $M_l = m$. Another way of thinking of this is that the entire latent space is observable to us.

Latent $\boldsymbol{V}$: $\boldsymbol{V}_l$

$$\begin{aligned} \boldsymbol{V}_l = f(\boldsymbol{V}) &= \left[f(\delta\boldsymbol{x}_0^b),\ f(\delta\boldsymbol{x}_1^b),\ ...,\ f(\delta\boldsymbol{x}_S^b)\right] \\ \boldsymbol{V}_l &= \left[\mathbf{z}_0^b,\ \mathbf{z}_1^b,\ ...,\ \mathbf{z}_S^b\right] \in \mathbb{R}^{m \times S} \end{aligned} \tag{4.4}$$

Note that we are representing the information in matrix $\boldsymbol{V}$ of size $\mathbb{R}^{n \times S}$ in the CVT formulation in a matrix of size $\mathbb{R}^{n \times S}$. In our implementation $m \sim 0.0001n$ so this is an $\mathcal{O}(10^3)$ reduction in the size of our background covariance representation.

Latent $\boldsymbol{d}$: $\boldsymbol{d}_l$

$$\boldsymbol{d}_l \coloneqq f^o \boldsymbol{d} \in \mathbb{R}^m \tag{4.5}$$

Latent $\boldsymbol{R}$: $\boldsymbol{R}_l$

Recall that $\boldsymbol{R}$ is the covariance matrix of the observation error $\boldsymbol{\epsilon} = \boldsymbol{H}\boldsymbol{x}^b - \boldsymbol{y}$ and as such, for observations $\boldsymbol{y}$, it can be calculated as:

$$\boldsymbol{R} = \mathbb{E}[\ \boldsymbol{\epsilon}\boldsymbol{\epsilon}^T\ ] \tag{4.6}$$

$$\boldsymbol{R} = \sigma_0^2 \boldsymbol{I} \tag{4.7}$$

where the second line follows from the assumption that all observations are uncorrelated. We define the latent observation covariance matrix $\boldsymbol{R}$ in the same way:

$$\boldsymbol{R}_l \coloneqq \mathbb{E}[\ \boldsymbol{\epsilon}_l \boldsymbol{\epsilon}_l^T\ ] \tag{4.8}$$

$$\text{where} \quad \boldsymbol{\epsilon}_l = f^o \boldsymbol{\epsilon} \tag{4.9}$$

As the latent observation errors $\boldsymbol{\epsilon}_l$ are derived from uncorrelated full space errors, the latent observation errors $\boldsymbol{\epsilon}$ will also be independent of one another giving:

$$\boldsymbol{R}_l = \sigma_l^2 \boldsymbol{I} \tag{4.10}$$

where σ_l is the latent observation error standard deviation. We require the result in 4.10 for the proof below but it is worth noting that the uncorrelated observation assumption becomes more tenuous as M increases. In this work we consider the case when $M = n$ which definitely breaks the assumption. However, we can ensure that 4.10 holds in all cases with the additional assumption that **all** latent features are orthonormal to one another.

Orthonormal latent representation

It is worth making a brief comment on the conditions under which this will be true.

i Firstly, to obtain unit length features, batch normalisation could be used. Unfortunately, like Chen et al. (91), we found that batch normalisation greatly hampered the AEs ability to produce good reconstructions and therefore we do not use it on the backbone network of our models. For this reason, we do not make statements on the relation between the magnitude of σ_0 and σ_l in the above section (although in practice we treat them as equal).

ii Secondly, latent dimensions are not orthogonal in the general case but they will be 'approximately' orthogonal. To see why intuitively: if an AE is producing a good reconstruction, the latent hyperplane must span the majority of the data distribution manifold. If it is able to do this with a very small number of latent features then it must be rare for latent features to double-up and span identical areas of the data distribution. Consider additionally that PCA, which produces orthogonal latent modes, is a special case of an AE with linear activations, although admittedly in AEs the orthogonality criteria is relaxed. For a formal derivation of this equivalence see Baldi et al.'s paper: (103).

iii It is possible, although not explored in this work, to enforce latent orthogonal features with the use of a Variational Autoencoder (48). As discussed, this approach was not attempted here as it typically leads to poorer reconstructions but this is a clear avenue for future work.

Calculation of latent misfit $\boldsymbol{d}_l$

Our mean-centering constraint gives:

$$\boldsymbol{d}_l = f^o \boldsymbol{d} \tag{4.11}$$

$$\boldsymbol{d}_l = f^o(\boldsymbol{y} - \boldsymbol{H}\boldsymbol{x}^b) \tag{4.12}$$
$$\boldsymbol{d}_l = f^o\boldsymbol{y} \tag{4.13}$$

In this work, we are using synthetic data meaning that the full state is available as observation and we can use:

$$\boldsymbol{H} = \boldsymbol{H}^+ = \boldsymbol{I} \tag{4.14}$$

which implies:

$$\begin{aligned} \boldsymbol{d}_l &= f\boldsymbol{H}^+\boldsymbol{y} \\ \boldsymbol{d}_l &\equiv f(\boldsymbol{y}) \equiv f(\boldsymbol{x}^{obs}) \equiv \boldsymbol{z}^{obs} \\ \text{where} \quad \boldsymbol{y} &= \boldsymbol{x}^{obs} \in \mathbb{R}^n \end{aligned} \tag{4.15}$$

So we can simply use the encoder to obtain our latent misfit. However, to order to make our scheme useful operationally where not all points are available to observe, it will be necessary to train a network $f^o(\boldsymbol{y})$ that can estimate the latent misfit $\boldsymbol{d}_l$ from observations. We hypothesise that once you have a trained f, it should not require much additional effort to create a trained f^o by adding an implicit interpolation network $\boldsymbol{H}^+$ as shown in Figure 4.1.

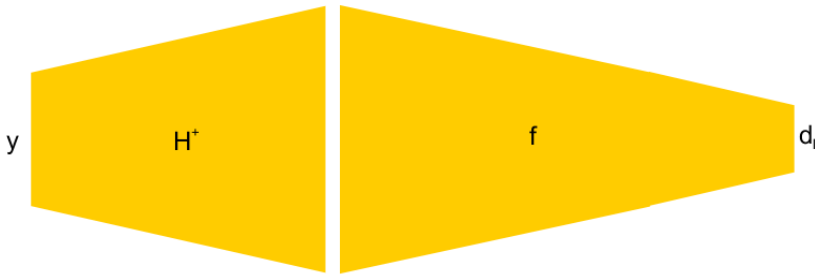

Figure 4.1: Scheme to create the f^o operator. Our hypothesis is that it should be possible to add the H^+ convolutional network and fine-tune the weights of f to create f^o.

Proposed 3D-VarDA formulation

Our proposed 'bi-reduced space' formulation is:

$$\begin{aligned} \mathbf{w}_l^{DA} &= \arg\min_{\mathbf{w}_l} J(\mathbf{w}_l) \\ J(\mathbf{w}_l) &= \frac{1}{2}\mathbf{w}_l^T\mathbf{w}_l + \frac{1}{2}\left\|\boldsymbol{d}_l - \boldsymbol{V}_l\mathbf{w}_l\right\|^2_{\boldsymbol{R}_l^{-1}} \end{aligned} \tag{4.16}$$

Note that, allowing for the different definitions of the latent variables $\boldsymbol{V}_l$, $\boldsymbol{d}_l$ and $\boldsymbol{R}_l$, this equation is identical to the mono-reduced space formulation in 2.17. In

fact, when comparing the two methods we were able to use the same cost-function and gradient implementations. This, coupled with the fact that we used the same minimisation routine (`scipy.optimize.minimize` in mode L-BFGS-B) gives us confidence that our comparisons between the execution-times of the two approaches is not biased towards either method as a result of implementation details.

Transformation to the full-space

Once equation 4.16 has been minimised in the reduced space to find $\mathbf{w}_l^{DA}$, the system can be restored to the full space in a two-stage transformation:

i Multiplication by $\boldsymbol{V_l}$ to move from the reduced space representation $\in \mathbb{R}^S$ to the latent space $\in \mathbb{R}^{\boldsymbol{m}}$.

ii Multiplication by $\boldsymbol{V_l}$ to move from the reduced space to the full space $\in \mathbb{R}^n$.

Overall this gives:

$$\delta \boldsymbol{x}^{DA} = g(\boldsymbol{V_l}\mathbf{w}_l^{DA}) \tag{4.17}$$

4.1.1 Proof of equivalence

Our proposed formulation is equivalent to the mono-space formulation in 4.16 in the sense that:

$$\mathbf{w}^{DA} = \mathbf{w}_l^{DA} \tag{4.18}$$

This is true under three assumptions:

i The AE compression is lossless meaning $g(f(\boldsymbol{x})) = \boldsymbol{x}$. This will not be exactly true but highlights the importance of a high-performing autoencoder.

ii All features in the latent representation $\mathbf{z}$ are orthonormal as discussed above.

iii The full *observation* space contains sufficient information to construct a good approximation of the full space $\boldsymbol{x}$. Note that if this holds, the ability to create the f^o operator in Figure 4.1 is assured. This assumption is discussed in more detail in the proof of Lemma 5.

We will state and prove a series of lemmas to produce the result in 4.18.

Lemma 1
The exact solution of $\mathbf{w}^{DA} = (\boldsymbol{I} + \boldsymbol{V}^T\boldsymbol{H}^T\boldsymbol{R}^{-1}\boldsymbol{H}\boldsymbol{V})^{-1}\boldsymbol{V}^T\boldsymbol{H}^T\boldsymbol{R}^{-1}\boldsymbol{d}$

Lemma 2
The exact solution of $\mathbf{w}_l^{DA} = (\boldsymbol{I} + \boldsymbol{V}_l^T\boldsymbol{R}_l^{-1}\boldsymbol{V}_l)^{-1}\boldsymbol{V}_l^T\boldsymbol{R}_l^{-1}\boldsymbol{d}_l$

Proof of Lemmas 1 and 2

The proof of both lemmas is the same as their cost functions are mathematically equivalent except for the fact that each operator in the first case is replaced by its equivalent latent operator in the second case (and $\boldsymbol{H}_l = \boldsymbol{I}$). Recall the reduced space cost function is:

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

which has gradient:

$$\nabla J(\mathbf{w}) = \mathbf{w} - \boldsymbol{V}^T\boldsymbol{H}^T\boldsymbol{R}^{-1}\big(\boldsymbol{d} - \boldsymbol{H}\boldsymbol{V}\mathbf{w}\big)$$

Setting this to zero and solving for $\mathbf{w}$ *will give the optimal value* $\mathbf{w}^{DA}$ *as required to complete the proof:*

$$\begin{aligned}\boldsymbol{V}^T\boldsymbol{H}^T\boldsymbol{R}^{-1}\boldsymbol{d} &= \big(\boldsymbol{I} + \boldsymbol{V}^T\boldsymbol{H}^T\boldsymbol{R}^{-1}\boldsymbol{H}\boldsymbol{V}\big)\mathbf{w}^{DA}\\ \mathbf{w}^{DA} &= (\boldsymbol{I} + \boldsymbol{V}^T\boldsymbol{H}^T\boldsymbol{R}^{-1}\boldsymbol{H}\boldsymbol{V})^{-1}\boldsymbol{V}^T\boldsymbol{H}^T\boldsymbol{R}^{-1}\boldsymbol{d}\end{aligned}$$

We can write the exact solutions of $\mathbf{w}^{DA}$ and $\mathbf{w}_l^{DA}$ in the following form:

$$\begin{aligned}\mathbf{w}^{DA} &= (\boldsymbol{I} + \boldsymbol{A})^{-1}\boldsymbol{b}\\ \mathbf{w}^{DA} &= (\boldsymbol{I} + \boldsymbol{A}_l)^{-1}\boldsymbol{b}_l\end{aligned}$$

$$\begin{aligned}\text{where}\qquad \boldsymbol{A} &= \boldsymbol{V}^T\boldsymbol{H}^T\boldsymbol{R}^{-1}\boldsymbol{H}\boldsymbol{V}\\ \boldsymbol{A}_l &= \boldsymbol{V}_l^T\boldsymbol{R}_l^{-1}\boldsymbol{V}_l\\ \boldsymbol{b} &= \boldsymbol{V}^T\boldsymbol{H}^T\boldsymbol{R}^{-1}\boldsymbol{d}\\ \boldsymbol{b}_l &= \boldsymbol{V}_l^T\boldsymbol{R}_l^{-1}\boldsymbol{d}_l\end{aligned}$$

Lemma 3

$$\boldsymbol{A} = \boldsymbol{A}_l$$

Lemma 4

$$\boldsymbol{b} = \boldsymbol{b}_l$$

The overall result 4.18 follows directly from Lemmas 3 and 4 but to prove these we need two further results.

Lemma 5

$$\boldsymbol{V}_l = f^o\boldsymbol{H}\boldsymbol{V}$$

Proof of Lemma 5

$$\begin{aligned}\boldsymbol{V}_l &\coloneqq f\boldsymbol{V} \\ \boldsymbol{V}_l &\approx f\boldsymbol{H}^+\boldsymbol{H}\boldsymbol{V} \\ \boldsymbol{V}_l &= f^o\boldsymbol{H}\boldsymbol{V}\end{aligned}$$

$\boldsymbol{H}^+\boldsymbol{H}$ acts is an information bottleneck operator in which only information contained in the observation locations is propagated from $\boldsymbol{V}$. Note that $\boldsymbol{H}^+\boldsymbol{H} \neq \boldsymbol{I}$ in general but that $f\boldsymbol{V} \approx f\boldsymbol{H}^+\boldsymbol{H}\boldsymbol{V}$ is a much weaker result that suggests:

- *The observation space of size M contains sufficient information to construct a good approximation of the latent representation. This is equivalent to assumption iii) above because, if the observation space contains all information in the full space, by assumption i) it should also contain all information in the latent space. To show why this might be true consider that there must be large redundancies in the full space in order for the CAE framework to have any success. We argue in the following section that, in all practical scenarios, $m, S < M$ so if a state of size m can contain all information of a state of size n it is not implausible that a state of size M could contain the same information. More concretely, in section 5.2 we demonstrate that the Arcucci et al. CVT with TSVD DA method suffers no degradation in accuracy when just 10% of the total state space is used as observations ($M = 0.1n$). There is only a 5% degradation when $M = 0.01n$.*
- *The reduced space of size S also contains sufficient information to construct the latent representation of size m. This condition is implied by the lossless compression assumption i) as the reconstruction of the full state passes through the reduced space and the latent space.*

Lemma 6

$$\boldsymbol{R}_l^{-1} = ((f^o)^T)^{-1}\boldsymbol{R}^{-1}(f^o)^{-1}$$

Proof of Lemma 6

$$\begin{aligned}\boldsymbol{R}_l &\coloneqq \mathbb{E}[\ \boldsymbol{\epsilon}_l\boldsymbol{\epsilon}_l^T\ ] \\ \boldsymbol{R}_l &= \mathbb{E}[f^o\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T(f^o)^T] \\ \boldsymbol{R}_l &= f^o\ \mathbb{E}[\ \boldsymbol{\epsilon}\boldsymbol{\epsilon}^T\ ]\ (f^o)^T \\ \boldsymbol{R}_l &= f^o\ \boldsymbol{R}\ (f^o)^T \\ \boldsymbol{R}_l^{-1} &= ((f^o)^T)^{-1}\boldsymbol{R}^{-1}(f^o)^{-1}\end{aligned}$$

where the third line follows because the observations are uncorrelated.

