ABSTRACT

Title of Thesis: ORDERING NON-LINEAR SUBSPACES FOR AIRFOIL DESIGN AND OPTIMIZATION VIA ROTATION AUGMENTED GRADIENTS

Alec Van Slooten
Master of Science, 2023

Thesis Directed by: Professor Mark Fuge
Department of Mechanical Engineering

Airfoil optimization is critical to the design of turbine blades and aerial vehicle wings, among other aerodynamic applications. This design process is often constrained by the computational time required to perform CFD simulations on different design options, or the availability of adjoint solvers. A common method to mitigate some of this computational expense in non-gradient optimization is to perform dimensionality reduction on the data and optimize the design within this smaller subspace. Although learning these low-dimensional airfoil manifolds often facilitates aerodynamic optimization, these subspaces are often still computationally expensive to explore. Moreover, the complex data organization of many current nonlinear models make it difficult to reduce dimensionality without model retraining. Inducing orderings of latent components restructures the data, reduces dimensionality reduction information loss, and shows promise in providing near-optimal representations in various dimensions while only requiring the model to be trained once. Exploring the response of airfoil manifolds to data and model selection and

inducing latent component orderings have potential to expedite airfoil design and optimization processes.

This thesis first investigates airfoil manifolds by testing the performance of linear and nonlinear dimensionality reduction models, examining if optimized geometries occupy lower dimensional manifolds than non-optimized geometries, and by testing if the learned representation can be improved by using target optimization conditions as data set features. We find that autoencoders, although often suffering from stability issues, have increased performance over linear methods such as PCA in low dimensional representations of airfoil databases. We also find that the use of optimized geometry and the addition of performance parameters have little effect on the intrinsic dimensionality of the data. This thesis then explores a recently proposed approach for inducing latent space orderings called Rotation Augmented Gradient (RAG) [1]. We extend their algorithm to nonlinear models to evaluate its efficacy at creating easily-navigable latent spaces with reduced training, increased stability, and improved design space preconditioning. Our extension of the RAG algorithm to nonlinear models has potential to expedite dimensional analyses in cases with near-zero gradients and long training times by eliminating the need to retrain the model for different dimensional subspaces.

INVESTIGATION OF LATENT SPACES FOR AIRFOIL DESIGN AND OPTIMIZATION

by

Alec Van Slooten

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2023

Advisory Committee:
Professor Mark Fuge, Chair/Advisor
Professor Shapour Azarm
Professor Johan Larsson

Acknowledgments

I would like to express my sincere gratitude to my advisor Dr. Mark Fuge for his guidance and mentorship. His advice was instrumental throughout the entirety of my research. I'd also like to thank Dr. Shapour Azarm and Dr. Johan Larsson for their invaluable and insightful feedback. Thanks to Milad Habibi as well for helpful technical discussions as well. I am also deeply greatful to my family and friends for their support over the years. Finally, I'd like to thank the Department of Energy's Advanced Research Projects Agency-Energy (ARPA-E) for funding that made this work possible.

Table of Contents

Acknowledgements ii

Table of Contents iii

List of Tables vi

List of Figures vii

List of Abbreviations ix

Chapter 1: Introduction 1

Chapter 2: Related Work 7
- 2.1 Overview . . . 7
- 2.2 Dimensionality Reduction . . . 7
  - 2.2.1 PCA . . . 8
  - 2.2.2 Autoencoders . . . 10
  - 2.2.3 Model Selection . . . 12
- 2.3 Airfoil Compact Modeling . . . 13
- 2.4 Databases . . . 13
  - 2.4.1 Case 1: Synthetic problem . . . 14
  - 2.4.2 Case 2: UIUC airfoil database . . . 14
  - 2.4.3 Case 3: SU2 Optimized airfoil database . . . 15
- 2.5 Airfoil Training Size and Dimensionality Estimation . . . 16
- 2.6 Effects of Optimized Geometry in Latent Spaces . . . 17
- 2.7 Feature Selection and Latent Spaces . . . 18
- 2.8 Ordered Models . . . 19
  - 2.8.1 Active Subspaces . . . 19
- 2.9 Ordered Generative Models . . . 19
  - 2.9.1 Dropout . . . 20
  - 2.9.2 Non-Uniform Weight Regularization . . . 21
  - 2.9.3 Rotational Augmented Gradient . . . 22
- 2.10 Optimization . . . 25
- 2.11 Autoencoder Training Stabilization . . . 27

Chapter 3: Effect of Optimal Geometries and Performance Parameters on Airfoil Latent Space Dimension 30

3.1 Overview . . . 30
3.2 Methods . . . 30
3.2.1 Preprocessing . . . 31
3.2.2 Dimension Reduction . . . 31
3.2.3 Model Tuning . . . 32
3.3 Results . . . 34
3.3.1 Model Performance on Optimized Data set . . . 34
3.3.2 Optimized Geometry vs Non-Optimized Geometry . . . 36
3.3.3 Including Parameters . . . 37
3.4 Discussion and Summary . . . 38
3.4.1 How Does Geometry Optimality Affect Latent Space Dimensionality? . . . 38
3.4.2 How Does Including Performance Parameters Affect Reconstruction? . . . 39
3.4.3 How Might the Autoencoder Architecture Alter Our Results? . . . 39
3.4.4 How Might the Activation Functions Alter Our Results? . . . 40
3.4.5 How Might Autoencoders Be Forced to Global Optima? . . . 41

Chapter 4: Non-Linear Latent Spaces for Airfoil Design using Rotation Augmented Gradients 42
4.1 Overview . . . 42
4.2 Theory . . . 42
4.3 Autoencoder Hyperparameter Tuning . . . 44
4.4 Experiment 1: Comparison of Reconstruction Accuracy Between NAEs and RAG 45
4.4.1 Reconstruction Error Methodology . . . 45
4.4.2 Results . . . 46
4.5 Experiment 2: Effect of RAG on Fitting Stability of Non-Linear Autoencoders . . . 51
4.5.1 Methodology . . . 51
4.5.2 Results . . . 51
4.6 Discussion and Summary . . . 53
4.6.1 Limitations . . . 53
4.6.2 How Can Models Be Fairly Compared? . . . 54

Chapter 5: In-Progress Ordered Components for Design Space Preconditioning 55
5.1 Overview . . . 55
5.1.1 Applied Optimization Methodology . . . 55
5.1.2 Autoencoder Hyperparameter Tuning . . . 57
5.1.3 Results . . . 58
5.2 Discussion and Summary . . . 61
5.2.1 Limitations . . . 61
5.2.2 How Well Does the RAG Algorithm Extend to More Complex Autoencoder Structures? . . . 63
5.2.3 In What Situations Is the RAG Algorithm Useful? . . . 64

Chapter 6: Conclusion 65
6.1 Future Work . . . 67
6.1.1 Optimization Implementation Issues . . . 67

6.1.2 Vary Latent Space Size 67
6.1.3 Examine Other Methods of Reducing Dimension of a Trained Model . . . 68
6.1.4 Decrease Number of Feature Vectors 68

Bibliography 69

List of Tables

4.1 VARIANCE OF MEAN TESTING MSE AND BOUNDS OF BOOTSTRAPPED 95% CONFIDENCE INTERVAL BETWEEN ONCE-TRAINED RAG AND NAIVE AE 52

List of Figures

2.1 BASIC AUTOENCODER STRUCTURE 12
2.2 RANDOM SUBSET OF AIRFOILS FROM THE UIUC DATA SET [1] 15
2.3 RANDOM SUBSET OF AIRFOILS FROM THE OPTIMIZED DATA SET [1] . 16
2.4 VISUALIZATION OF LOSS LANDSCAPE WITH EXPECTED CONVERGENCE OF RAG AND NON-UNIFORM WEIGHT REGULARIZATION METHODS . 24

3.1 EFFECT OF TRAINING SAMPLE SIZE ON MEAN PCA AND AUTOENCODER TESTING MSE VALUES 33
3.2 EFFECT OF WEIGHT DECAY ON MEAN TEST MSE AND TRAINING SIZE 34
3.3 HISTOGRAM OF AVERAGE TESTING RECONSTRUCTION ERROR FOR VARIOUS LATENT SPACE SIZES USING OPTIMIZED DATA SET 36
3.4 MEAN PCA AND AUTOENCODER TESTING MSE VALUES VS. DIMENSION ON OPTIMIZED AND UIUC AIRFOIL DATASETS 37
3.5 EFFECT OF INPUT PARAMETERS ON MEAN PCA AND AUTOENCODER TESTING MSE VALUES VS. DIMENSION 38

4.1 COMPARISON OF RECONSTRUCTION ERROR BETWEEN ONCE-TRAINED RAG AUTOENCODERS AND INDIVIDUALLY TRAINED AUTOENCODERS FOR EACH LATENT DIMENSION SIZE ON SYNTHETIC DATA SET 47
4.2 ORIGINAL SYNTHETIC COORDINATES PLOTTED ALONGSIDE 1D REPROJECTED COORDINATE PREDICTIONS USING RAG, AE, AND PCA . . 48
4.3 ORIGINAL SYNTHETIC COORDINATES PLOTTED ALONGSIDE 2D REPROJECTED COORDINATE PREDICTIONS USING RAG, AE, AND PCA . . 49
4.4 COMPARISON OF RECONSTRUCTION ERROR BETWEEN ONCE-TRAINED RAG AUTOENCODERS AND INDIVIDUALLY TRAINED AUTOENCODERS FOR EACH LATENT DIMENSION SIZE ON UIUC AIRFOIL DATABASE . . 50

5.1 INITIALIZATION OF LATENT COORDINATES AND RESPECTIVE PERFORMANCES 56
5.2 OPTIMIZATION WORKFLOW 56
5.3 OPTIMIZATION HISTORY USING RAG AND NAIVE AE MODELS ON THE UIUC AIRFOIL DATABASE 58
5.4 BEST PERFORMING BO-GENERATED AIRFOIL GEOMETRIES USING RAG AND NAIVE AE MODELS ON THE UIUC AIRFOIL DATABASE 59
5.5 OPTIMIZATION HISTORY USING RAG AND NAIVE AE MODELS ON THE OPTIMIZED AIRFOIL DATABASE 59

5.6 BEST PERFORMING BO-GENERATED AIRFOIL GEOMETRIES USING RAG AND NAIVE AE MODELS ON THE OPTIMIZED AIRFOIL DATABASE . . . 60

List of Abbreviations

AE	AutoEncoder
ASM	Active Subspace Method
BO	Bayesian Optimization
CI	Confidence Interval
DR	Dimensionality Reduction
GAN	Generative Adversarial Network
GHA	Generalized Hebbian Algorithm
GP	Gaussian Process
MNIST	Modified National Institute of Standards and Technology database
MSE	Mean Squared Error
PCA	Principal Component Analysis
RAG	Autoencoder with Rotation Augmented Gradient update
SVD	Singular Value Decomposition
UIUC	University of Illinois Urbana-Champaign airfoil database
OPT	OPTimized airfoil database

Chapter 1: Introduction

Airfoil design and optimization is crucial in the design of turbine blades and aerial vehicle wings. While much progress has been made in the past in this field, numerical optimizations are still time consuming. These optimization techniques need to be executed repeatedly to find an optimal design resulting in a process often constrained by the run time [2]. In the case of airfoil geometry, 2D points, or coordinates corresponding to the shape of the airfoil are essentially design variable that require optimization over to determine a design with the best performance. With higher-fidelity models, this optimization becomes time-consuming and inefficient. Using an airfoil geometry with 100 points for instance would necessitate optimization over a 200 dimensional design space.

The dimensionality reduction (DR) type of unsupervised machine learning can be leveraged to help solve this problem. Intuitively, optimizing over a 200 dimensional space is overkill for airfoil design. Best performing airfoils typically share certain properties between all coordinates (such as smoothness). DR creates a smaller subspace from the original data wherein little of the data information, or explained variance, is lost. In other words, dimensionality reduction essentially learns the fundamental design variables that effect all of the airfoil coordinates. Since the dimensionality of the space is lower, optimizations typically converge much faster within this space . Once finished, the results can be reprojected from the latent, or low dimensional space,

back into the real space to obtain the final design. This workflow is becoming increasingly common in many areas of optimization and has been used recently in everything from aerodynamic design optimization [3, 4] to molecular drug design [5] and optimization of robotic motion [6].

Although this methodology has already been established, exploring this reduced subspace may still be computationally expensive. The cost of sampling the design space of high dimensional models often increases exponentially with dimensionality [7]. This phenomenon, dubbed the curse of dimensionality [8], illustrates the importance of minimizing the subspace dimensionality to reduce the computational time necessary for design space exploration. To streamline this reduction, it is critical to understand the properties of low-dimensional design manifolds. With respect to airfoils, the effects of using optimal geometry and including performance parameters the geometry was optimized for are of particular interest. Were the use of optimal geometry or performance parameters to result in lower dimensional manifolds, they could be leveraged to expedite aerodynamic shape optimization. These properties are studied using PCA and autoencoder models as investigative tools to learn generalities about the manifold. In the first part of this thesis, we investigate airfoil manifolds. Specifically, the key contributions of this first part of the thesis are three-fold.

1. We compare the performance of PCA and autoencoders on an optimized airfoil data set to provide insight into each model's behavior with respect to the amount of training data used and the latent space size. Performance was measured by reconstruction testing MSE values using 1-20 dimensions and 0.5% to 98% training data. We show that the autoencoders perform better in low dimensional spaces but PCA has greater stability with extremely low or high fractions of training samples.

2. We test the hypothesis that optimal airfoil geometries live on a lower dimensional space than non-optimized geometries. We found, surprisingly, that this is not necessarily true and show that UIUC airfoil and optimized airfoil data sets have similar intrinsic dimension.

3. We studied the effect of including the properties that the airfoils were optimized for (target lift coefficient, Mach number, angle of attack, and Reynolds number) as features on the intrinsic dimensionality of the latent space. We found that appending these features to the optimized airfoil data set do not result in a significant change in dimensionality.

In the second part of our work, we investigate methods for ordering latent components as a potential solution to deter autoencoder models from getting stuck in local minima. The utility of inducing ordering to precondition design spaces and create a versatile model which can be applied to other dimensional spaces is studied.

One of the greatest computational expenses that constrains the design process is the time required to perform CFD simulations on the design iterations, particularly in highly-turbulent regimes requiring Large Eddy Simulations and where adjoint simulations might suffer from numerical instability [9]. While finding a low dimensional design space is helpful and generally saves computational cost, selecting the right order for this space is often an unclear and difficult process. Furthermore, even after establishing orderings, there is often a need to navigate the design space dimension. Doing so allows for the trade-off of surrogate modeling and optimization costs with accuracy, which is advantageous in iterative design processes.

The goal of surrogate modeling is to mitigate the 'curse of dimensionality' while losing minimal information. Optimization often runs into this same problem of exponential complexity growth with higher dimensional model sampling. To avoid this, some current airfoil optimization

practices involve running optimizations on low dimensional spaces. When these converge and begin to plateau, the resulting information is used to warm-start another optimization in a higher-dimensional space [1]. This is done to avoid sampling the entire high-dimensional space, which would be costly. The drawback of this method is that several models need to be trained: at minimum one trained on the low dimensional space and one on the high dimensional space. Additional spaces are also often utilized to ensure convergence. Training multiple models is thus computationally expensive, and not always necessary in this staged optimization approach. We investigate a method to perform multi-stage optimization by leveraging an algorithm to induce ordering in the latent space and reduce training to just one model.

This method hinges on the ability to induce orderings in the latent spaces of machine learning models, allowing the user to reduce dimensionality while minimizing the amount of information lost in the process. The primary method of ordering latent components in linear models is known as Principal Component Analysis (PCA). In PCA, principal components which account for the smallest percentage of explained variance in the data set are often omitted, resulting in a lower dimensional model that accounts for most of the explained variance. PCA and other methods such as Active Subspaces do well in many cases, but are ultimately limited as linear models and often struggle with more complex, nonlinear data [10, 11].

Many other algorithms for nonlinear models have been proposed to induce orderings of latent spaces. Dropout, nested dropout, and non-uniform weight regularization have all been demonstrated as successful methods for doing this, but are not universally applicable. Moreover, they have been shown to be ill-conditioned for many applications, often leading to slow convergence rates [12].