Proof of Lemma 3

We use Lemmas 5 and 6 to give:

$$\begin{aligned} \boldsymbol{A}_l &= \boldsymbol{V}_l^T \boldsymbol{R}_l^{-1} \boldsymbol{V}_l \\ \boldsymbol{A}_l &= \boldsymbol{V}^T \boldsymbol{H}^T (f^o)^T ((f^o)^T)^{-1} \boldsymbol{R}^{-1} (f^o)^{-1} f^o \boldsymbol{H} \boldsymbol{V} \\ \boldsymbol{A}_l &= \boldsymbol{A} \end{aligned} \tag{4.19}$$

Proof of Lemma 4

$$\begin{aligned} \boldsymbol{b}_l &= \boldsymbol{V}_l^T \boldsymbol{R}_l^{-1} \boldsymbol{d}_l \\ \boldsymbol{b}_l &= \boldsymbol{V}^T \boldsymbol{H}^T (f^o)^T ((f^o)^T)^{-1} \boldsymbol{R}^{-1} (f^o)^{-1} f^o(\boldsymbol{d}) \\ \boldsymbol{b}_l &= \boldsymbol{b} \end{aligned} \tag{4.20}$$

This completes the proof that $\mathbf{w}^{DA} = \mathbf{w}_l^{DA}$.

4.1.2 Advantages over TSVD: Theory

Eigenanalysis techniques such as PCA and TSVD are alternative methods of producing reduced space representations of data and, as discussed in previous sections, have been used canonically in preconditioned 3D-VarDA (9). Having summarised the key components of our proposed system, it is now possible to detail the theoretical reasons why our method produces a) higher quality compression and b) is faster than the traditional methods. For experimental verification of these advantages, see section 5.2.

Compression quality

For a given compressed size, a good CAE will produce reconstructions of higher quality than those using TSVD[1]. Note that we evaluate our systems on data assimilation performance rather than reconstruction quality but we find that systems that compress and decompress the state with a small loss of information perform better in DA. There are a number of reasons why the CAE approach produces better reconstructions than the truncated SVD approach:

i Redundancies in the training data distribution can be stored 'for free' by the decoder leaving space in the latent representation to encode the sample variations.

ii CAEs take location data into account and can therefore use properties like local smoothness to compress the input more efficiently[2].

[1] To calculate reconstruction quality for TSVD, it is necessary to first create the reconstructed state increment $\hat{\delta \boldsymbol{x}}$ by projecting to the reduced space with $\boldsymbol{V}^+$ and then returning to the full space with $\boldsymbol{V}$ as follows: $\hat{\delta \boldsymbol{x}} = \boldsymbol{V}\boldsymbol{V}^+ \delta \boldsymbol{x}$. The reconstruction L1 or L2 loss can then be calculated.

[2] Note that DA localisation approaches do utilise location information but not in a way that increases compression quality (31).

iii The latent features can be created from non-linear combinations of the inputs meaning they are likely to be of greater expressive quality than those in SVD in which linear combinations of the inputs are enforced. This latter approach is only optimal for compression when the data is drawn from a Gaussian distribution[3].

iv By design, in *Truncated* SVD, some of the information is intentionally discarded. This is not true in the CAE framework.

DA speed

The second advantage of the proposed method is its lower computational complexity in an online setting. In this analysis we distinguish between the offline costs and online costs. Here 'online costs' refers to any calculation that must take place when a new set of observations are made. 'Offline costs' are everything else which includes the TSVD computation and the CAE training.
The online complexities of the Parish et al. reduced space approach R_{on} (8) and our bi-reduced space approach B_{on} are:

$$R_{\text{on}} = \mathcal{O}(I_1 M^2 + nS) \tag{4.21}$$
$$B_{\text{on}} = \mathcal{O}(nm) \tag{4.22}$$

where I_1 is the number of iterations in the reduced space VarDA minimisation routine, M is the number of observations S is the reduced space size (which is equal to the size of the historical data sample) and m is the latent dimension size in our proposed method. We note that to achieve comparable accuracy with the two methods we will typically have $S > m$, (and $M > S$) so $R_{\text{on}} > B_{\text{on}}$. We derive these results in the following sections.

Note that our proposed method's online complexity B_{on} is independent of the number of observations M meaning it is never necessary to arbitrarily reduce the number of assimilated observations in order to meet the latency requirements of the system. We also note that the cost of training a CAE is *considerably* larger than of performing TSVD but as these operations occur offline they are not of primary importance in the creation of an operational system that, in the words of the MAGIC project organisers: "*allow rapid calculations for real-time analysis and emergency response*".
We discuss the derivation of the online and offline complexities in turn in the following sections but first give the encoder and decoder inference complexities.

$f(\boldsymbol{x})$ and $g(\mathbf{z})$ complexity

With input of size n and output of size m, the encoder and decoder inference complexities are $\mathcal{O}(nK) = \mathcal{O}(nm)$ for constant K.

[3]Another way of thinking of this is that PCA truncated at mode τ gives optimal reconstruction for all *linear* models of rank τ.

In the simplest possible encoder: a single fully connected layer, the complexity of mapping from the full to the latent space would be $\mathcal{O}(nm)$ exactly. The convolutional case is more complex but we think that logical CAE design choices lead to the same result. The encoder complexity is dominated by the layer in which the largest number of FLOPS take place so with 3D spatial inputs the complexity is $\mathcal{O}(nk^3C_1C_2)$[4] where k is the kernel width and C_1 and C_2 are the input and output channel sizes respectively when the feature map is at its largest. The choice of $K = k^3C_1C_2$ is an architectural design decision but K will never be $<< m$ as this would mean the encoder was introducing the information bottleneck at a location other than the latent space. Similarly, K should not be $>> m$ as this would negate the computational benefit of using convolutions over a linear network. This means K and m are of the same order and the above result follows[5]. A symmetric argument gives the decoder complexity as $\mathcal{O}(nm)$ too.

4.1.3 Computational Complexity Derivation: Online

There are two steps that contribute to the online-cost:

i The evaluation of the cost function and its gradient during the minimisation routine.

ii Restoring the calculated $\mathbf{w}^{DA}$ to the full space.

Lemma 7
The online complexity of step i) in the Reduced space method is:

$$\mathcal{O}(I_1(MS + M^2))$$

where I_1 is the number of iterations in the minimisation routine.

In the following sections we use the symbol ' $\cdot$ ' to make clear which operation we are considering at a given point in time.

Proof of Lemma 7

We repeat the cost function 2.17 here for convenience:

$$\mathbf{w}^{DA} = \arg\min_{\mathbf{w}} J(\mathbf{w}) \tag{4.23}$$

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{1}{2}\left\|\boldsymbol{d} - \boldsymbol{HV}\mathbf{w}\right\|^2_{\boldsymbol{R}^{-1}} \tag{4.24}$$

[4]The provided formula assumes that the largest number of FLOPS takes place in a layer before down-sampling has occurred which will be true in most systems. The result still follow without this assumption but including this complexity does not aid clarity.

[5]By accident, but in service of this point, in our Backbone network in 4.2, we have $k = (3, 3, 2)$, $C_1 = 1$ and $C_2 = 16$ which gives $K = 288$ which is the same size as our latent dimension m.

A naive implementation of 4.23 (and its derivative) would be dominated by the matrix multiplication $\boldsymbol{H} \cdot \boldsymbol{V}$ but, as $\boldsymbol{H}$ and $\boldsymbol{V}$ always appear together, this quantity can be precomputed and the minimisation complexity is independent of n. A single iteration of the VarDA minimisation has complexity $\mathcal{O}(MS + M^2)$ where the first term originates from $\boldsymbol{HV} \cdot \mathbf{w}$ while the second is from $(\boldsymbol{d} - \boldsymbol{HV}\mathbf{w})^T \cdot (\boldsymbol{d} - \boldsymbol{HV}\mathbf{w})$ where we are assuming $\boldsymbol{R}$ is diagonal. With I_1 iterations this gives Lemma 7.

Lemma 8
The online complexity of step i) in our bi-reduced space method is:

$$\mathcal{O}(I_2(mS + m^2))$$

where I_2 is the number of iterations in the minimisation routine and m is the latent dimension size.

Proof of Lemma 8

The argument to find the complexity of minimising the bi-reduced space cost function is almost identical to Lemma 7 above except that we replace $\boldsymbol{HV} \in \mathbb{R}^{M \times S}$ with $\boldsymbol{V}_l \in \mathbb{R}^{m \times S}$ in the cost function. Altering these dimensions gives the required result.

Lemma 9
Restoring the calculated $\mathbf{w}^{DA}$ to the full space in the mono-reduced space formulation is in $\mathcal{O}(nS)$.

Proof of Lemma 9

This is dominated by the product $\boldsymbol{V} \cdot \mathbf{w}^{DA}$ which is in $\mathcal{O}(nS)$.

Lemma 10
Restoring $\mathbf{w}_l^{DA}$ to the full space in the bi-reduced space formulation is in $\mathcal{O}(nS)$.

Proof of Lemma 10

This requires computing $\boldsymbol{V_l} \cdot \mathbf{w}_l^{DA}$ followed by $g \cdot \boldsymbol{V_l}\mathbf{w}_l^{DA}$ which has complexity $\mathcal{O}(mS + nm) = \mathcal{O}(nm)$.

This gives an an overall **reduced space online complexity** of:

$$R_{\text{on}} = \mathcal{O}(I_1(MS + M^2) + nS) \tag{4.25}$$

and a **bi-reduced space online complexity** of:

$$B_{\text{on}} = \mathcal{O}(I_2(mS + m^2) + nm) \tag{4.26}$$

Hence, the key comparison between the online complexity of the two methods is

down to the relative sizes of variables I_1, I_2, M, n, m and S. We assert that in most practical cases:

$$R_{\text{on}} > L_{\text{on}} \tag{4.27}$$

As many of these are user-chosen parameters we cannot prove that 4.27 always holds but we will make concrete arguments about their ranges in practical settings.

Argument 1

$I_1 > I_2$ in the majority of cases.

Justification: Argument 1

Following the discussion on optimisation in section 2.2.3 we have good reason to believe that the problem is better conditioned in the bi-reduced space than in the reduced space as for exactly the same reasons that it is better conditioned in the reduced space in comparison with the full space. See (37) for more information.

Argument 2

$M > S$ in the vast majority of cases.

Justification: Argument 2

The Met Office uses $M = 0.01n = 10^7$ (2). They employ a combination of KFs and VarDA approaches (30) but in the VarDA scheme, it is implausible that they would use anything close to 10^7 historical background estimates as this would be a matrix of size $10^7 \cdot 10^9 = 10^{16}$. We believe the same constraints will hold in all practical scenarios even if the number of observations is relatively small: $M \sim 0.00001n$.

Argument 3

$S > m$ for useful systems.

Justification: Argument 3

We found that a value of m that was a factor of x2.5 smaller than S still gave superior DA accuracy to the traditional method (even when $M = n$).

Argument 4

$B_{on} = \mathcal{O}(nS)$ *in the vast majority of cases.*

Justification: Argument 4

According to Argument 3 we have:

$$B_{on} = \mathcal{O}(I_2(mS + m^2) + nm)$$
$$B_{on} = \mathcal{O}((I_2 S + n)m)$$

We found $I_2 = \mathcal{O}(10)$ *meaning* $I_2 S << n$ *and hence* $B_{on} = \mathcal{O}(nm)$*. Concretely, with our values of* $m = 288$ *and* $S = 791$ *we found that upwards of 97% of the execution time of the bi-reduced DA procedure was restoring* $\mathbf{w}_l^{DA}$ *to the full space.*

Combining **Arguments 2-4**, gives the stated results in the previous section:

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

and since we argue that $n < S$, R_{on} is strictly greater than L_{on}. When M is large $M \geq \mathcal{O}(0.01n)$ (which is often crucial for good DA accuracy) we can go further than this:

Argument 5

When $M > \mathcal{O}(0.05n)$*,* $R_{on} = \mathcal{O}(I_1 M^2)$.

Justification: Argument 5

Clearly the exact values here will vary from one implementation to another but we found that with $M = (0.01n)$*, steps i) and ii) had approximately equal execution time but when* M *rose much above this, the minimisation term dominated as a result of the quadratic complexity.*

We investigate the negative effect of using fewer observations on DA accuracy in section 5.2.

4.1.4 Computational Complexity: Offline

In comparison, the offline costs are much larger for the proposed method in comparison with the formulation with TSVD:

Reduced space offline cost

$$R_{\text{off}} = \mathcal{O}(nS(M+S)) \tag{4.28}$$

Bi-reduced space offline cost

$$B_{\text{off}} = \mathcal{O}(nSEm) \tag{4.29}$$

for E epochs of training. We prove these results below.

Typically, $Em > M, S$ so $R_{\text{off}} < B_{\text{off}}$. In practice we found that $R_{\text{off}} << B_{\text{off}}$. Training a model took something on the order of 10-20 hours on a GPU whilst TSVD required approximately 10 minutes on the CPU.

Derivation of 4.28

Calculating SVD for $\boldsymbol{V} \in \mathbb{R}^{n \times S}$ where $S < n$ has complexity of $\mathcal{O}(S^2 n)$ (104). It is also necessary to precompute $\boldsymbol{H} \cdot \boldsymbol{V_\tau}$ which is in $\mathcal{O}(MnS) =$ giving the overall complexity in 4.28.