Furthermore, nonlinear models such as autoencoders frequently suffer from training sta-

bility issues. Training deep neural networks is inherently onerous because the magnitudes of gradients may differ from layer to layer. Autoencoders also have many parameters which require tuning, and the landscape of the objective function may be difficult for gradient descent to navigate [13]. Such training difficulties often lead to the creation of manifolds with incorrect local geometry and sensitivities to noisy training data [14]. As training stability often directly impacts the utility of models, it is critical model property to analyze. Overfitting is also a frequent issue, and is typically addressed via the addition of a penalty or normalization to mitigate weight peaking, or by adding sparsity to the weights to encourage the model to learn less noisy representations [15]. Moreover, autoencoders are also lossy by nature with imperfect decoding. All of these issues combine to make training instability a major complication in autoencoder models.

A recently proposed method to mitigate both problems was given by the Rotation Augmented Gradient method proposed by Bao et al. [12]. Their work proposed an algorithm to order the latent components by applying a rotation to the autoencoder weight matrices and learning this rotation independently from the gradient descent update. This method resulted in faster convergence and efficient recovery of the correct representation, but was only evaluated in linear autoencoders. We extend their work by conducting a series of experiments to test the RAG algorithm on nonlinear autoencoder models and evaluate its ability to generate navigable representations, increase autoencoder training stability, and expedite aeronautical optimization. The key contributions of the second part of this thesis are detailed below

4. We introduce an approach for non-linear latent spaces with ordered principal directions via the use of autoencoders with Rotation Augmented Gradients. We investigate the ability of this single architecture, once trained, to provide similar reconstruction Mean Squared Error

(MSE) to those of multiple individual autoencoders trained for separate latent space sizes.

5. We demonstrate that the proposed model stabilizes autoencoder training, particularly in autoencoders with extraneous nodes, as measured by a reduction in variance in the reconstruction MSE.

6. We experimentally show the ability of the RAG algorithm to precondition the design space for gradient-free optimization, as measured by Bayesian Optimization (BO) convergence speed on a Reynold-Averaged Navier Stokes (RANS) solver.

The results of our key contribution 6 are still in-progress and are relegated to Chapter 5, along with a discussion about the current challenges of implementation.

Chapter 2: Related Work

2.1 Overview

In this chapter we first present an overview of dimensionality reduction and the selected models used in our work. We then explore previous work relating to airfoil compact modeling and introduce the databases used throughout our work. Next, we review methods for estimating dimensionality and previous work relating to latent spaces of optimized geometry data sets. We then examine the impact of feature selection on latent spaces, and review ordered and ordered generative models. Finally, we explore Bayesian Optimization and the training stability for autoencoder models. This chapter provides an inclusive overview of common models, general design and optimization practices, theory, and data relating to the subsequent chapters.

2.2 Dimensionality Reduction

Dimensionality reduction is the process of deriving a set of degrees of freedom which can be used to reproduce most of the variability of a data set [16]. There are numerous methods of DR which are classified by their linearity. The most popular linear methods are PCA and metric Multi-Dimensional Scaling [17]. Commonly used nonlinear models include autoencoders, isomaps, locally-linearly embedding (LLE), non-metric Multi-Dimensional Scaling and kernel

PCA [17, 18]. Although each modeling method comes with its advantages, PCA and autoencoders were selected as the DR models to keep the scope of this investigation sufficiently narrow.

2.2.1 PCA

PCA is a fast and computationally inexpensive method of dimensionality reduction that is used as a baseline in this paper. PCA is a basic linear model commonly used to perform a change of basis on a given data set [19]. Although the transformation is linear, many principal components can be neglected depending on the percentage of explained variance they account for. The explained variance ratio is the percentage of variance attributed by each component and is a typical metric used when determining how many principal components are necessary [20]. Besides speed and simplicity, PCA also has the advantage of axes ordered based on their representational power. This makes it easier to interpret and identify the component with the greatest explained variance [21]. While we direct interested readers to past overviews of PCA [19, 22, 23, 24], we will briefly mention its main elements.

The first step in Principal Component Analysis is to scale and center the data. Since PCA performs a linear change of basis, the goal of this method is to take a data set X represented as a m x n matrix and transform it into Y, another m x n matrix such that for some m x m matrix P,

$$Y = PX \tag{2.1}$$

To perform this operation and determine P, information about the relationships between individual points in X is necessary. While measures such as standard deviation and variance only operate on one dimension, covariance is a measure of how much the the two dimensions vary

from the mean with respect to each other. The standard equation for covariance between X and Y is given as:

$$cov(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)} \tag{2.2}$$

The covariance matrix C can then be calculated using:

$$C = cov(d_i, d_j) \tag{2.3}$$

where d is the dimension of X. In a three dimensional data set with dimensions x, y, and z, the covariance matrix will have the following values [25]:

$$C = \begin{bmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{bmatrix} \tag{2.4}$$

For $i = 1, 2, ..., d$ the eigenvalues, λ_i, and eigenvectors, u_i of the covariance matrix are then calculated using [26]:

$$Cu_i = \lambda_i u_i \tag{2.5}$$

Since eigenvectors are the directions with the most explained variance, ordering them by eigenvalue gives the principle components in order of the explained variance each respective component accounts for. In other words, the directions of the data that account for the maximum variance are ordered into principal components. The components that account for only a small amount of variance are often neglected or truncated, resulting in a linear operator P that transforms the original feature space into a lower-dimensional space that minimizes the total least

squares error between the reconstructed and original data.

This change of basis is useful in many applications but can face struggles when variables are weakly correlated or outliers exist. Unless explicitly given by the training data, even simple variance cannot be captured using this model [27]. Nevertheless, PCA (synonymous with proper orthogonal decomposition) is a commonly utilized model in airfoil decomposition and aeronautical design [28, 29, 30].

Kernel PCA is an extension of PCA that leverages kernel functions in the input space to compute dot products in the feature space, thus enabling PCA to generalize to nonlinear cases. A major challenge in KPCA is selecting the ideal kernel for a given task [31]. Furthermore, KPCA suffers from poor scalability with large data [32]. Although kernel methods may be faster to train than neural networks, they require storage of the training data to make predictions which often offsets this lower initial computational cost of training. On the other hand, neural networks have no need for storing training data after establishing the model parameters, resulting in better scalability [33, 34].

2.2.2 Autoencoders

An autoencoder is a neural network commonly used in unsupervised learning. A basic visual of this model is displayed in Fig 2.1. Autoencoders are composed of two primary structures: the encoder and the decoder. Fundamentally, the encoder takes the original data in the $\mathcal{X}$ space (the original coordinate space) and learns a low dimensional representation of this data in the $\mathcal{Z}$ space (the latent space). The latent dimension (z) is calculated using Equation 2.6 where σ is the

activation function and b is the bias.

$$z = \sigma(W^\top x + b) \tag{2.6}$$

The decoder reprojects this representation back into the original domain, transforming it from the $\mathcal{Z}$ back to $\mathcal{X}$ space. This is represented in Equation 2.7 where the activation, weights, and biases may differ from those of the encoder.

$$x' = \sigma'(W'^\top z + b') \tag{2.7}$$

The encoder and decoder structure is necessary to evaluate the performance of the model. The weights applied to each neuron are learned by evaluating the reconstruction error between the original samples and the samples that have been passed through the model. This reconstruction error is typically the MSE between each element of the input x and each element of the output x', as shown in Equation 2.8 .

$$\mathcal{L} = \|(x - x')\|^2 \tag{2.8}$$

Additional terms may be added to this loss function as a form of regularization. A common example of this is the sparsity-inducing $L1$ norm which acts to shrink the autoencoder weights to zero and thus inhibit the weights from growing too large and the model from overfitting.

Autoencoders are more computationally expensive processes that are better able to model nonlinear data and data with recurring patterns [10, 11]. Nonlinear deep autoencoders typically reduce dimensionality better than linear methods such as PCA [35, 36]. In fact, PCA is a special

case of an autoencoder using squared loss and identity activation functions [37]. Additional layers and nonlinear activation functions allow autoencoders to model more complex and nonlinear functions.