Derivation of 4.29

In training for E epochs, with S historical samples, the encoder/decoder must be evaluated ES times giving a complexity of $\mathcal{O}(SEmn)$. Before performing DA we must also precompute $\boldsymbol{V}_l = f \cdot \boldsymbol{V}$ which is in $\mathcal{O}(nmS)$ but this is comparatively small so the offline cost is as in equation 4.29.

4.2 Architecture Search Framework

In our attempt to find a good CAE architecture, we were concerned that the successes of 2D image compression systems might not always be transferable to 3D spatial inputs. As a result, we aimed to minimise this risk by investigating as wide a range of SOTA architectures as possible. We created a framework within which it was feasible to systematically search an architecture space that approximately encompasses the design of every top-5 CLIC finisher (see Table 3.1) from the last two years. In order to achieve this, it was necessary to make small alterations to some of the original systems. As such, although we found that our implementation of the Tucodec system was vastly superior to our implementations of the other systems, there is a possibility that our small design variations mean that another system should have performed even better. We were not unduly worried by this possibility as our aim in this process was to find systems that performed well rather than make exact comparisons between image compression networks.

In our search we enforced the following design constraints:

i All encoder-decoders are symmetric to reduce the design complexity. As a result, in subsequent sections, we refer to properties of encoders only on the basis that their decoder counterpart has equivalent mirrored properties[6].

ii All systems have a fixed latent dimension size of 288 to enable like-with-like comparisons between models. A size of 288 (which is a reduction is state size by approximately three orders of magnitude) was chosen because preliminary investigations suggested it gave good results.

iii All convolutions have kernel size 3 or less.

iv Convolutional down-sampling parameters are generated by our `ConvScheduler` class that has priorities of:

   (a) Avoiding addition of padding in later encoder layers as these can introduce artefacts in the reconstructed state.

   (b) Avoiding creation of feature maps that are not centred on the input as these are much harder to reconstruct in the decoder. This is achieved by refusing stride, s, padding, p and kernel size, k combinations that, when acting on input of width W, result in rounding in the floor operation when calculating the output size $= \left\lfloor \frac{W-k+2p}{s} \right\rfloor + 1$.

v Batch Normalisation is not used in the down-sampling backbone of the CAE architectures because preliminary investigation showed that it resulted in reconstructions of considerably poorer quality[7]. This finding was also made by Chen et al. (91).

[6]The single exception to this is the Tucodec architecture in which the multi-scale path is removed in the decoder.

vi Batch Normalisation *is* used in the residual blocks of the network. This was necessary to prevent activation and gradient overflow in deep networks.

For all models we investigate ReLU, GDN and PReLU activations and four types of RB: vanilla and NeXt RBs each with and without the lightweight CBAM module (see Figure 4.2). If present, CBAMs are placed after the original RB. Note that vanilla RBs have considerably more parameters than NeXt RBs: for the 32 channel input versions shown in Figure 4.2, NeXt blocks have 2k parameters while vanilla blocks have almost 60k parameters.

We note that a majority of encoders in the systems of Table 3.1 fit into one of two categories. Either they are very closely based on the design of the Tucodec 2018 entry in Figure 3.17 or they alternate between residual feature extraction and fully convolutional down-sampling operations. We can capture most of the variation in the second category with the backbone encoder architecture shown in Figure 4.3. As we have already described the Tucodec system in detail, in the rest of this section we describe our Backbone network and its variants.

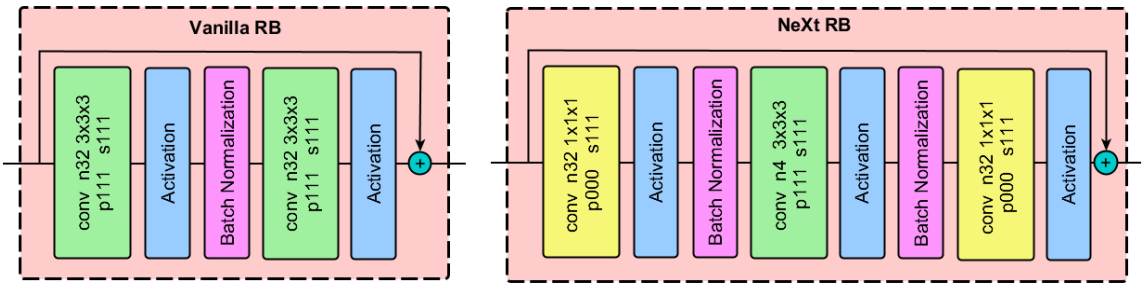

Figure 4.2: The two basic RBs we evaluated. We also investigated the effect of placing CBAM blocks after each of these. Note that NeXt RBs use 1x1 convolutions to bottleneck the inputs.

4.2.1 Backbone

Our backbone design, without any 'optional blocks' is a fully convolutional network with seven layers, the latter five of which down-sample the feature map. It is very similar to the encoder of Theis et al. from 2017 that was used in the first system which outperformed JPEG compression (5)[8]. The design is also relatively similar to a single path of Cheng et al.'s parallel convolutional filter network (87). The exact

[7]We hypothesise that this degradation in quality was likely the result of loosing batch-specific averages that are crucial to reconstruction. In our work this problem was likely exacerbated by the fact that, as a result of memory pressures created by 3D input data, we were using small batch sizes of just 16 or 8 meaning the batch statistics have high variance.

[8]The authors decoder was not symmetric to their encoder and contained fewer layers and also that they used 5x5 convolutions while we only use 3x3 kernels here.

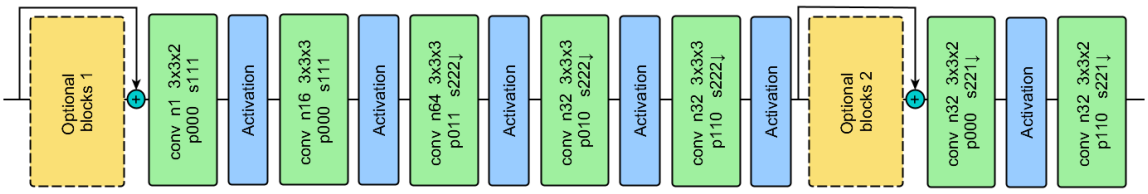

Figure 4.3: Backbone encoder architecture. Convolutional parameters are specific to our input data with dimensions (C, Hx, Hy, Hz) = (1, 91, 85, 32) and latent dimensions of (32, 3,3,1). For 'backbone' models, the optional block is empty. This architecture is loosely based on that in (5).

number of layers and convolutional parameters given in Figure 4.3 were chosen after an exploratory phase in which we compared a range of fully convolutional designs with between 5 and 11 layers[9].

The backbone is responsible for down-sampling while the optional blocks can introduce innovative feature extraction mechanisms. We require that any added blocks do not change the feature map size and are bypassed with a skip connection so that, at least in theory, they cannot hinder the down-sampling process. By placing a GRDN withing the first optional block, we can replicate the work in (65). The decision to placing the second optional block between the 5th and 6th layers creates a network similar to that in (90) and is compromise between early placement which enables the blocks to have a larger effect on the system and late placement in which the computational cost is lower as the feature maps are a small fraction $\left(\text{approximately } \frac{1}{20}\right)$ of their original size.

In section 5.1 we evaluate four variants on this backbone that are described in Table 4.1. The RAB-L variant is investigated in an attempt to separate the Tucodec model's success from its use of RABs and, as discussed, the GRDN model follows the work of (65). The ResNeXt variant requires slightly more explanation which is given in the following section.

Model Name	Optional Blocks 1	Optional Blocks 2
ResNeXt-L-N	None	L ResNeXt layers, cardinality N
RAB-L	None	L RABs
GRDN	1 GRDN	None
Backbone	None	None

Table 4.1: Variants on the backbone encoder network in Figure 4.3 that are evaluated in section 5.1.

[9]We compared approximately 20 candidate architectures created with our `ConvScheduler` according to their L2 reconstruction errors. We set the learning rate by cross-validation and used activation functions $\in \{\text{relu}, \text{leaky relu}\}$. During this phase we found that batch normalisation and dropout both harmed performance.

4.2.2 ResNeXt variant

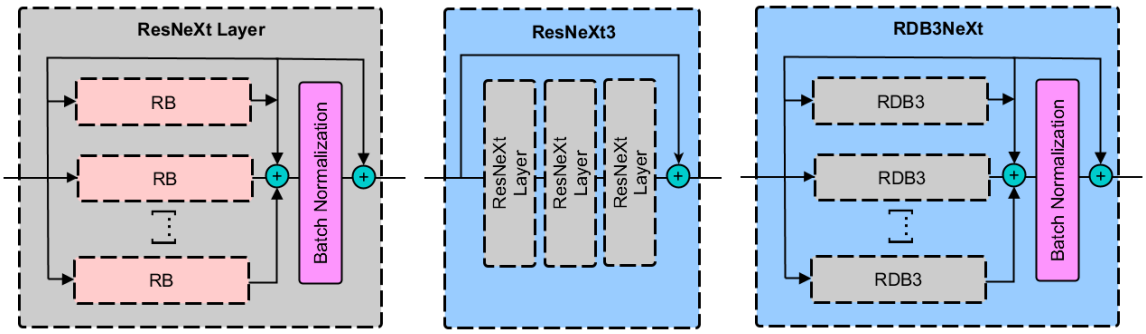

Figure 4.4: a) A single ResNeXt layer, repeated from Figure 3.10 for clarity (15). The ResNeXt cardinality describes the number of RBs in each layer, **b)** three stacked ResNeXt layers with an extra residual connection, and **c)** a ResNeXt layer with RDBs instead of simple RBs. Each 'RDB3' has 3 RBs.

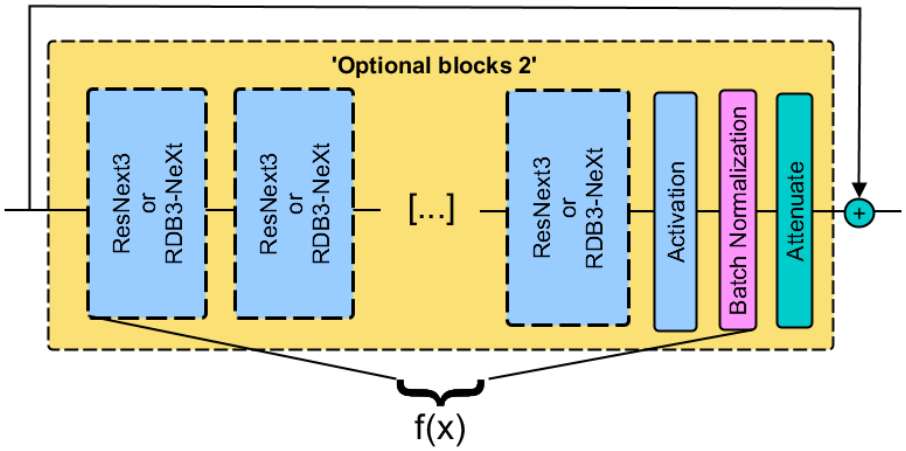

Figure 4.5: Our ResNeXt variant in which ResNeXt layers are grouped in threes. The residual attenuation coefficient in green is applied before leaving the block.

We found that placing a flexible variant of the ResNeXt system (15) within the second optional block of our backbone was sufficient to describe almost all of the top non-Tucodec-based CLIC entries. This is true despite the fact that vanilla ResNeXts are not traditionally viewed as image compression networks. In order to include the Chen et al. 2018 entry which used RDBs (91), each with three RBs, we extended the ResNeXt system to allow these building blocks as shown in Figure 4.4c). In order to make our vanilla ResNeXt variants comparable with these 'RDB3s', we added an extra skip connection over every third ResNeXt layer as in (90).

Within this system we refer to an architecture as:

'RBD3NeXt-L-N-RB' or 'ResNeXt3-L-N-RB'

for an encoder that consists of the ResNeXt variant in Figure 4.5 with L layers each of cardinality N arranged in either the ResNeXt3 or the RBD3NeXt structure with residual blocks of type RB all embedded within the second optional block of our backbone in Figure 4.3. When the backbone network is included, these encoders have (L + 7) layers.

In this way, Chen et al.'s encoder can be described as a 'RBD3NeXt-8-1-vanilla' (91) while Mentzer et al.'s is a 'ResNeXt3-27-1-vanilla' (90). As an aside, we note that this latter system, which was also utilised very successfully in (105), has a total of 33 convolutional layers making it almost identical to a ResNet-34 (14) without the final fully connected layer. This influence is not referenced by the authors which is a considerable oversight given the ubiquity of ResNets in early 2018.

By placing the CLIC entries within this structure, the landscape *between* the entries in Table 3.1 becomes available to search. In section 5.1 we evaluate the grid search of options within this space and find that 27 layers of width 4 RDB3s blocks (with CBAMs) perform best. This design is unlike any CLIC entry meaning that we would not have found it by simply following the examples in the literature. We believe that this validates our architectural search approach.

Attenuation coefficient

We found that it was difficult to train ResNeXt variants with large cardinality but, as the backbone trained easily, it was clear that the new residual blocks were interfering with the backbone's ability to down-sample the inputs. Therefore we introduced a residual attenuation coefficient α at the exit to the block shown in 4.5 such that the computed function is:

$$g(\mathbf{x}) = \mathbf{x} + \alpha\ f(\mathbf{x}) \tag{4.30}$$

α was initialised to 0.05 at the start of training and then updated with the other network parameters. This down-weights the ResNext block's importance initially so that the backbone has time to learn a good compression.

4.3 Software

As part of this work we have produced an open-source, well-tested Python module `VarDACAE` which is available on Github at: `https://github.com/julianmack/Data_Assimilation`. This module can be used to easily replicate our experiments, load our trained models or train CAEs for any 3D-Variational DA problem. In the following section we describe the API before moving onto a discussion of key implementation details in section 4.3.2.

```
from VarDACAE import TrainAE, BatchDA
from VarDACAE.settings.models.CLIC import CLIC

model_kwargs = {"model_name": "Tucodec",
                "block_type": "NeXt", "Cstd": 32}

settings = CLIC(**model_kwargs)
expdir = "experiments/"      #dir to save results/models

trainer = TrainAE(settings, expdir, batch_sz=16)
model = trainer.train(num_epochs=150)

#evaluate DA on the test set:
results_df = BatchDA(settings, AEModel=model).run()
```

Figure 4.6: Training and evaluating the Tucodec model is very straightforward. 'Cstd' sets the number of channels in the main body of the Tucodec model.

4.3.1 API

The software is designed so that a user can define, train and evaluate a CAE for data assimilation on Fluidity data with just a few lines of code as shown in Figure 4.6. The API is based around a monolithic `settings` object that is required to initialise key classes like `TrainAE` and `BatchDA`. A `settings` instance defines all configuration parameters including the CAE model, the VarDA constants and the seed. This single point of truth is used so that an experiment can be repeated exactly by simply loading a pickled `settings` object.

It is also straightforward for a user to extend our system to use their own data. To do this they must inherit from the default `GetData` class and override the `get_X(..)` method as shown in 4.7. For more information on how to change model definitions, load trained models or run our experiments, see the repository `README.md`.

4.3.2 Implementation

`VardDACAE` comes with a test suite that can be run from the home directory using `pytest` or equivalent. As we were always planning to provide an open source version of our work, we endeavoured to keep the design modular and create good documentation for all experiments. `VardDACAE` has very few external library dependencies: all models are trained in Pytorch and `vtk` and `pyevtk` are used to read and save Fluidity data. We also use `scipy` and `numpy`, although the latter is a torch and scipy dependency.

```
from VarDACAE import GetData
from VarDACAE.settings.models.CLIC import CLIC

class NewLoaderClass(GetData):
    def get_X(self, settings):
        "Arguments:
               settings: (A settings.Config class)
        returns:
            np.array of dimensions B x nx x ny x nz "

        # ... calculate / load or download X
        # For an example see VarDACAE.data.load.GetData.get_X"""
        return X


class NewConfig(CLIC):
    def __init__(self, CLIC_kwargs, opt_kwargs):
        super(CLIC, self).__init__(**CLIC_kwargs)
        self.n3d = (100, 200, 300)  # Define input domain size
                                    # This is used by ConvScheduler
        self.X_FP = "SET_IF_REQ_BY_get_X"
        # ... use opt_kwargs as desired



CLIC_kwargs =  {"model_name": "Tucodec", "block_type": "NeXt",
                "Cstd": 64, "loader": NewLoaderClass}
                # NOTE: do not initialize NewLoaderClass

settings = NewConfig(CLIC_kwargs, opt_kwargs)
```

Figure 4.7: To use your own data, simply define a `NewLoaderClass` and `NewConfig` as shown. If the data does not fit into CPU memory as a `numpy` array then you should override `NewLoaderClass.get_train_test_loaders(...)` to define and return your own torch `DataLoaders`.

We adapted open-source implementations of GDN activations by jorge-pessoa (`https://github.com/jorge-pessoa/pytorch-gdn`), CBAM residual blocks by Jongchan (`https://github.com/Jongchan/attention-module/blob/master/MODELS/cbam.py`) and `torchvision`'s DenseNet (`https://github.com/pytorch/vision/blob/master/torchvision/models/densenet.py`). Our alterations meant the the above implementations could a) accept 3D spatial inputs and b) in the latter two cases, utilise a range of activation functions and residual blocks in order to facilitate the experiments in section 5.1[10]. `VarDACAE` is distributed under the MIT License. We defer the

[10]In addition, we altered the batch normalisation and activation order implemented in the

ethical considerations of software licensing to Appendix D but note that we found it necessary to obtain written consent to use the CBAM implementation above as it was not distributed with a licence.

4.4 Training Configuration

4.4.1 Data

We used simulated data from a single run (988 time-steps) of the open-source, finite-element, fluid dynamic software Fluidity on a small domain in Elephant and Castle in South London. This system had 100,040 states spread over a region of size (x, y, z) = (700m, 650m, 250m). The data was placed in time-step order and the first 80% was used for the training set[11]. All of the data was normalised using the training data statistics. In order to ensure a fair comparison with traditional TSVD methods, the reduced space $\boldsymbol{V}_r$ was calculated using the training set only.

Fluidity uses an adaptive unstructured mesh in order to provide high resolution in regions of interest without requiring this same resolution in locations where there is very little variation. This is not an appropriate input for a CAE as convolutional kernels work on the assumption that adjacent states are equally spaced. Therefore, we used the `vtk` Python package to interpolate between points in the unstructured mesh and create a regular 3D grid. We found that keeping the number of points constant (at 100,040) resulted in a large amount of detail being lost as the regions of interest were under-sampled. As such, we up-sampled the field by a factor of 2.5 to produce 247,520 evenly spaced points in the shape (91, 85, 32). When evaluating the reduced space approach, this interpolated input was flattened before use.

4.4.2 Regularisation

With a view to increasing model generalisation, we investigated a number of regularisation techniques including data augmentation, dropout and weight decay. This is good practice in all machine learning systems but was particularly important in this case as we had such a small number of training examples. We found that dropout harmed performance, even when it was applied channel-wise, and only to latter layers as recommended in (106). Similarly, preliminary experiments showed that weight decay resulted in a small degradation in performance.

As such, data augmentation was our only method of regularisation. We did not find any augmentation strategies for physical fields in the literature and decided that the

DenseNet module so that (BN $\rightarrow$ ReLU $\rightarrow$ conv $\rightarrow$ BN $\rightarrow$ ...) was replaced with BN $\rightarrow$ conv $\rightarrow$ ReLU $\rightarrow$ BN.

[11] We did not use a validation data-set in this case due to the small number of available samples. This may result in slight over-fitting during our architecture search but we preferred this to reducing our training set size.

only appropriate augmentation strategy was one borrowed from imaging: 3D 'field-jitter' (the 3D mono-channel equivalent of colour-jitter). This involves injection of normal noise of amplitude $p\boldsymbol{\sigma}$ at r of the state locations where $p, r \in [0, 1]$ and $\boldsymbol{\sigma}$ is the state standard deviation. This strategy may result in local violations of mass and momentum conservation laws but, as shown in Figure 5.3 in section 5.1.4 this did not appear to affect the DA accuracy. For a description of the values of r and p that we experimented with, see Appendix B.

We avoided cropping the inputs because, as the state space is of fixed size, there is no benefit in the CAE learning to compress variable size inputs. Similarly, it was also not appropriate to flip the inputs horizontally as there are buildings in our domain which should stay fixed. Finally, we did not use the recommended medical imaging augmentation technique of plastically deforming the inputs (64) because while this is appropriate for body tissue, we judged it likely to produce highly non-physical pressure fields.

4.4.3 Training Duration

Our Backbone architecture takes approximately 15 hours, and 400 epochs to converge on a NVIDIA Tesla K80. Some the more heavyweight models take upwards of 40 hours. In order to conserve our resources and reduce the design iteration time, we used a maximum of 150 epochs during our architecture search which amounted to an 6-15 hour period. Most systems had not converged by this point but we found that, for the sample of models that we trained to convergence, the performance ordering was almost completely unchanged after 150 epochs (see Figures 5.3 and 5.2[12]). As such, all comparisons between AE architectures in section 5.1 are made with this constraint and the quoted data assimilation Figures in these sections should be used for comparative purposes only. We note that this may bias results towards smaller models that train more quickly but, all things considered, we would prefer a bias in this direction. We only trained a single model for each configuration so only large performance differences should be given any importance.

4.4.4 Loss function and evaluation metrics

As discussed in section 3.1, all networks were trained with L2 reconstruction error rather than L1 loss as our preliminary work suggested these gave models that performed consistently better for DA. We hypothesised that this was because outliers, which are reduced more with the L2 than the L1 loss, are particularly damaging to DA. However, we did not investigate this further and there may be alternative reasons for this difference. Following (52), we use the L1 for fine-tuning but found that it did not confer any benefit.

[12]Note that the training data-set metrics in these Figures are noisy but that the test-set values are stable and consistently ordered.

When evaluating our systems' data assimilation performance we follow (9) in using the following quantity which we refer to as the 'DA MSE':

$$MSE(\boldsymbol{x}^{DA}) = \frac{\left\|\boldsymbol{x}^{DA} - \boldsymbol{x}^{obs}\right\|_2}{\left\|\boldsymbol{x}^{obs}\right\|_2} \tag{4.31}$$

The equivalent quantity for the background state $MSE(\boldsymbol{x}^b)$ is referred to as the 'ref MSE' and is the value of the $MSE(\cdot)$ before DA has taken place. If a DA MSE is lower than the ref MSE, this implies the approach is performing better than the DA baseline system. This baseline system predicts that the state is equal to the historical mean $\boldsymbol{x}^b$ regardless of the observations. The average ref MSE over the 197 training samples is 1.0001[13]. Unless otherwise stated, in all cases in which a single DA MSE is provided, this is the average over all training examples.

4.4.5 Hyperparameters

We trained our models with the Adam optimiser with the Pytorch default parameters of β_1= 0.9 and β_2 = 0.999. We implemented a system in which the learning rate was set by cross validation according to the lowest MSE after 25 epochs but found that this systematically overestimated the appropriate learning rate and resulted in many unstable runs. We eventually concluded that a relatively low fixed learning rate of 0002[14] performed relatively well for all systems although even in this case there were 'spikes' of instability in the training cycle as shown in Figure 5.2.
We used He et al. initialization (84) and batch size of 16 for most models as this was largest multiple of eight at which the fp32 model, gradients and data could fit in 11GB of available GPU memory. One exception to this was the GDRN model which would only run at batch size 8.

[13]The fact that this value is close to 1 is coincidental as we undo the normalisation before calculating this value.

[14]We note that Adam uses an adaptive step-size strategy so the learning rate is not precisely 'fixed'.

Chapter 5

Experiments

In this chapter we detail validate our proposed approach experimentally. In the first section 5.2 we report our experiments on the training of a range of CAEs and in the following section 5.2 we empirically demonstrate the success of our proposed method over reduced space VarDA with TSVD.

5.1 Architecture Search

Model	Best DA MSE	Relative Improvement over Backbone	Best RB Type	Best Activation
Backbone	0.2309	0.00%	-	PReLU
ResNeXt-L-N	0.1900	17.71%	Vanilla + CBAM	GDN
RDB3-L-N	0.1865	19.21%	Vanilla + CBAM	GDN
RAB-L	0.1917	16.98%	NeXt	PReLU
GRDN	0.1689	26.85%	NeXt + CBAM	GDN
Tucodec	**0.0858**	**62.86%**	vanilla	PReLU
Ref MSE	1.0001	-	-	-

Table 5.1: A summary of the data assimilation performance of the best version of each of the five model variants **after 150 epochs of training** and the ref MSE for comparison. The Tucodec model performed significantly better than the other systems over this training duration.

In this section we report the results of our investigations into the best architectures for data assimilation on the data described in section 4.4.1. We did not have compute available for the whole grid-search of options so performed these experiments sequentially: only accepting a change when it improved on the previous system. We remind the reader that in this section 5.1, all models are trained for just 150 epochs. We consider five variants of model in this section: the four backbone variants and the Tucodec model. We investigate the best activation and residual block (where appropriate) for all models in sections 5.1.2 and 5.1.1 respectively. A high-level summary of these results is given in Table 5.1. We conducted experiments to find the best L and N values for the ResNeXt-L-N and RAB-L which we briefly describe first. Unless otherwise stated, we use PReLU activations applied channel-wise.

ResNeXt width and cardinality

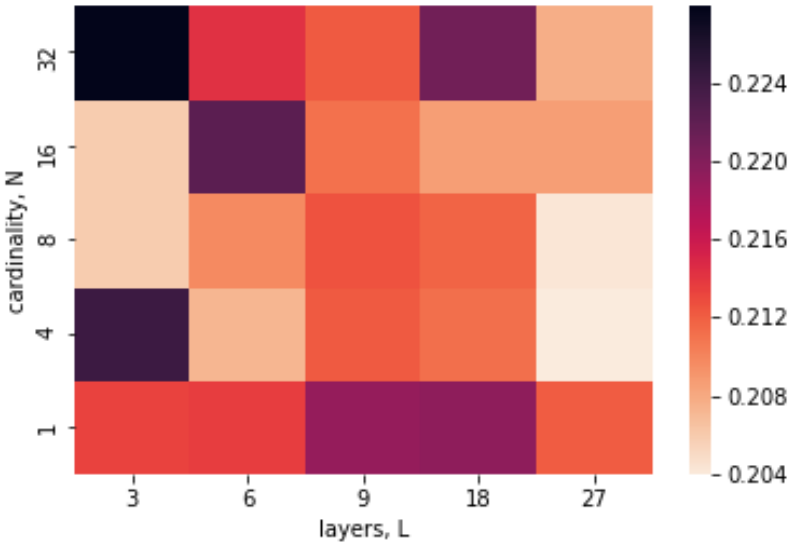

Figure 5.1: The DA MSE heatmap for a ResNeXt3-L-N-NeXt models with a range of cardinalities and layers. All use PReLU activations.

In the first experiment, we investigated the effect of changing cardinality and number of layers within residual component of our ResNeXt variant. We used NeXt RBs for these experiments. The results are in Figure 5.1. The best system was ResNeXt3-27-4-NeXt but there was also an interesting group of models with three layers on the left of Figure 5.1. We decided to preserve this diversity and take three models forward to the next stage of experiments. These were ResNeXt3-27-4-NeXt, ResNeXt3-3-8-NeXt and ResNeXt3-27-1-NeXt[1].

RAB-L

Number of RABs	DA MSE	Relative Improvement over Backbone
1	0.2005	13.17%
2	0.2188	5.24%
4	0.1917	16.98%
8	0.2071	10.31%

Table 5.2: The DA performance of the RAB-L architectures.

[1]This final model did not perform particularly well but is a ResNet-34 with the final linear layer removed and, as we knew we would be evaluating the systems with vanilla RBs, we thought that in view of its historical successes, there was a good argument for keeping this architecture.

In our second experiment we investigated the effect changing the number of consecutive RAB blocks in the backbone network. The results are in Table 5.2. The the best of these systems with four RABs, gives a 17% improvement relative to the Backbone but is poor in comparison with the best Tucodec model.