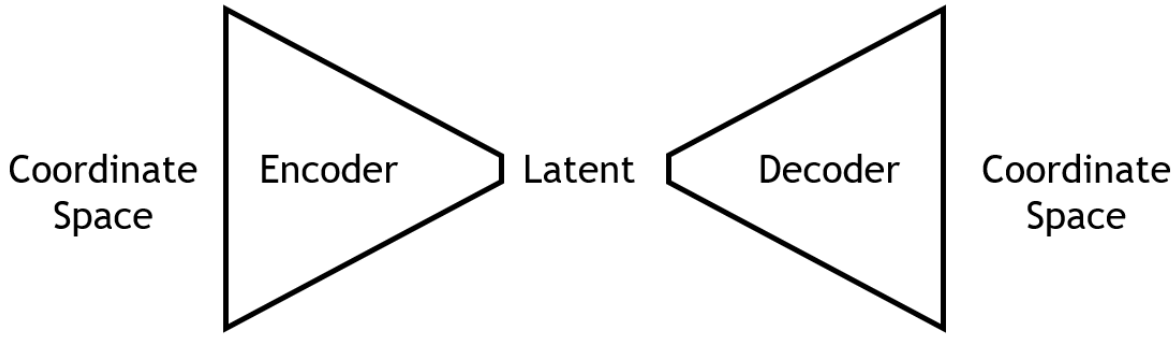

Figure 2.1: BASIC AUTOENCODER STRUCTURE

2.2.3 Model Selection

Our work uses both PCA and autoencoder models to reduce data set dimensionality. These models were selected to establish comparisons between linear and nonlinear models as well as between models producing ordered and unordered latent spaces. PCA was selected because as described in Section 2.2.1 it orders components based on the amount of information about the data each component accounts for. This ordered structure results in an easily-interpreted latent space which can be reduced to a lower dimension by simply omitting the principal component accounting for the least explained variance. On the other hand, autoencoders are nonlinear models which are trained to minimize the reconstruction loss and as such do not typically have ordered latent components. This lack of ordering decreases the interpretability of the latent space and adds to the difficulty of reducing the latent dimension without retraining. The ability to add ordering by effectively changing the weight regularization scheme of an autoencoder using the

Rotational Augmented Gradient algorithm is examined later in this work.

2.3 Airfoil Compact Modeling

Previous works in airfoil decomposition have generally used proper orthogonal decomposition (POD). This method is synonymous with PCA and Karhunen–Loève expansion [38]. POD has been successfully implemented for airfoil design optimization, reducing computational cost by approximately three orders of magnitude for a two-dimensional invicid flow calculation [29]. It has also been applied to micro air vehicle wing CFD simulation data [39] and demonstrated to be effective for reconstructing flowfields from incomplete aerodynamic data sets [30].

More recently, models such as generative adversarial networks and autoencoders have been used in airfoil decomposition. A modified generative adversarial network dubbed Bézier-GAN has been applied to learn an interpretable low-dimensional space that encodes major shape variation of aerodynamic designs [1]. Autoencoders were applied to unsteady flows around a two-dimensional airfoil [40], as well as inverse design of airfoil shapes [41]. These deep learning techniques aim to improve upon the limitations of linear methods and model complex and non-linear data with greater accuracy.

2.4 Databases

Models were tested using three main data sets: a synthetic pedogogical example, the UIUC airfoil database, and an optimized subset of the UIUC airfoil database.

2.4.1 Case 1: Synthetic problem

To initially develop our models and validate their performance against other methods of dimensionality reduction, we first created a synthetic data set. This set consists of 100 points from a simple 1D function projected in 3D space given by $[2t, t^2, -t]$ for $-1 \leq t \leq 1$. We then applied a centered Gaussian noise of with a standard deviation of 0.02 to these points to create complexity. This creates a regime where the performance of linear models is functionally limited and nonlinear models are more applicable. This data set is still relatively simple and is used as an initial benchmark of model performance.

2.4.2 Case 2: UIUC airfoil database

To validate model performance on a real airfoil data set, we selected the University of Illinois Urbana-Champaign (UIUC) airfoil database. This database consists of 2D coordinates for approximately 1,600 airfoils created for various flow conditions and applications such as wind turbines, and a wide variety of aircraft including UAVs [42]. This data set has previously been used in airfoil dimensionality reduction studies [43, 44, 45] as well as in several inverse design applications [46, 47], allowing us to compare our approach to those past published studies. Previous work has established that this database has an intrinsic dimensionality of approximately four dimensions [43]. This data set is used both as a stand-alone test of the RAG model performance and in the optimization trials. A randomized subset of these airfoils are visualized in Fig. 2.2.

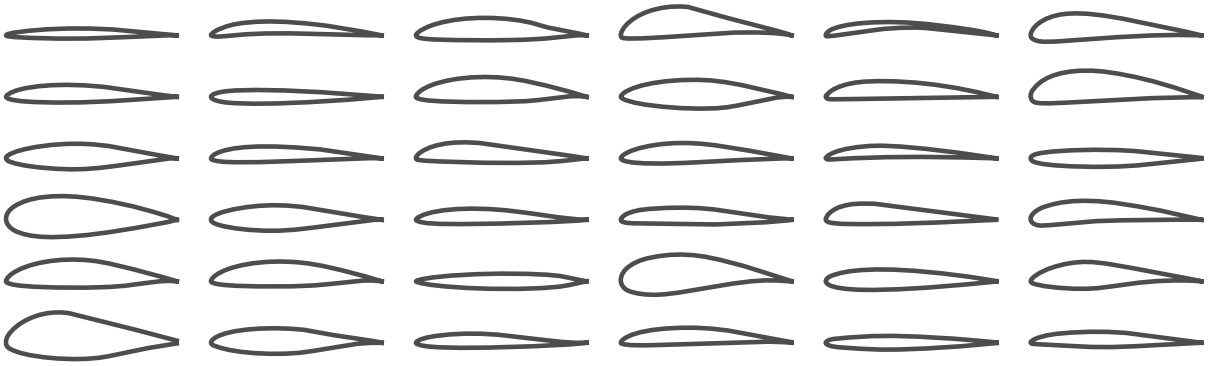

Figure 2.2: RANDOM SUBSET OF AIRFOILS FROM THE UIUC DATA SET [1]

2.4.3 Case 3: SU2 Optimized airfoil database

The final dataset used in this study is an optimized airfoil data set, originally generated using the open-source PDE analysis toolset SU2 to perform gradient-based shape optimization on a subset of the UIUC airfoils. This adjoint optimization used the Hick-Henne parameterization method to maximize lift over drag at a constant lift coefficient and to determine optimal angle of attack of the generated airfoil. This procedure was conducted with eight diverse candidate airfoils using 1042 sets of input conditions (Mach number, Reynolds number, and target lift coefficient). At each input condition, each candidate airfoil was optimized and the best performing shape was selected, resulting in 1042 sets of 192 optimized 2D airfoil coordinates, along with optimized angle of attack and input boundary conditions for each airfoil [48]. A randomized subset of these airfoils are visualized in Fig. 2.3. Even though the candidate samples used in generating this dataset were selected using Latin Hypercube sampling ensure sample diversity, many of the airfoils share similar shapes when compared to the UIUC airfoils in Fig. 2.2. This database was initially developed and used for inverse design of airfoils using generative adversarial networks [48]. It has since been utilized in conjunction with the UIUC database in assessing

the effect of optimized geometries on the latent space [45]. For a more detailed analysis of the optimization process, we refer the reader to Chen et al. (2021) [48].

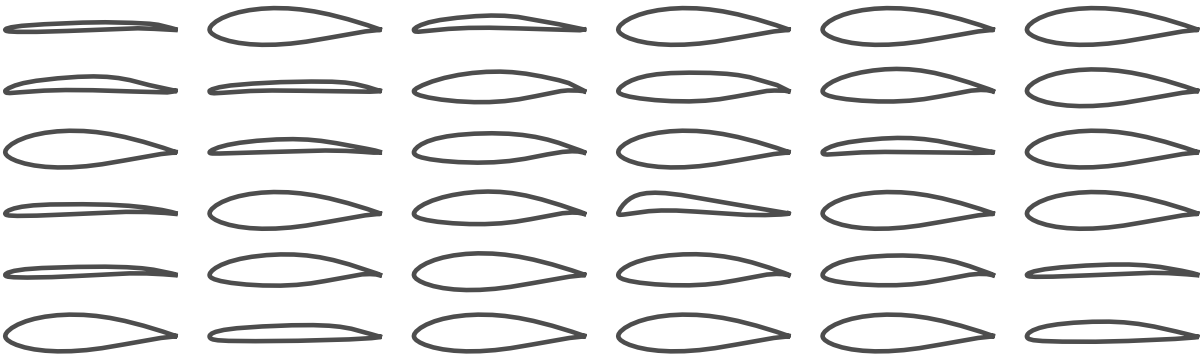

Figure 2.3: RANDOM SUBSET OF AIRFOILS FROM THE OPTIMIZED DATA SET [1]

Experiments in Chapters 3 and 5 made use of cases 2 and 3, while experiments in Chapter 4 used cases 1 and 2.