5.1.1 Residual Block

Model	Vanilla	Vanilla + CBAM	NeXt	NeXt + CBAM	Relative Improvement over Backbone
ResNeXt3-27-4	0.2108	0.1998	0.2028	**0.1907**	17.41%
RDB3-27-4	0.1950	**0.1893**	0.1964	0.2005	18.02%
ResNeXt3-27-1	0.2031	**0.1948**	0.2106	0.2110	15.63%
RDB3-27-1	0.2064	0.2167	**0.1968**	0.2060	14.75%
ResNeXt3-3-8	0.2196	**0.2051**	0.2174	0.2148	11.16%
RDB3-3-8	**0.1958**	0.2125	0.2004	0.2013	15.20%
RAB-4	0.2277	0.1970	**0.1917**	0.1927	16.98%
GRDN	0.2297	0.2204	0.2300	**0.1893**	18.00%
Tucodec	**0.0858**	0.3172	0.0890	0.1870	**62.86%**

Table 5.3: DA MSE variation with residual block in models trained for 150 epochs.

In this experiment we investigated the effect of RB type on our pool of architectures. We found that no single RB was superior for all systems but, for a given architecture, there were large variations in model performance with RB. For example, vanilla + CBAM RBs were better than NeXt blocks in five of six cases within the ResNeXt/RBD framework. This demonstrates the power of the former given that NeXt blocks were used when we tuned the vales of L and M.

In comparison, for the Tucodec model, networks with CBAMs performed much worse than those with without. This may be a result of interference between the coarse-grained attention mechanism of the CBAM blocks and the highly specific attention in the RAB blocks.

5.1.2 Activation function

In this experiment we investigated the effect of activation function on the best performing systems from the previous section 5.1.1. As the Tucodec models were performing well, we investigated the effect of the different activations on all four types of block. Note that, the Tucodec model has three GDN activations in its core encoder design (as shown in Figure 3.17) which were present throughout all experiments and we investigated altering its other activations within the RBs and RABs. The results are shown in Table 5.4.

PReLU activations were superior in a majority of cases although in the ReLU case, our Backbone-based systems either diverged or did not train when ReLUs were used. This was true even when we experimented with a large range of learning rates. The

Model	PReLU	ReLU	GDN	Relative Improvement over Backbone
Backbone	**0.2309**	0.9857	0.2970	0.00%
ResNeXt3-27-1-vanilla+CBAM	0.1948	1.0058	**0.1900**	17.71%
RDB3-3-8-vanilla	**0.1958**	0.9887	0.2027	15.20%
RDB3-27-4-vanilla+CBAM	0.1893	-	**0.1865**	19.21%
RAB-4-NeXt	**0.1917**	0.9992	0.2146	16.98%
GRDN-NeXt+CBAM	0.1893	1.0001	**0.1689**	26.85%
Tucodec-NeXt	**0.0890**	0.1624	0.1586	61.47%
Tucodec-NeXt+CBAM	**0.1870**	0.2662	0.2539	19.02%
Tucodec-vanilla	**0.0858**	0.0939	0.1212	**62.86%**
Tucodec-vanilla+CBAM	0.3172	**0.1788**	0.2805	22.56%

Table 5.4: DA MSE variation with activation function in models trained for 150 epochs.

backbone architecture was tuned using LReLUs and PReLUs and we came to the conclusion that the network had insufficient capacity to down-sample the inputs when ReLUs were employed. This is unfortunate as it prevents us from drawing any wider conclusions on the applicability of PReLUs over ReLUs in this case.

Our findings with regards to GDNs were more concrete. The denominator in the GDN equation 3.12 was zero, or close to zero, relatively frequently resulting in division by zero errors. As an indicator of this, in two of the three cases in 5.4 for which GDNs performed 'best' the models actually produced `inf` predictions for one of the 197 test set samples. These were dropped from the final statistic calculations. This process occurred more often earlier in training than later, and with the test data than with the training data. As GDNs work well in the Tucodec backbone, we reasoned that they are unstable when the input distribution is unpredictable. Therefore, while we were not able to utilise them particularly successfully here, we hope future work will determine the range of scenarios in which GDNs are reliable.

5.1.3 L1 fine tuning

We experimented with the use of L1 fine-tuning late in the training process as recommended in (52) but found that it did not give an appreciable benefit. In fact, as shown in Figure 5.2, in our experiments it increased the degree of over fitting without providing any generalisation advantage.

5.1.4 Augmentation

To quantify what, if any, effect our augmentation technique was having, we retrained the Tucodec-NeXt model with a range of augmentation strengths as shown in Figure 5.3. We did not observe a large difference between the methods so choose the strongest augmentation that did not harm performance (augmentation strength 2 in Table 5.5) when training our models to convergence.

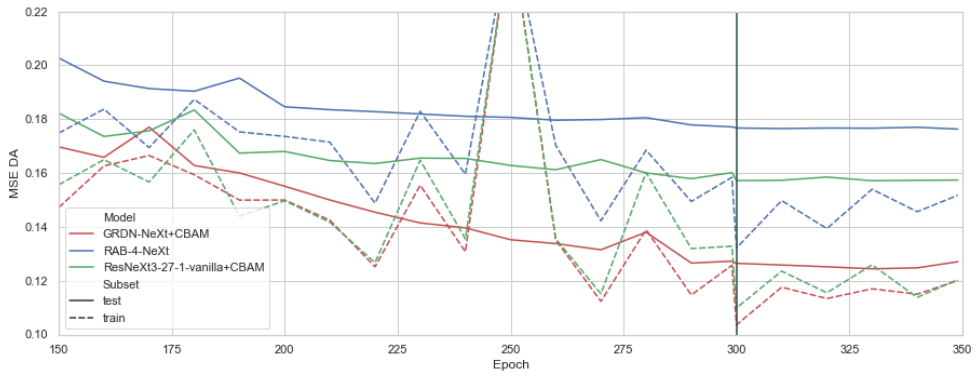

Figure 5.2: The MSE DA with L1 fine-tuning from epoch 300 onwards. The training-set DA MSE decreased but the this was not accompanied with a test-set decrease. We investigated L1 fine tuning for all models in Table 5.6 but just present a representative selection here.

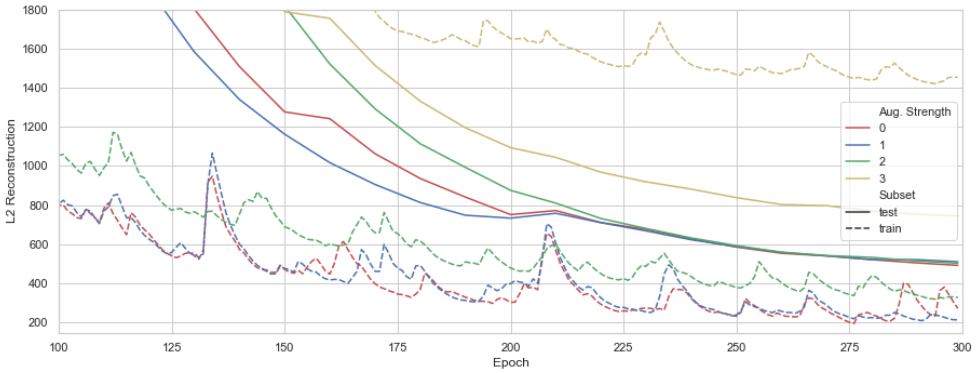

Figure 5.3: The training and validation MSE reconstruction errors during training with different strengths of augmentation detailed in Table 5.5. These graphs have been smoothed with an exponential moving average with $\alpha = 0.4$ as the spikes in the training curves made this diagram too noisy to be illustrative. A non-smoothed version is given in Appendix B.

Augmentation Strength	Jitter Amplitude	Jitter Frequency	Jitter Amplitude per Location
0	None	None	None
1	0.005	0.5	0.0025
2	0.05	0.25	0.0125
3	0.1	0.5	0.0500

Table 5.5: The field-jitter augmentation strengths we investigated. We added Gaussian noise with standard deviation of 'Jitter Amplitude' at 'Jitter Frequency' of the total locations in the state.

5.1.5 Architecture Summary

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

Table 5.6: A comparison of the DA MSE and inference speeds of a selection of our best models. Note that unlike in previous sections, these systems were trained to convergence.

We trained a selection of our best models to convergence and found that the Tucodec-NeXt and Tucodec-vanilla models performed best as shown in Table 5.6. The two models have very similar DA MSE values meaning that it is not appropriate to conclude that the NeXt model is 'better' at DA (particularly when we aren't using a validation data-set). However, the NeXt model is almost x2.5 faster during inference which gives a very compelling reason for using it instead of the vanilla variant when making comparisons with reduced space DA in the following section.

5.2 Comparison with TSVD

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 5.7: Comparison of our best Tucodec model with the Arcucci et al. approach (9) which sets $\sigma_\tau = \sqrt{\sigma_1} = 32$. Our DA MSE is 37% lower than the best performing Arcucci et al. system.

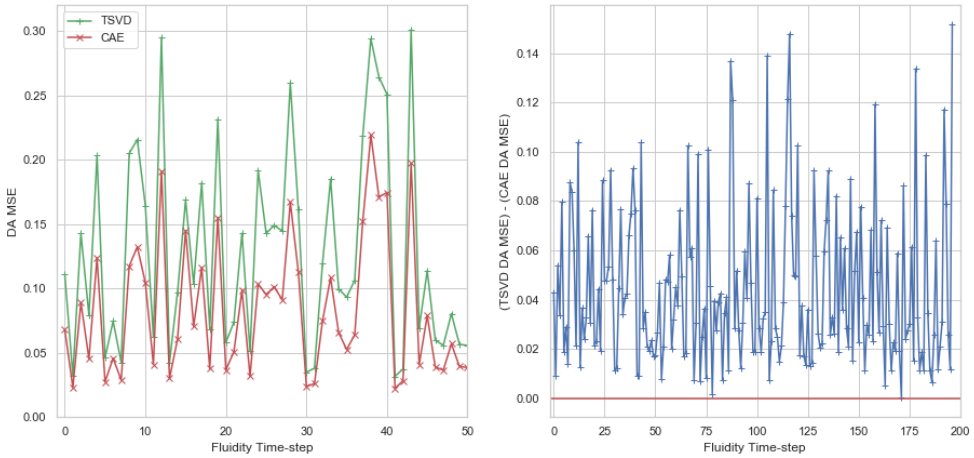

Figure 5.4: Comparison of TSVD ($\tau = 32$, $M = n$) and AE data assimilation performance across sequential Fluidity time-steps. Note in Figure a) that the two methods find the same states difficult. In Figure b) we give the difference between the DA MSEs of the two methods for the whole test set. The proposed method performs better (is above the red line) in the vast majority of cases.

In this section, we compare our system against Reduced space VarDA with TSVD as described in (9). Our system has superior DA performance when the results are averaged over the test set as shown in Table 5.7. This is not just true on average: our system is *consistently* better in both time (see Figure 5.4) and space (see Appendix C).

Moreover, we show in Figure 5.5 (and Appendix C) that our method has a DA MSE that is 15% lower than the reduced space approach with $M = n$ even in the limit in which there is no truncation of $\boldsymbol{V}$ ($\tau = S$) and the method becomes Parish et al.'s approach. This is surprising: even if our CAE was truly lossless, which it is not,

the matrices $\boldsymbol{V}$ and $\boldsymbol{V}_l$ contain the same information (albeit the latter stores it more efficiently)[2]. The better performance of our bi-reduced space approach might be explained by the poor conditioning in the mono-reduced space and the resulting in numerical errors. This requires further research.

5.2.1 Performance-speed tradeoff

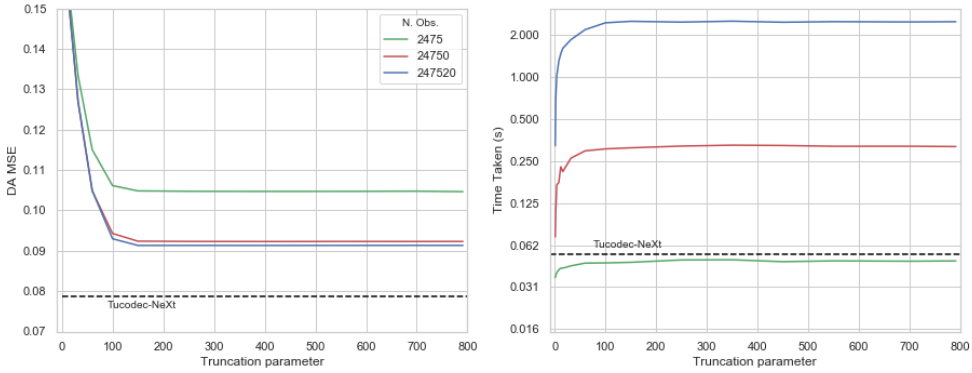

Figure 5.5: Effect of truncation parameter τ on a) DA MSE and b) online time. Tucodec DA MSE and CPU execution time are marked with dashed black lines. Note the logarithmic scale on the y-axis in b).

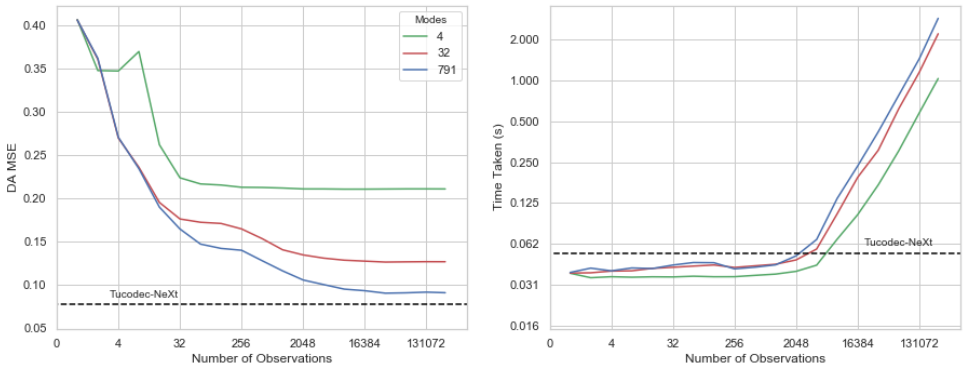

Figure 5.6: Effect of number of observations M on a) DA MSE and b) execution time. The fact that the MSE is not monotonically increasing with modes=4 is due to the fact that we randomly choose a different subset of observations for each experiment.

The reduced space approach in (9) has an acute performance-speed tradeoff occurring along three axes:

i The size of the truncation parameter τ. As this increases, the DA performance increases but the speed decreases as shown in Figure 5.5.

[2]As an aside, note that in this case our method is x43 faster than the traditional approach.

ii The number of observations M. As this increases, the DA performance increases but the speed decreases as shown in Figure 5.6. This is a stronger effect than with τ. Noting the logarithmic scales in this Figure 5.6b, it is clear that there is only a small range of M for which our method is slower than TSVD.

iii The size of the observation variance σ_0. We do not consider this here but Arcucci et al. show that as this parameter increases, performance increases but speed decreases (9)[3]. In all experiments here we used $\sigma_0 = \sigma_l = 0.005$.

Our system's evaluation speed is not sensitive to the number of observations[4], nor the value of σ_l. There is not a direct equivalent of τ in our system although we expect that altering the latent dimension size m will likely result in a similar tradeoff.

We show the performance-speed tradeoff for a range of models in Figure 5.7. All timing measurements were carried out on the Intel Xeon E5-2690 v3 (Haswell) 2.60 GHz CPU and averaged over the test set. The clock was started at the beginning of the minimisation routine when all relevant data was already in memory.

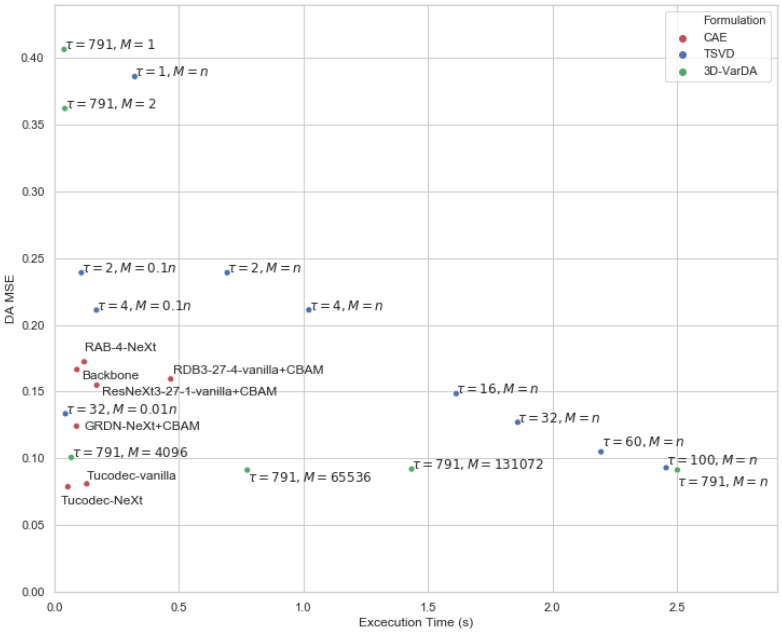

Figure 5.7: Performance-speed tradeoff for a range of systems.

[3]The authors make this comparison indirectly via the parameter α: a constant which they introduce to the cost function that is inversely proportional to σ_0^2.

[4]The performance of our system *will* be affected by decreasing the number of observations.

Chapter 6

Discussion

In this section we briefly cover a range of discussion topics raised by this work.

Optimal τ and M

Considering the Figure 5.7 in the previous chapter 5.7 it appears that, for the combinations of τ and M that we investigate here, the pairing that is most successful in the performance-speed tradeoff if $\tau = 791$ (i.e. no truncation) and $M = 4096$. It is worth making a few observations on this result:

i There was no way to know that this combination was the best in advance as they will vary from one data-set to another. It took something on the order of 60 CPU hours to calculate the DA MSE on the test-set for the range of τ and M displayed in Figure 5.7. We note that this value could be reduced by a more intelligent search method, but draw the reader's attention to the fact that this is of the same order as the 15 hours required to train a Tucodec-NeXt model to convergence on a GPU.

ii This result is still 30% slower and 20% less accurate for DA than the Tucodec-Next model.

Hardware Accelerators

All timing comparisons were made on the CPU as we did not have a GPU implementation of Arcucci et al.'s routine. Using a K80 GPU with our method provided a speed-up of approximately 40% for our method. This was with a *very* poor implementation in which the data was transferred from the CPU to the GPU and back again. We would expect an optimised implementation of our system on a modern accelerator to achieve a much larger relative speed-up over the Figures here than the equivalent optimised version of Arcucci et al.'s method. To give a comprehensive argument for why we believe this to be the case would require an extra chapter here. Instead we briefly sketch our our argument in a single paragraph.

Recall that the bottleneck in our system is a fully convolutional decoder while, in the Arcucci et al. routine, the cost is dominated by large matrix multiplications and

vector dot-products. The convolutional kernel parameters are used many times in a forward pass but there is almost no data reuse in the Arcucci et al. case. As such, the latter will be bandwidth-limited on every hardware platform we can imagine but the former will enter the compute-bound domain in some cases[1]. Historically, it has proven easier to accelerate compute-bound processes than memory bound ones and there is reason to believe that this will continue, not least because there is a whole industry built-around the design of systems that specifically accelerate convolutional inference workloads. We will not attempt to review the hardware options here but if the Graphcore 'IPU', which is set to ship early in 2020, delivers on its marketing promises (107), it would speed up our inference by up to three order of magnitude. There are also lower-cost, lower-power options such as FPGAs (108).

As such, we believe the quoted timing Figures underestimate the advantage of our approach.

Other Acceleration Options

During our architecture search, we optimised for DA performance rather than inference latency. Had we been focusing for the latter, there are a number of techniques separate to hardware acceleration that could be used to achieve this. Firstly, a thinner decoder could be used as suggested in (5) since only the decoder is evaluated in the online setting. Secondly, the existing network could be quantized (109) or pruned (110) (111) or both (112) to provide a substantial speed-increase. Additionally, convolutional acceleration approaches such as Pixel Shuffle (113) or factorised convolutions (114) could be employed to reduce the number of FLOPs in the forward decoder pass. Finally, there are innumerable small architectural changes that could be made: the replacement of vanilla blocks in the original Tucodec model with NeXt RBs being one such example.

None of the above strategies are available to traditional VarDA approaches. We note that some of these techniques will reduce the performance of our system but, as our approach has a considerable performance cushion over traditional approaches in its current form, this may be acceptable in some settings.

Importance of Architecture

The results in this paper demonstrate the central importance of using good CAE architectures. This field is moving exceptionally fast: our Backbone network, was state-of-the-art for image compression in 2017 (5) but gives a DA MSE that is a) double that of the Tucodec models and b) considerably poorer for DA than the Arcucci et. al. approach with $\tau = 32$ and $M = 0.01$[2].

[1] Convolution may be memory-bound depending on channel size and dimensions of the feature map.

[2] Unlike for $\tau = 791$, $M = 4096$, these parameters would be some of the first investigated under the Arcucci et al. approach and are therefore not subject to the ~ 60 hour search penalty.

On a related note, we found that it was non-trivial to extend many architectures to three spatial input dimensions and it required a large amount of manual tuning of the channel sizes so as not to create unreasonably large 4D feature maps (three spatial dimensions and one channel). In particular, our implementation of the GDRN (13) had extreme computational requirements in 3D which meant that it took almost three times longer to train than any other network. This was despite its modest number of parameters (see 5.6).

Chapter 7

Conclusions and Future Work

We present a new Bi-reduced space 3D-VarDA formulation and show that, in combination with the Zhou et al. or 'Tucodec' image compression CAE, this method gives superior data assimilation performance in comparison with Reduced space VarDA regardless of the parameters used in the latter case. We have demonstrated that our method is also faster in the majority of scenarios. On the theoretical side, we show that our method produces approximately equivalent solutions to the traditional method at lower computational complexity. Unlike the previous approach which is in $\mathcal{O}(M^2)$ for large M, our method does not penalize the collection of more observation data. We have released our work in a well tested Python module `VarDACAE`.

There were many extensions to this work which we would have liked to explore further. We feel that the most important of these is the validation of our hypothesis that is possible to create an observation encoder network f^o to calculate the latent misfits $\boldsymbol{d}_l$. We would also have liked to apply our approach to 4D-VarDA, validate it on other data sets and investigate alternatives to the L-BFGS minimization routine. A more substantial extension would involve integrating our method with CAE-based ROM approaches to produce a single end-to-end network for reduced space data assimilation and we believe this would be complemented by the use of data assimilation localization techniques (31).

Other directions for future work include a systematic evaluation of activation functions for compression including an investigation of the conditions under which GDNs are suitable. On the modelling side, there is potential for the use of VAEs to enforce orthogonality in the latent dimension or the use of GAN-CAEs. There are also many architectural variants and implementation details, some of which were discussed in the previous section, that might provide a considerable speed or accuracy increase relative to the system we present in this work.

Bibliography

[1] Met. Office, "The Cray XC40 supercomputer," 2019. pages 1

[2] Met. Office, "Data Assimilation Methods," 2019. pages 1, 4, 43

[3] P. Vincent, H. Larochelle, Y. Bengio, and P. A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine Learning*, no. April, pp. 1096–1103, 2008. pages 1, 11

[4] K. G. Lore, A. Akintayo, and S. Sarkar, "LLNet: A deep autoencoder approach to natural low-light image enhancement," *Pattern Recognition*, vol. 61, pp. 650–662, 2017. pages 1, 11

[5] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy Image Compression with Compressive Autoencoders," pp. 1–19, 2017. pages 1, 15, 16, 19, 23, 30, 47, 48, 66

[6] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," 2018. pages 1, 16, 20, 30

[7] P. Courtier, J.-N. Thépaut, and A. Hollingsworth, "A strategy for operational implementation of 4D-Var, using an incremental approach," *Quarterly Journal of the Royal Meteorological Society*, vol. 120, no. 519, pp. 1367–1387, 1994. pages 1, 3, 6, 7

[8] D. Parrish and J. Derber, "The National Meteorological Center's Spectral Statistical-Interpolation Analysis System.pdf," 1992. pages 1, 2, 5, 7, 32, 40

[9] R. Arcucci, L. Mottet, C. Pain, and Y. K. Guo, "Optimal reduced space for Variational Data Assimilation," *Journal of Computational Physics*, vol. 379, pp. 51–69, 2019. pages 1, 2, 3, 5, 8, 9, 39, 55, 62, 63, 64

[10] J. Song, 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 and Information*, vol. 46, no. 8, pp. 809–828, 2018. pages 1

[11] Z. Wang, D. Xiao, F. Fang, and C. Pain, "Model identification of reduced order fluid dynamics systems using deep learning Model identification of reduced order fluid dynamics systems using deep learning," no. October, 2017. pages 1, 13

[12] L. Zhou, Z. Sun, X. Wu, and J. Wu, "End-to-end Optimized Image Compression with Attention Mechanism," 2019. pages 1, 2, 15, 16, 21, 23, 25, 28, 29, 30, 31

[13] D.-W. Kim, J. R. Chung, and S.-W. Jung, "GRDN:Grouped Residual Dense Network for Real Image Denoising and GAN-based Real-world Noise Modeling," 2019. pages 2, 27, 28, 67

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 770–778, 2016. pages 2, 16, 22, 27, 50

[15] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 5987–5995, 2017. pages 2, 23, 24, 30, 49

[16] A. C. Lorenc, "Analysis methods of numerical weather prediction," *Quarterly Journal of the Royal Meteorological Society*, vol. 112, no. 474, pp. 1177–1194, 1986. pages 3

[17] A. C. Lorenc, "Optimal Nonlinear objective analysis," *Quarterly Journal of the Royal Meteorological Society*, vol. 114, no. 479, pp. 205–240, 1988. pages 3

[18] P. Courtier, E. Anderson, W. Heckley, D. Vasiljevec, M. Hamrud, A. Hollingsworth, F. Rabier, M. Fisher, and J. Pailleux, "The ECMWF implementation of three-dimensional variational assimilation (3D-Var). I: Formulation," *Quarterly Journal of the Royal Meteorological Society*, vol. 124, no. 550, pp. 1783–1807, 1998. pages 3

[19] W. Huang, A. J. Bourgeois, Q. N. Xiao, D. M. Barker, and Y.-R. Guo, "A Three-Dimensional Variational Data Assimilation System for MM5: Implementation and Initial Results," *Monthly Weather Review*, vol. 132, no. 4, pp. 897–914, 2004. pages 3

[20] G. Evensen, "The Ensemble Kalman Filter: Theoretical formulation and practical implementation," *Ocean Dynamics*, vol. 53, no. 4, pp. 343–367, 2003. pages 3

[21] S. Dobricic and N. Pinardi, "An oceanographic three-dimensional variational data assimilation scheme," *Ocean Modelling*, vol. 22, no. 3-4, pp. 89–105, 2008. pages 3

[22] M. Lang and M. J. Owens, "A Variational Approach to Data Assimilation in the Solar Wind," *Space Weather*, vol. 17, no. 1, pp. 59–83, 2019. pages 3

[23] J. Tribbia and D. P. Baumhefner, "Scale Interactions and Atmospheric Predictability: An Updated Perspective," *Monthly Weather Review*, vol. 132, no. 3, pp. 703–713, 2004. pages 3

[24] R. N. Bannister, "Elementary 4d-VAR: DARC Technical Report No. 2," *DARC Technical Report No. 2*, no. 2, pp. 1–16, 2001. pages 3

[25] R. N. Bannister, "A review of operational methods of variational and ensemble-variational data assimilation," *Quarterly Journal of the Royal Meteorological Society*, vol. 143, no. 703, pp. 607–633, 2017. pages 3, 4, 7

[26] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960. pages 3, 80

[27] P. J. van Leeuwen, "Particle Filtering in Geophysical Systems," *Monthly Weather Review*, vol. 137, no. 12, pp. 4089–4114, 2009. pages 4

[28] M. Morzfeld, D. Hodyss, and C. Snyder, "What the collapse of the ensemble Kalman filter tells us about particle filters," *Tellus, Series A: Dynamic Meteorology and Oceanography*, vol. 69, no. 1, pp. 1–15, 2017. pages 4

[29] M. M. Graham and A. H. Thiery, "A scalable optimal-transport based local particle filter," 2019. pages 4

[30] A. C. Lorenc and M. Jardak, "A comparison of hybrid variational data assimilation methods for global NWP," *Quarterly Journal of the Royal Meteorological Society*, vol. 144, no. 717, pp. 2748–2760, 2018. pages 4, 43

[31] T. Montmerle, Y. Michel, É. Arbogast, B. Ménétrier, and P. Brousseau, "A 3D ensemble variational data assimilation scheme for the limited-area AROME model: Formulation and preliminary results," *Quarterly Journal of the Royal Meteorological Society*, vol. 144, no. 716, pp. 2196–2215, 2018. pages 5, 8, 39, 68