2.5 Airfoil Training Size and Dimensionality Estimation

Having selected the models and established the airfoil databases, determining model latent space size is the next essential step. The dimensionality of the latent space is a crucial model hyperparameter that has been shown to greatly impact the autoencoder model accuracy [49]. The low complexity MNIST data set [50] is a large set of handwritten numbers and labels commonly used for training dimensionality reduction models [35, 51, 52]. In a previous study using this database, best results were achieved when the number of hidden layer nodes in the autoencoder is set around the intrinsic dimensionality of the data [35]. This concept of intrinsic dimensionality refers to the minimum number of components or variables necessary to represent the data. Although this specific trend was not observed in a greater complexity data set in the same study, it is clear that latent space size is an important modeling parameter. Intrinsic dimension also has

a close correlation with the number of samples needed for learning [53]. The effects of latent space dimensionality and sample size are investigated in this paper.

A common method to estimate data set dimensionality is to perform Maximum Likelihood Estimation (MLE). This method has been applied to a range of real and simulated datasets, demonstrating its performance over other dimensionality estimators [54]. The efficacy of MLE was confirmed on the UIUC airfoil database as estimates were found to agree with brute-force hyperparameter tuning [44]. Estimators such as MLE rely on independent and identically distributed data as well as smooth and locally uniform data density. While MLE has been previously applied to the UIUC database, it is not clear that these assumptions were met. Since the optimized airfoil data set runs into this same issue, intrinsic dimensionality is determined based off model performance.

2.6 Effects of Optimized Geometry in Latent Spaces

While this investigation is specifically interested in the effect that these models have on uncovering the intrinsic dimension of non-optimized and optimized geometries, to the best of our knowledge we were not able to find prior research addressing this question. The closest field of research is in geometric parameterization for dimensionality reduction where largely studies have centered around automotive evolutionary shape optimization. Rios et al. explored the efficiency of the bottleneck layer of a point cloud autoencoder as a geometric representation for optimization [55]. In later work, aerodynamic drag and lift of five car shapes were optimized within latent spaces of PCA, K-PCA, and autoencoder models to identify model advantages in shape optimization. Autoencoder representations were shown to be more sensitive to changes

in latent variables [56]. While this is not directly related, aerodynamic optimizations of vehicles share many similarities with airfoils. These types of techniques have been helpful in investigating the latent space of various optimization generations. From intuition, optimized geometry should generally be able to be represented in fewer dimensions than an entire database of non-optimized geometries.

2.7 Feature Selection and Latent Spaces

Using all of the features of a data set is not always necessary to create an efficient and accurate model. In fact, several studies have found that using more features than necessary may increase the computational load and thereby slow down the learning process, while yielding similar results as those obtained with a much smaller feature subset [57, 58, 59]. For these reasons, a common goal of feature selection is to use as few features as possible. Although it is generally best to follow this rule of thumb, features having a direct impact on the latent space representation may be useful points to consider.

Disentanglement and consistency are shared properties favorable latent spaces [60, 61]. One such measure of consistency within this basis is Latent Space Consistency, which uses the Pearson coefficient as a metric of how the data changes along any basis of the latent space [61]. While we do not apply these metrics since the autoencoder latent space does not specifically refer to the parameters; we focus on using the MSE between testing data and reprojected samples to evaluate the performance of each latent space dimension.

2.8 Ordered Models

Inducing orderings in latent spaces has been previously explored, primarily for linear models. Most previous work in this field can be split into either PCA or active subspace methods. A thorough overview of PCA is provided in Section 2.2.1, so in this section we focus on active subspace methods.

2.8.1 Active Subspaces

The Active Subspace Method (ASM) was first proposed as a method to mitigate the 'curse of dimensionality' by finding a low dimensional subspace that captures the trends in the objective function [62]. This subspace is generated by detecting important directions in the model input space along which the target function has the greatest variability [63]. ASM differs significantly from PCA in that it reduces the input space via gradients and model outputs, while PCA simply uses the data's covariance matrix[64]. ASM was originally developed for uncertainty quantification and has since been implemented in a wide range of applications including analysis and attacking of deep neural networks [65], surrogate-based aerodynamic shape optimization [66], and accelerating the design process of a multi-fidelity Bayesian Optimization framework for materials design [67].

2.9 Ordered Generative Models

To induce orderings in the latent space of a generative model, most previous work focuses on the following areas: dropout strategies, non-uniform weight penalties, and rotational aug-

mented gradients. These methods are regularization methods which are applied to autoencoders to reduce overfitting and make latent spaces more interpretable.

2.9.1 Dropout

Dropout is a method for reducing overfitting by omitting a fraction of the feature detectors randomly at each iteration. This limits the dependence of each hidden unit on others, and thereby encourages each unit to learn useful independent features. Dropout has been shown to increase model performance over standard feed-forward neural networks on the MNIST dataset, while being simple to implement [68].

A modification to this method called Nested Dropout, applies a regularization scheme for stochastic removal of neural network units. This method differs from dropout in that it assigns a distribution over nested subsets of representation units rather than using an independent distribution over each unit. The nesting process results in an ordering of latent components by their importance in terms of information capacity. The ordering of components in nested dropout mitigates the issue of redundancy in autoencoders, resulting in a reduction of latent space complexity. This method was shown to recover PCA exactly in the case of a linear autoencoder [69].

Nested dropout has been successfully implemented in flow normalization, where significant redundancies exist in how flows parameterize distributions [70]. This method has also been applied to convolutional neural networks, where it was shown to be efficient in learning test accuracy and model capacity correspondences [71]. Deterministic versions of Nested Dropout on linear autoencoders were explored by [12].

Despite its capabilities, Dropout has significant limitations. It was found to be impractical

for flows with expensive inverses due to doubled training time, and other methods were recommended for domains with poorly defined data space densities [70]. Furthermore, the condition number of deterministic Nested Dropout was shown to grow (at minimum) quadratically with latent dimension, causing slow convergence [12]. This same phenomenon was observed in another work which found high ordered units to be less stable and have increased error [72]

2.9.2 Non-Uniform Weight Regularization

Non-uniform L_2 regularization is another scheme often applied to autoencoders to encourage decaying network weights. With respect to linear autoencoders, this regularization scheme has been shown to converge to learning the ordered, axis-aligned principal components[12]. Linear autoencoders with L_2 regularization were proven to be symmetric at all critical points and to learn the principal directions in an ordered fashion [73]. More recent work further advanced this idea, developing a model called Principal Component Analysis Autoencoder to both disentangle the latent space and to impose order on it. A series of models were trained by iteratively increasing the latent space size, wherein the combination of L_2 regularization and an additional loss term were used to minimize the magnitude of covariance matrix off-diagonal entries [74]. While this work did succeed in organizing the latent space into statistically independent components in order of decreasing importance, it was noted that the non-uniform L_2 regularization used was sub-optimal. Unfortunately, Non-Uniform Weight Regularization also suffers from the same problem as nested dropout: the ill-conditioned Hessian results in slow convergence [12]. Although these two methods can be used to order latent components, they are generally not the best solutions.

2.9.3 Rotational Augmented Gradient

Learning optimal representations is inherently slow in ordered representations learned using nested dropout and non-uniform L_2 regularization. This is primarily because the objectives of the previous methods result in ill-conditioning that is exacerbated by increasing the latent dimension. RAG is essentially an update rule that accounts for the latent space "rotation" along with the standard gradient. This method has local linear convergence to the axis-aligned optimal representations. The RAG algorithm, recently introduced by Bao et al. was demonstrated to have better symmetry breaking properties and expedite convergence to optimal representations compared to other methods of inducing latent space ordering [12].