[32] R. N. Bannister, "A review of forecast error covariance statistics inatmospheric variational data assimilation. II: Modelling the forecast error covariance statistics," vol. 1996, no. November, p. 496, 2008. pages 5, 7

[33] D. Zupanski, "A General Weak Constraint Applicable to Operational 4DVAR Data Assimilation Systems," *Monthly Weather Review*, vol. 125, no. 9, pp. 2274–2292, 2002. pages 6

[34] L. D. Navon, I. M., "Conjugate Gradient Methods for Large-Scale Minimization in Meteorology," 1987. pages 8

[35] A. K. Alekseev, I. M. Navon, and J. L. Steward, "Comparison of advanced large-scale minimization algorithms for the solution of inverse ill-posed problems," *Optimization Methods and Software*, vol. 24, no. 1, pp. 63–87, 2009. pages 8

[36] J. Nocedal and D. C. Liu, "On the limited memory BFGS method for large scale optimization," *Mathematical Programming*, vol. 45, pp. 503–528, 1989. pages 8

[37] P. C. Hansen, J. G. Nagy, and D. P. O'Leary, "Deblurrinq Images: Matrices, Spectra and Filtering," *Matrices, Spectra, and Filtering*, pp. 1–145, 2006. pages 8, 43

[38] T. Chai, G. R. Carmichael, Y. Tang, A. Sandu, M. Hardesty, P. Pilewskie, S. Whitlow, E. V. Browell, M. A. Avery, P. Nédélec, J. T. Merrill, A. M. Thompson, and E. Williams, "Four-dimensional data assimilation experiments with International Consortium for Atmospheric Research on Transport and Transformation ozone measurements," *Journal of Geophysical Research Atmospheres*, vol. 112, no. 12, pp. 1–18, 2007. pages 9

[39] H. Cheng, M. Jardak, M. Alexe, and A. Sandu, "A hybrid approach to estimating error covariances in variational data assimilation," *Tellus, Series A: Dynamic Meteorology and Oceanography*, vol. 62, no. 3, pp. 1–18, 2010. pages 9

[40] E. N. Lorenz, "Empirical Orthogonal Functions and Statistical Weather Prediction," 1956. pages 9

[41] A. Dertat, "Applied Deep Learning - Part 3: Autoencoders," 2017. pages 10

[42] D. E. Rumelhart, G. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," *Cognitive Science*, no. V, 1986. pages 10, 17

[43] P. Baldi, "Autoencoders, Unsupervised Learning, and Deep Architectures," pp. 37–50, 2012. pages 10, 12

[44] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," no. Ml, pp. 1–14, 2013. pages 10

[45] M. J. Kusner, B. Paige, and J. M. Hemández-Lobato, "Grammar variational autoencoder," *34th International Conference on Machine Learning, ICML 2017*, vol. 4, pp. 3072–3084, 2017. pages 10

[46] Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. Carin, "Variational autoencoder for deep learning of images, labels and captions," *Advances in Neural Information Processing Systems*, no. Nips, pp. 2360–2368, 2016. pages 10

[47] I. Goodfellow, "Deep Learning," *The Brain & Neural Networks*, vol. 24, no. 1, pp. 1–2, 2017. pages 11

[48] C. Doersch, "Tutorial on Variational Autoencoders," pp. 1–23, 2016. pages 11, 34

[49] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," *ACM International Conference Proceeding Series*, vol. 02-Decembe, pp. 4–11, 2014. pages 11

[50] C. Baur, B. Wiestler, S. Albarqouni, and N. Navab, "Deep autoencoding models for unsupervised anomaly segmentation in brain MR images," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11383 LNCS, pp. 161–169, 2019. pages 11

[51] S. Hayou, "Cleaning the correlation matrix with a denoising autoencoder," pp. 1–11, 2017. pages 11

[52] M. Lu, T. Chen, H. Liu, and Z. Ma, "Learned Image Restoration for VVC Intra Coding," pp. 2–5, 2019. pages 12, 16, 23, 25, 26, 30, 54, 59

[53] J. Deng, W. Dong, R. Socher, L.-j. Li, K. Li, and L. Fei-fei, "ImageNet : a Large-Scale Hierarchical Image Database ImageNet : A Large-Scale Hierarchical Image Database," no. May 2014, 2009. pages 13, 17

[54] L. Cordier, B. R. Noack, G. Tissot, G. Lehnasch, J. Delville, M. Balajewicz, G. Daviller, and R. K. Niven, "Identification strategies for model-based control Topics in Flow Control. Guest editors J.P. Bonnet and L. Cattafesta," *Experiments in Fluids*, vol. 54, no. 8, 2013. pages 13

[55] R. van der Merwe, T. K. Leen, Z. Lu, S. Frolov, and A. M. Baptista, "Fast neural network surrogates for very high dimensional physics-based models in computational oceanography," *Neural Networks*, vol. 20, no. 4, pp. 462–478, 2007. pages 13

[56] M. Wang, H. X. Li, X. Chen, and Y. Chen, "Deep Learning-Based Model Reduction for Distributed Parameter Systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 12, pp. 1664–1674, 2016. pages 13

[57] K. Loh, P. S. Omrani, and R. van der Linden, "Deep Learning and Data Assimilation for Real-Time Production Prediction in Natural Gas Wells," 2018. pages 14

[58] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in Neural Information Processing Systems*, vol. 4, no. January, pp. 3104–3112, 2014. pages 14

[59] J. Zhu, S. Hu, R. Arcucci, C. Xu, J. Zhu, and Y.-k. Guo, "Model error correction in data assimilation by integrating neural networks," *Big Data Mining and Analytics*, vol. 2, no. 2, pp. 83–91, 2019. pages 14

AEs in DA

AEs in DA

[60] M. Liu and D. Grana, "Ensemble-based seismic history matching with data reparameterization using convolutional autoencoder," *2018 SEG International Exposition and Annual Meeting, SEG 2018*, no. August, pp. 3156–3160, 2019. pages 14

[61] C. A. Quilodran Casas, N. Sparks, and R. Toumi, "Fast ocean data assimilation using a neural-network reduced-space regional ocean model of the North Brazil Current," *Progress in Oceanography*, 2019. pages 14

[62] L. Zhou, C. Cai, Y. Gao, S. Su, and J. Wu, "Variational Autoencoder for Low Bit-rate Image Compression," *IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 2617–2620, 2018. pages 15, 16, 21, 23, 25, 30

[63] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. Van Gool, "Practical Full Resolution Learned Lossless Image Compression," pp. 1–14, 2018. pages 15

[64] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9351, pp. 234–241, 2015. pages 16, 24, 25, 54

[65] S. Cho, J. Lee, J. Kim, Y. Kim, D.-w. Kim, J. R. Chung, and S.-w. Jung, "Low Bit-rate Image Compression based on Post-processing with Grouped Residual Dense Network," pp. 1–5, 2019. pages 16, 23, 27, 29, 30, 31, 48

[66] Y. Fan, J. Yu, and T. S. Huang, "Wide-activated Deep Residual Networks based Restoration for BPG-compressed Images," *Cvpr2018*, pp. 2621–2624, 2018. pages 16, 23, 25, 30

[67] J. Zhou, S. Wen, A. Nakagawa, K. Kazui, and Z. Tan, "Multi-scale and Context-adaptive Entropy Model for Image Compression," tech. rep., 2019. pages 16, 23, 25, 30

[68] M. Li, C. Xia, J. Hu, Z. Huang, Y. Zhang, D. Chen, J. Zan, G. Li, and J. Nie, "VimicroABCnet: An Image Coder Combining A Better Color Space Conversion Algorithm and A Post Enhancing Network," pp. 2–6, 2019. pages 16, 23, 25, 30

[69] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," pp. 1–15, 2014. pages 16, 26

[70] G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar, "Variable Rate Image Compression with Recurrent Neural Networks," pp. 1–12, 2016. pages 16, 17, 23

[71] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimization of nonlinear transform codes for perceptual quality," *2016 Picture Coding Symposium, PCS 2016*, no. 1, 2017. pages 16

[72] M. Li, W. Zuo, S. Gu, D. Zhao, and D. Zhang, "Learning Convolutional Networks for Content-Weighted Image Compression," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3214–3223, 2018. pages 16, 30

[73] G. Toderici, D. Vincent, N. Johnston, S. J. Hwang, D. Minnen, J. Shor, and M. Covell, "Full resolution image compression with recurrent neural networks," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 5435–5443, 2017. pages 17

[74] C. Huang, H. Liu, T. Chen, S. Pu, Q. Shen, and Z. Ma, "Extreme Image Compression via Multiscale Autoencoders With Generative Adversarial Optimization," pp. 0–4, 2019. pages 17

[75] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," pp. 1–31, 2018. pages 17, 18

[76] K. Fukushima, "Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position," *Biol. Cybernetics*, vol. 202, pp. 193–202, 1980. pages 17

[77] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. pages 17

[78] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks Alex," pp. 60–1–60–16, 2012. pages 17, 18, 20

[79] A. Lucas, M. Iliadis, R. Molina, and A. K. Katsaggelos, "Using Deep Neural Networks for Inverse Problems in Imaging: Beyond Analytical Methods," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 20–36, 2018. pages 18

[80] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for Simplicity: the All Convolutional Net," pp. 1–14, 2015. pages 18

[81] G. Hinton and V. Nair, "Rectified Linear Units Improve Restricted Boltzmann Machines," *Proceeding ICML'10 Proceedings of the 27th International Conference on International Conference on Machine Learning*, pp. 807–814, 2010. pages 20

[82] H. Touvron, A. Vedaldi, M. Douze, and H. Jégou, "Fixing the train-test resolution discrepancy," 2019. pages 20, 21

[83] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier Nonlinearities Improve Neural Network Acoustic Models," vol. 28, 2013. pages 20

[84] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers : Surpassing Human-Level Performance on ImageNet Classification," 2015. pages 20, 55

[85] J. Ballé, V. Laparra, and E. P. Simoncelli, "Density Modeling of Images using a Generalized Normalization Transformation," pp. 1–14, 2015. pages 20

[86] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation Functions: Comparison of trends in Practice and Research for Deep Learning," pp. 1–20, 2018. pages 21

[87] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Deep Convolutional AutoEncoder-based Lossy Image Compression," *2018 Picture Coding Symposium, PCS 2018 - Proceedings*, pp. 253–257, 2018. pages 21, 22, 30, 47

[88] K. Ma, W. Liu, K. Zhang, Z. Duanmu, Z. Wang, and W. Zuo, "End-To-end blind image quality assessment using deep neural networks," *IEEE Transactions on Image Processing*, vol. 27, no. 3, pp. 1202–1213, 2018. pages 21, 22

[89] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June, pp. 1–9, 2015. pages 22, 23, 24

[90] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. V. Gool, "Conditional Probability Models for Deep Image Compression," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 4394–4402, 2018. pages 23, 30, 48, 49, 50

[91] Z. Chen, Y. Li, F. Liu, Z. Liu, X. Pan, W. Sun, Y. Wang, Y. Zhou, H. Zhu, and S. Liu, "CNN-Optimized Image Compression with Uncertainty based Resource Allocation," pp. 2559–2562, 2018. pages 23, 27, 30, 34, 46, 49, 50

[92] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," pp. 1–13, 2016. pages 24

[93] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," pp. 1–14, 2015. pages 24

[94] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *32nd International Conference on Machine Learning, ICML 2015*, vol. 1, pp. 448–456, 2015. pages 24

[95] W. J.Z., L. I. J., G. R.M., and W. G., "Unsupervised Multiresolution Segmentation for Images with Low Depth of Field," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 1, pp. 85–90, 2001. pages 24

[96] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 2017-Decem, no. Nips, pp. 5999–6009, 2017. pages 26

[97] S. Woo, J. Park, J.-y. Lee, and I. S. Kweon, "CBAM: Convolutional Block Attention Module," pages 27, 28

[98] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-Octob, pp. 618–626, 2017. pages 27

[99] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 2261–2269, 2017. pages 27, 28

[100] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual Dense Network for Image Restoration," vol. 13, no. 9, pp. 1–14, 2018. pages 27, 31

[101] Y. Zhang, K. Li, K. Li, B. Zhong, and Y. Fu, "Residual Non-local Attention Networks for Image Restoration," pp. 1–18, 2019. pages 28, 29, 30

[102] "Versatile video coding reference software version 4.0 (VTM-4.0)." pages 30, 31

[103] P. Baldi and K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima," *Neural Networks*, vol. 2, no. 1, pp. 53–58, 1989. pages 34

[104] A. Cline and I. Dhillon, "Computation of the Singular Value Decomposition," pp. 1027–1039, 2013. pages 45

[105] D. Alexandre, C.-P. Chang, W.-H. Peng, and H.-M. Hang, "An Autoencoder-based Learned Image Compressor: Description of Challenge Proposal by NCTU," no. Clic, pp. 0–3, 2019. pages 50

[106] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, "Efficient object localization using Convolutional Networks," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June, pp. 648–656, 2015. pages 53

[107] D. Lacey, "Graphcore: PRELIMINARY IPU BENCHMARKS," 2018. pages 66

[108] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing FPGA-based accelerator design for deep convolutional neural networks," *FPGA 2015 - 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 161–170, 2015. pages 66

[109] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2704–2713, 2018. pages 66

[110] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning Filters for Efficient ConvNets," no. 2016, pp. 1–13, 2016. pages 66

[111] G. Bellec, D. Kappel, W. Maass, and R. Legenstein, "Deep Rewiring: Training very sparse deep networks," pp. 1–24, 2017. pages 66

[112] S. Han, H. Mao, and W. J. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," pp. 1–14, 2015. pages 66

[113] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "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*, vol. 2016-Decem, pp. 1874–1883, 2016. pages 66

[114] M. Wang, B. Liu, and H. Foroosh, "Factorized Convolutional Neural Networks," *Proceedings - 2017 IEEE International Conference on Computer Vision Workshops, ICCVW 2017*, vol. 2018-Janua, pp. 545–553, 2018. pages 66

[115] A. Lacey and N. Thacker, "Tutorial: The Kalman filter," *Imaging Science and Biomedical . . .* , pp. 133–140, 1998. pages 80

[116] K. B. Petersen and M. S. Pedersen, "The Matrix Cookbook," *Book*, vol. 16, no. 4, pp. 1–16, 2012. pages 81

Appendices

Appendix A

KF and VarDA equivalence

In this Appendix we prove that the Variational Data Assimilation and Kalman Filter approaches are equivalent under the assumption that:

i The observation operator is linear $\boldsymbol{\mathcal{H}}[\boldsymbol{x}] = \boldsymbol{H}\boldsymbol{x}$.