The difference between other gradient-based methods such as non-uniform $L2$ weight regularization and the RAG algorithm is visualized in Fig 2.4 on a simple loss landscape defined such that shallow gradients exist along the subspace. Most gradient-based method like non-uniform weight regularization would would first converge to the subspace before slowly moving around it and eventually converging to the optima. This bi-stage approach is a result of the weak symmetry-breaking properties of non-uniform weight regularization and the shallow gradients in the rotational direction which decelerate convergence of gradient-based methods that traverse along it. The Rotational Augmented Gradient update addresses this by explicitly accounting for the rotation in the latent space. Doing so provides stronger symmetry breaking and mitigates the issue of slow rotational convergence, as rotation is used in every update. As displayed in Fig 2.4, Accounting for this rotation can result in faster and more direct convergence to the optima.

The RAG pseudocode is given in Algorithm 1 [12]. After initializing the encoder (W_1) and decoder (W_2) weight matrices, the gradient of these matrices is calculated using the autoencoder

reconstruction loss function $\mathcal{L}$. The latent representation Y_t is then calculated using the encoder and data (X) weight matrices, and used to determine the skew-symmetric matrix A_t. Finally, the weight updates to the encoder and decoder are calculated using A_t to perform a first-order Taylor approximation of the weight matrices rotations. The term α refers to the learning rate and n refers to the number of data points in the weight update equations.

Algorithm 1 RAG

for t = 0, ..., T - 1 **do**
$\nabla(W_1)_t = \nabla_{W1}\mathcal{L}((W_1)_t, (W_2)_t)$
$\nabla(W_2)_t = \nabla_{W2}\mathcal{L}((W_1)_t, (W_2)_t)$
$Y_t = (W_1)_t X$
$A_t = 0.5(\triangledown(Y_t Y_t^\top) - \triangle(Y_t Y_t^\top))$
($\triangledown$ and $\triangle$ operators mask lower and upper matrix triangles with 0)
$(W_1)_{t+1} \leftarrow (I + \frac{\alpha}{n} A_t)(W_1)_t - \alpha\nabla(W_1)_t$
$(W_2)_{t+1} \leftarrow (W_2)_t(I - \frac{\alpha}{n} A_t) - \alpha\nabla(W_2)_t$
end for

In a linear autoencoder, the weight update can be manipulated to form Equation 2.9.

$$W \leftarrow W + \frac{\alpha}{n}[(YX^\top - \triangle(YY^\top)W) - 0.5(YY^\top - diag(YY^\top))W] \tag{2.9}$$

Assuming balanced weight initialization, the first two terms of Equation 2.9 are equivalent to the Generalized Hebbian Algorithm (GHA), a method to train neural networks to find the eigenvectors of the input distribution autocorrelation matrix [75]. In other words, the first terms of the weight update learn the PCA subspace and are responsible for the algorithm's property of inducing latent orderings. The final term, $-0.5(YY^\top - diag(YY^\top))W$, represents the non-reconstruction gradient component of the RAG algorithm, which decays correlation between the columns in W by performing a first-order rotation of the latent representations. The RAG algorithm was constructed specifically for loss landscapes such as the one visualized in Fig 2.4 where

using rotation is likely to amount to stronger symmetry breaking and faster convergence to the optimal representation than regularization schemes that do not use rotational information. For more information about the RAG algorithm, including detailed proofs and derivations, we direct the reader to the original formulation by Bao et al [12].

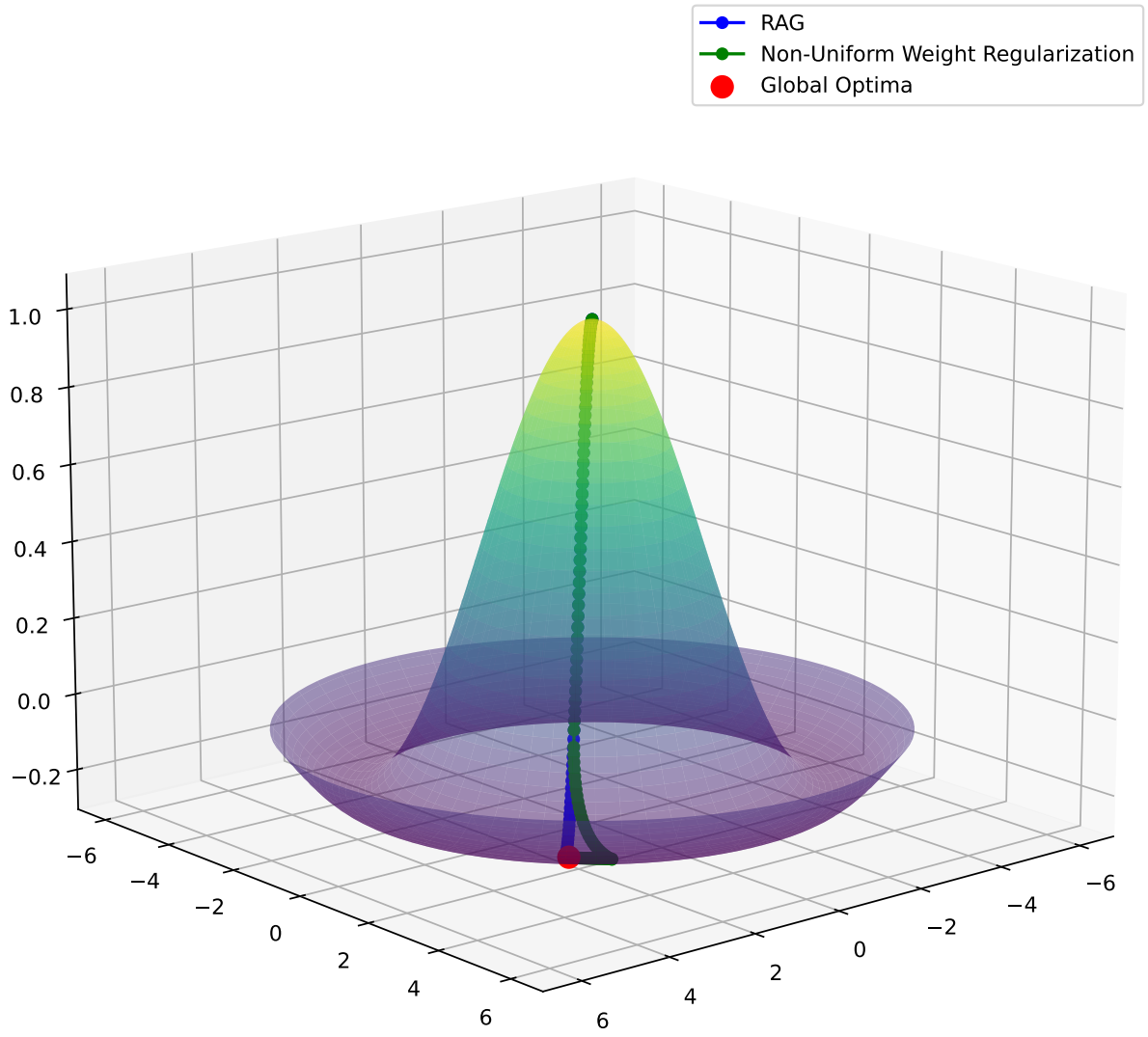

Figure 2.4: VISUALIZATION OF LOSS LANDSCAPE WITH EXPECTED CONVERGENCE OF RAG AND NON-UNIFORM WEIGHT REGULARIZATION METHODS

2.10 Optimization

Current airfoil optimization practice is split into two major approaches: gradient-based and gradient-free methods. Within gradient-based methods, adjoint methods and automatic differentiation are the standard modi operandi for the calculation of gradients based on CFD simulations. Since these methods are fast and provide exact gradients, they have been used in many past aerodynamic design optimization works [76, 77, 78, 79]. Unfortunately, gradient-based methods are not universally applicable. For example, gradients are not always accessible in an appropriate multi-physics solver. Even with accessible gradients, gradient-based methods often get trapped in local optima due to zero gradients or suffer from numerical instabilities.

In this paper, we focus on gradient-free optimization, which mitigates some of these issues, though it is not without its own disadvantages. Specifically, we focus on a surrogate-based method known as Bayesian Optimization (BO).