The KF approach states that:

$$\boldsymbol{x} = \boldsymbol{x}^b + \boldsymbol{K}(\boldsymbol{y} - \boldsymbol{H}\boldsymbol{x}^b) \tag{A.1}$$

$$\text{where} \quad \boldsymbol{K} = \boldsymbol{B}\boldsymbol{H}^T\left(\boldsymbol{H}\boldsymbol{B}\boldsymbol{H}^T + \boldsymbol{R}\right)^{-1} \tag{A.2}$$

Note that we are considering a single assimilated state $\boldsymbol{x}$ here to make the comparison with 3D-VarDA but the KF method is an iterative process in which the previous estimate becomes the new prior $\boldsymbol{x}^b$. The reader should consult the original 1960 paper (26) for the details of this formulation. Alternatively, Lacey et al. give a good tutorial on KFs that contains most, but not all, of the proof below (115).

Proof of equivalence

Recall that the 3D-VarDA cost from equation 2.11 is:

$$J(\boldsymbol{x}) = \frac{1}{2}\left\|\boldsymbol{x} - \boldsymbol{x}^b\right\|^2_{\boldsymbol{B}^{-1}} + \frac{1}{2}\left\|\boldsymbol{y} - \boldsymbol{\mathcal{H}}[\boldsymbol{x}]\right\|^2_{\boldsymbol{R}^{-1}} \tag{A.3}$$

Under the linear operator assumption this becomes:

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

In VarDA this is solved by a minimisation routine but the exact solution can be obtained by differentiating and setting to zero. The resulting expression can be rearranged to give the KF formulation:

$$\boldsymbol{B}^{-1}(\boldsymbol{x} - \boldsymbol{x}^b) - \boldsymbol{H}^T\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{H}\boldsymbol{x}) = 0$$
$$\boldsymbol{B}^{-1}(\boldsymbol{x} - \boldsymbol{x}^b) = \boldsymbol{H}^T\boldsymbol{R}^{-1}\boldsymbol{y} - \boldsymbol{H}^T\boldsymbol{R}^{-1}\boldsymbol{H}\boldsymbol{x}$$

$$
\begin{aligned}
\boldsymbol{B}^{-1}(\boldsymbol{x}-\boldsymbol{x}^b) &= \boldsymbol{H}^T\boldsymbol{R}^{-1}\boldsymbol{y} - \boldsymbol{H}^T\boldsymbol{R}^{-1}\boldsymbol{H}\boldsymbol{x} \\
&\quad + \boldsymbol{H}^T\boldsymbol{R}^{-1}\boldsymbol{H}\boldsymbol{x}^b - \boldsymbol{H}^T\boldsymbol{R}^{-1}\boldsymbol{H}\boldsymbol{x}^b \\
\boldsymbol{B}^{-1}(\boldsymbol{x}-\boldsymbol{x}^b) &= \boldsymbol{H}^T\boldsymbol{R}^{-1}\boldsymbol{y} - \boldsymbol{H}^T\boldsymbol{R}^{-1}\boldsymbol{H}(\boldsymbol{x}-\boldsymbol{x}^b) \\
&\quad - \boldsymbol{H}^T\boldsymbol{R}^{-1}\boldsymbol{H}\boldsymbol{x}^b \\
(\boldsymbol{B}^{-1} + \boldsymbol{H}^T\boldsymbol{R}^{-1}\boldsymbol{H})(\boldsymbol{x}-\boldsymbol{x}^b) &= \boldsymbol{H}^T\boldsymbol{R}^{-1}(\boldsymbol{y}-\boldsymbol{H}\boldsymbol{x}^b) \\
(\boldsymbol{H}^T\boldsymbol{R}^{-1})^{-1}(\boldsymbol{B}^{-1} + \boldsymbol{H}^T\boldsymbol{R}^{-1}\boldsymbol{H})(\boldsymbol{x}-\boldsymbol{x}^b) &= \boldsymbol{y}-\boldsymbol{H}\boldsymbol{x}^b \\
\big(\boldsymbol{R}(\boldsymbol{H}^T)^{-1}\boldsymbol{B}^{-1} + \boldsymbol{H}\big)(\boldsymbol{x}-\boldsymbol{x}^b) &= \boldsymbol{y}-\boldsymbol{H}\boldsymbol{x}^b \\
\big(\boldsymbol{R} + \boldsymbol{H}\boldsymbol{B}\boldsymbol{H}^T\big)(\boldsymbol{B}\boldsymbol{H}^T)^{-1}(\boldsymbol{x}-\boldsymbol{x}^b) &= \boldsymbol{y}-\boldsymbol{H}\boldsymbol{x}^b \\
\big(\boldsymbol{B}\boldsymbol{H}^T(\boldsymbol{R} + \boldsymbol{H}\boldsymbol{B}\boldsymbol{H}^T)^{-1}\big)^{-1}(\boldsymbol{x}-\boldsymbol{x}^b) &= \boldsymbol{y}-\boldsymbol{H}\boldsymbol{x}^b \\
\boldsymbol{x} &= \boldsymbol{x}^b + \boldsymbol{B}\boldsymbol{H}^T\big(\boldsymbol{H}\boldsymbol{B}\boldsymbol{H}^T + \boldsymbol{R}\big)^{-1}(\boldsymbol{y}-\boldsymbol{H}\boldsymbol{x}^b)
\end{aligned}
$$

which proves the equivalence. Use of the Woodbury Identity (see (116)) gives this result much more directly.

Appendix B

Augmentation

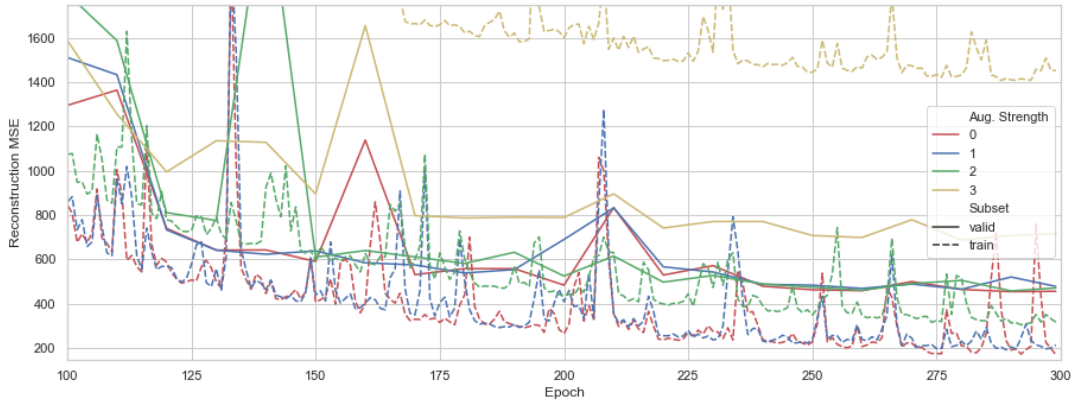

Figure B.1: A repetition of Figure 5.3 without exponential moving average smoothing. The augmentation strengths are repeated in Table B.1 below for reference.

Augmentation Strength	Jitter Amplitude	Jitter Frequency	Jitter Amplitude per Location
0	None	None	None
1	0.005	0.5	0.0025
2	0.05	0.25	0.0125
3	0.1	0.5	0.0500

Table B.1: The augmentation strengths used in Figures 5.3 and B.1. We added Gaussian noise with standard deviation of 'Jitter Amplitude' at 'Jitter Frequency' of the total locations in the state.

Appendix C

Further Comparison

In this Appendix we provide two graphs that would have been repetitious in the full text but provide useful context to the comparison between reduced space VarDA and bi-reduced space VarDA.

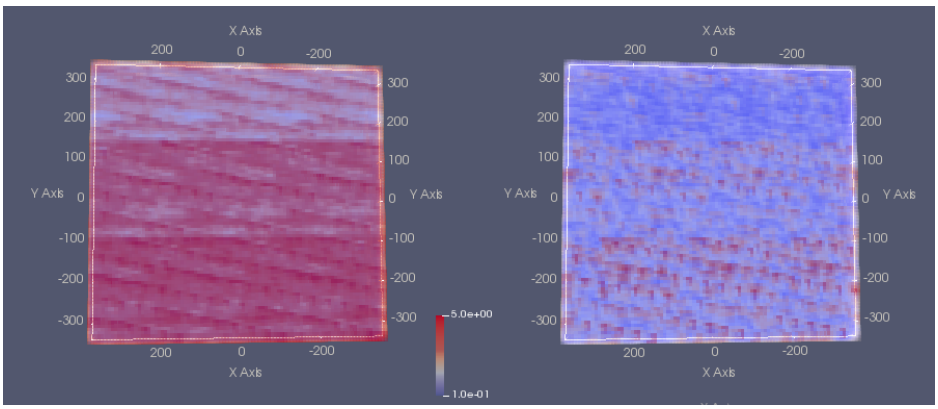

Figure C.1: DA MSE across a slice of the spatial domain averaged over all test-set time-steps. We show a) the reduced-space variant with TSVD ($\tau = 32$ and $M = n$) and b) Bi-reduced space variant with the Tucodec-Next model.

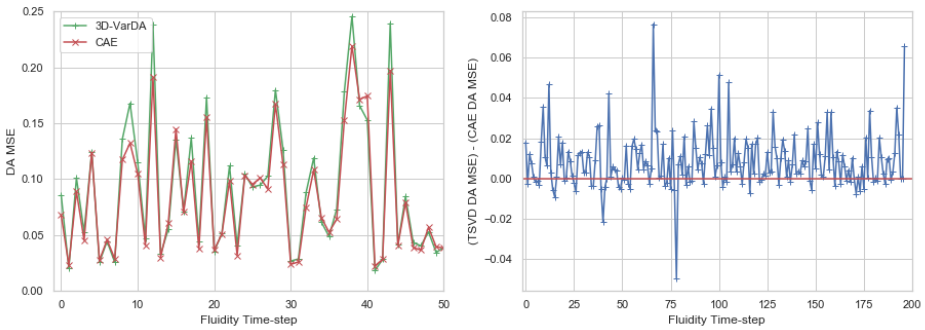

Figure C.2: Repetition of Figure 5.4 with $\tau = 791$ (i.e. no truncation) instead of $\tau = 32$ and $M = n$ as before. Although the performance is more similar in this case, our method still performs better on average. We also note that, in order to achieve this level performance, the reduced space method takes 2.5s, or x43 longer than our approach.

Appendix D

Ethical and Professional Considerations

In this appendix we consider the legal and ethical implication of this work. See Table D.1 on the next page for a completed Imperial College ethics checklist. We will cover the single case which we believe is relevant here.

Copyright Licensing Implications

We have released our `VarDACAE` Python module open-source. We used the MIT Licence and this should be visible on Github. As discussed in section 4.3.2, we used a number of third-party libraries and code so a key ethical consideration is in making sure this is all correctly licensed and attributed. We encountered a challenge in this regard as the CBAM implementation at `https://github.com/Jongchan/attention-module/blob/master/MODELS/cbam.py` did not have a license attached. Therefore, we wrote to the individual and obtained written consent to use the code. We see that, since then, they have uploaded a licence to this repository.

	Yes or no
Section 1: HUMAN EMBRYOS/FOETUSES	
Does your project involve Human Embryonic Stem Cells?	×
Does your project involve the use of human embryos?	×
Does your project involve the use of human foetal tissues / cells?	×
Section 2: HUMANS	
Does your project involve human participants?	×
Section 3: HUMAN CELLS / TISSUES	
Does your project involve human cells or tissues? (Other than from "Human Embryos/Foetuses" i.e. Section 1)?	×
Section 4: PROTECTION OF PERSONAL DATA	
Does your project involve personal data collection and/or processing?	×
Does it involve the collection and/or processing of sensitive personal data (e.g. health, sexual lifestyle, ethnicity, political opinion, religious or philosophical conviction)?	×
Does it involve processing of genetic information?	×
Does it involve tracking or observation of participants? It should be noted that this issue is not limited to surveillance or localization data. It also applies to Wan data such as IP address, MACs, cookies etc.	×
Does your project involve further processing of previously collected personal data (secondary use)? For example Does your project involve merging existing data sets?	×
Section 5: ANIMALS	
Does your project involve animals?	×
Section 6: DEVELOPING COUNTRIES	
Does your project involve developing countries?	×
If your project involves low and/or lower-middle income countries, are any benefit-sharing actions planned?	×
Could the situation in the country put the individuals taking part in the project at risk?	×
Section 7: ENVIRONMENTAL PROTECTION AND SAFETY	
Does your project involve the use of elements that may cause harm to the environment, animals or plants?	×
Does your project deal with endangered fauna and/or flora /protected areas?	×
Does your project involve the use of elements that may cause harm to humans, including project staff?	×
Does your project involve other harmful materials or equipment, e.g. high-powered laser systems?	×
Section 8: DUAL USE	
Does your project have the potential for military applications?	×
Does your project have an exclusive civilian application focus?	×
Will your project use or produce goods or information that will require export licenses in accordance with legislation on dual use items?	×
Does your project affect current standards in military ethics – e.g., global ban on weapons of mass destruction, issues of proportionality, discrimination of combatants and accountability in drone and autonomous robotics developments, incendiary or laser weapons?	×
Section 9: MISUSE	
Does your project have the potential for malevolent/criminal/terrorist abuse?	×
Does your project involve information on/or the use of biological-, chemical-, nuclear/radiological-security sensitive materials and explosives, and means of their delivery?	×
Does your project involve the development of technologies or the creation of information that could have severe negative impacts on human rights standards (e.g. privacy, stigmatization, discrimination), if misapplied?	×
Does your project have the potential for terrorist or criminal abuse e.g. infrastructural vulnerability studies, cybersecurity related project?	×
SECTION 10: LEGAL ISSUES	
Will your project use or produce software for which there are copyright licensing implications?	✓
Will your project use or produce goods or information for which there are data protection, or other legal implications?	×
SECTION 11: OTHER ETHICS ISSUES	
Are there any other ethics issues that should be taken into consideration?	×

Table D.1: Imperial College Ethics Checklist.