Bayesian Optimization is a method of determining global optima of black-box objective functions using previous information about the function to determine the next point to sample. Instead of using the objective function $f(x)$ directly, which is often unknown, BO leverages a Gaussian Process (GP) surrogate model which approximates the $f(x)$ based on previously sampled points. This GP model creates Bayesian posterior probability distribution of the objective function, which is then used to find the greatest probable area of $f(x)$ to sample to achieve improved performance. For best results, the surrogate model should be a reasonable approximation of the objective function.

Covariance functions, or kernels, are used in the GP model to evaluate the correlation between points. Since the kernel selection determines the shape of the GP prior and also impacts

the posterior, hyperparameter studies are often performed to determine the best kernel functions to use for improved fit between surrogate model and objective function. Radial basis function, power exponential, and Matern kernels are often selected.

Once the prior has been determined an acquisition function is needed to select the next point to sample. In general, the next point to sample (x_t) is given by:

$$x_t = argmax_x(\sigma(x)) \tag{2.10}$$

where $\sigma(x)$ is the acquisition function [80].

There are many types of acquisition functions. Several popular options include Maximum Probability of Improvement, Upper Confidence Bound, and Expected Improvement. The different acquisition functions have various trade-offs of design space exploration and prior exploitation. After a point is sampled, the posterior of the GP model is updated and the process is iterated. This method is primarily used when evaluations of the objective function are computationally expensive. Bayesian Optimization is also useful because it does not require derivatives or convexity to find global optima.

There are two major disadvantages to BO. First, like many other methods that work well in low-dimensional optimization, BO struggles in high-dimensional parameter spaces. Good coverage of the parameter space is necessary to ensure global optima are found, but the number of evaluations needed to provide this coverage increases exponentially with dimensionality [81, 82]. Secondly, BO requires the objective function to be accurately modeled by the Gaussian Process. If the Bayesian posterior probability distribution does not accurately reflect the objective function, the candidates proposed by the acquisition function may not result in an optimal solution.

Fitting the data to a probability distribution relies heavily on the selected kernel and on the data. Preconditioning this data has been shown to reduce bias and variance, resulting in less noisy search directions and fewer necessary evaluations, ultimately accelerating training [83]. Preconditioning the design space essentially makes it easier for the Gaussian Process to create a more compact probability distribution that requires fewer evaluations to explore. Common preconditioners for Gaussian processes include randomized and truncated Singular Value Decomposition (SVD), randomized Nystrom, RFF, QFF, and partial Cholesky [83]. Matrix factorization models like SVD are closely related to PCA [84, 85]. Other methods of inducing latent space ordering such as RAG have potential to better precondition the design space, allowing for accelerated optimization convergence.

2.11 Autoencoder Training Stabilization

The reference paper Bao et al.notes that the RAG algorithm will contribute to training stability[12]. This term is nebulous and may refer to concepts such as convergence consistency, or reduced oscillations in performance through optimization. We interpreted training stability to refer to the repeatability of model creation with varying training data, using the variance of model performance as a metric.

Training stability is a highly desirable trait of most machine learning models. This concept is a measure of algorithm consistency with respect to training sample perturbations [86].The primary method of estimating how well machine learning models generalize on perturbed training data is a data resampling method known as cross-validation [87, 88]. In general this method mitigates overfitting by splitting the original data into subsets, and testing the model on a different

subset than it was trained with. The average model performance after repeating this process with different subsets is an estimate of final model performance [89]. Bootstrapping is often used to estimate standard errors and to calculate confidence intervals [90]. Applying the bootstrap to the cross-validated models can create estimates of the variance and confidence intervals.

Previous work addressed autoencoder stability by comparing the average standard deviation of feature coefficients across a number of iterations for each model, or by using the ratio of standard deviation to mean as a normalized stability metric [91]. Other work evaluating state autoencoders used the bitwise variance of the state encoding to directly measure the representation stability [92].

In contrast to past work, we measure stability using the average variance of the mean squared reconstruction errors across test sets created using randomly split training and testing data. This was computed using the standard equation given by Equation 2.11 on each model and averaging the resulting variance between trials.

$$S^2 = \frac{\Sigma(x_i - \bar{x})^2}{n-1} \tag{2.11}$$

Where S^2 is the sample variance, x_i is the observed value, $\bar{x}$ is the total observed mean, and n is the number of observations. We selected this metric because when cross-validation is implemented, the model will sample different points at each iteration, resulting in variation in the surrogate model coefficients. While we could compare these different coefficients directly as in Shankaranarayana and Runje [91], it is perhaps more intuitive to examine the robustness of the model by simply noting how the final reconstruction error of the model changes across iterations. The smaller the variance of this error, the more stable the model is. We additionally implement a

bootstrapped empirical 95% confidence interval to graphically depict our results.

Chapter 3: Effect of Optimal Geometries and Performance Parameters on Airfoil Latent Space Dimension

3.1 Overview

In this chapter, latent airfoil manifolds are investigated. We compare the performance of PCA and autoencoders on an optimized airfoil data set to provide insight into each model's behavior with respect to the amount of training data used and the latent space size. We test the hypothesis that optimal airfoils live on a lower dimensional space than non-optimized geometries and study the effect of including the properties that the airfoils were optimized for as features on the intrinsic dimensionality of the latent space.

3.2 Methods

In contrast to the related work, our contributions focus on the effects of optimized geometry and training models using performance parameters as additional features on the latent space dimensionality. To evaluate these contributions, we performed three steps: (1) preprocessing of the airfoil data sets, (2) construction of the dimension reduction models, and (3) model hyperparameter tuning.

3.2.1 Preprocessing

Preprocessing of the data was necessary to ensure all features were equally valued. The parameters of Reynold's number, Mach number, angle of attack, and target lift coefficient for each individual sample were added to the optimized airfoil data set to create a new third set. A data frame was then created for each set, where all values were scaled between 0 and 1 using the MinMaxScaler from scikit-learn [93].

3.2.2 Dimension Reduction

After scaling the data sets, two main trials were conducted to evaluate our models. The first trial compared our models to better understand the effects of using PCA and autoencoders on variable training fractions. The second trial holds this training fraction constant and evaluates model performance with respect to latent space size.

Trial 1 iterated through fractions of training data using the optimized airfoil data set and fixed latent space dimensions. Sample sizes ranged from 5-1025 airfoils to get a full range of sizes while saving close to 2% of the data for testing. The lower bound was selected to be 5 since the number of components must be less than or equal to than the number of samples. For each sample size, 25 training and testing splits were conducted so that reconstruction error could be averaged and a confidence interval could be determined. This was done in sub-trials with latent space sizes of 3, 4, and 5 for PCA and autoencoder models.

Trial 2 iterated through latent space sizes of 1-20 while holding the training testing split constant at 80%-20%. For each latent space size, 25 training and testing splits were conducted. This cross validation was performed so reconstruction error could be averaged over many training

and testing splits, and to generate confidence intervals for PCA and autoencoder models. This trial was conducted with three datasets: UIUC airfoil, optimized airfoil, and the optimized airfoil data set with target optimization conditions.

3.2.3 Model Tuning

After creating these models, optimization of their performance is necessary to ensure a fair comparison between PCA and autoencoders. This is accomplished via hyperparameter tuning. One such hyperparameter that requires fine tuning is an autoencoder's learning rate. This sets the step size of the optimizer at each iteration while it is minimizing loss. Since its value directly affects the training of the model, it also impacts the model accuracy. If this parameter is too small, the model will likely get trapped in local minimum. If this parameter is too large, its step size will be too big resulting in passing over the global minimum.

To mitigate the effects of learning rate on the model accuracy, this value needs to be optimized. From grid search hyperparameter tuning, the optimal learning rate was found to be approximately 1.0×10^{-4}. Additionally, ReduceLROnPlateau, a PyTorch dynamic scheduler, was utilized to further optimize the learning rate [94]. This scheduler works by reducing the value of the learning rate every time that learning stagnates and the loss remains the same for a given number of epochs. Unfortunately, after tweaking the parameters of this function, the results were no better than the constant learning rate, so this constant value was used throughout experimentation. An additional sensitivity study suggested that further optimization of learning rate would not result in significant performance enhancement with the data and model architecture used.

Model architecture may additionally impact autoencoder accuracy. The encoder was simplified to have one fully connected layer followed by a hidden layer with 1-20 nodes depending on the trial. The decoder is identically opposite to the encoder, expanding from the hidden layer to the fully connected layer. The number of nodes in the hidden layer were variable to account for many latent space sizes. The number of features in the data set were used as the number of nodes in the fully connected layers. This value was typically 384, however it was increased to 388 in the trial where additional airfoil properties were included as features. Reconstruction error was calculated by taking the MSE of encoder input and decoder output. We conducted a brief study by examining the testing MSE of autoencoders with additional hidden layers, finding that extra hidden layers do not result in significant performance benefits on the optimized airfoil data set. However, the inclusion of additional hidden layers resulted in increased computational expense. For these reasons only the simple autoencoder with one hidden layer was used throughout the study.

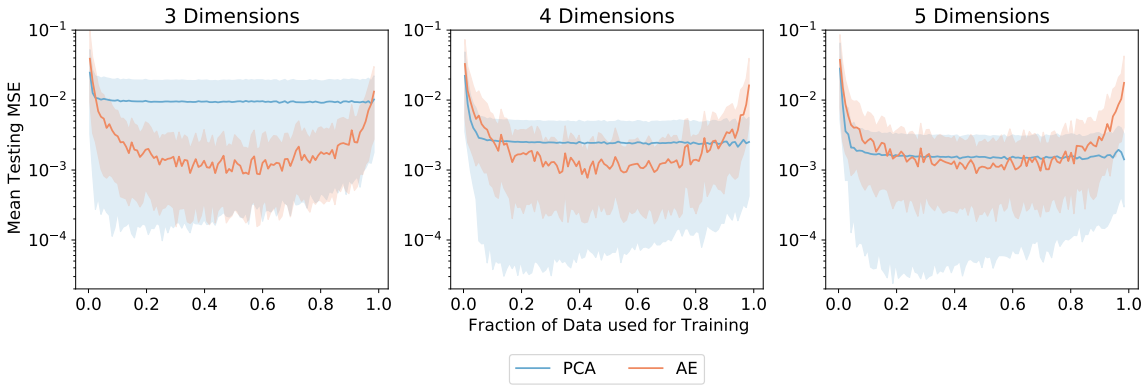

Figure 3.1: EFFECT OF TRAINING SAMPLE SIZE ON MEAN PCA AND AUTOENCODER TESTING MSE VALUES

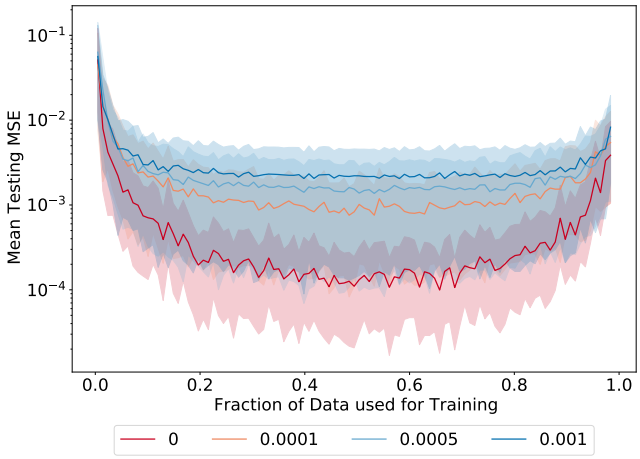

Figure 3.2: EFFECT OF WEIGHT DECAY ON MEAN TEST MSE AND TRAINING SIZE

3.3 Results

This section first demonstrates the model performance on the optimized airfoil data set. We then highlight the effects on the latent space of using non-optimized airfoils and finally exhibit the effects on the latent space of adding the specific input conditions the airfoils were optimized for as features of the data set.

3.3.1 Model Performance on Optimized Data set

Figure 3.1 displays the averaged mean testing MSE for each sample size along with bootstrapped empirical 95% confidence intervals over mean testing MSE values. Both PCA and autoencoder (AE) models are evaluated with latent space sizes of 3, 4, and 5 dimensions. In low dimensions, the autoencoder models have a significantly more narrow confidence interval and perform better than PCA on the majority of the training data fractions. With larger latent space sizes, extreme training sample sizes begin to result in high MSE values. As latent size increases, the mean testing confidence interval for PCA narrows considerably until it is similar to that of

autoencoders at five dimensions. Please also note that it is somewhat deceptive to evaluate the confidence interval in a log-based scale. In this case the upper bound is substantially more telling of the confidence interval than the lower bound.

Secondly, the mean testing MSE of the autoencoder models increase considerably with high training sample fractions and greater latent dimensions. The confidence interval is much wider in this area for the autoencoder than for PCA. This may have occurred simply because the testing data size was small or because the autoencoder models became overfit with large sampling sizes. On the other hand, PCA is more stable with extreme training sample sizes and there was no systematic increase in testing MSE in the high training sample size regiment.

To address the concerns of overfitting, L2 regularization was applied and the previous trial was repeated for several weight decay parameters. Figure 3.2 shows that increasing weight decay results in higher test MSE values. From these results, there does not appear to be much overfitting as an increase in the weight decay parameter (as shown below the plot) typically corresponds to an increase in testing MSE. Nevertheless, there is still significant variance in testing MSE between cross validation trials that could potentially obfuscate overfitting.

Figure 3.3 displays histograms of the reconstruction error for each testing sample averaged over 25 trials for latent space sizes of 3, 4, and 5 dimensions. This suggests that in low dimensional spaces, autoencoders perform significantly better than PCA on this data set. Figure 3.4 shows the full story where autoencoder models start off with lower reconstruction error with low latent space dimensionality but are outperformed in higher dimensions by PCA. We interpret this as the autoencoder is likely getting stuck in local optima, a common issue for this model type.

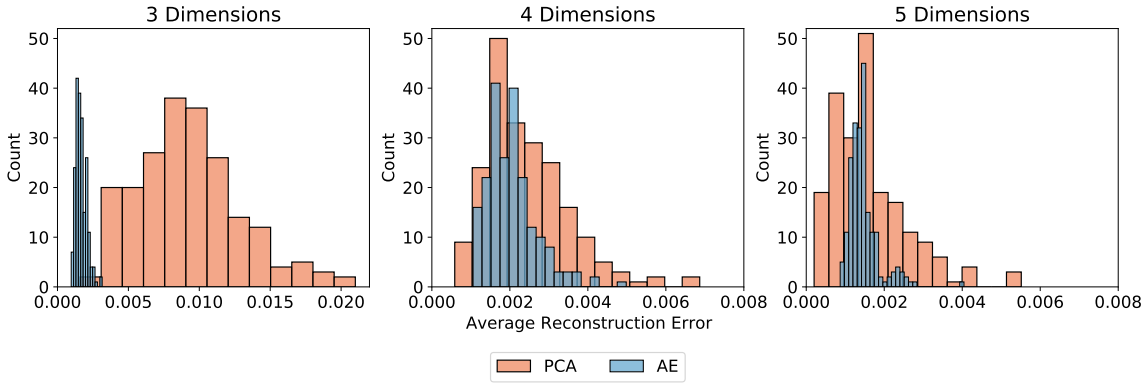

Figure 3.3: HISTOGRAM OF AVERAGE TESTING RECONSTRUCTION ERROR FOR VARIOUS LATENT SPACE SIZES USING OPTIMIZED DATA SET

3.3.2 Optimized Geometry vs Non-Optimized Geometry

We had originally hypothesized that the optimized airfoil coordinates would likely lie on a lower dimensional design manifold. Although this study did not confirm our hypothesis, it did show us some of the differences between these two datasets. For one, from Fig. 3.4 it is clear that the optimized airfoil data set is highly nonlinear. Testing MSE for PCA was substantially higher with this data set than that of the UIUC airfoil database. It took an additional two dimensions for the PCA models to reach the same accuracy as the autoencoder models for the optimized airfoil dataset. On the UIUC data set, autoencoder and PCA models performed similarly in lower dimensions.

The bootstrapped empirical 95% confidence interval over mean testing MSE is given by transparent shading for PCA models and darker shading for autoencoder models. This confidence interval is significantly wider for the optimized airfoil data set than the UIUC set. This implies that that both PCA and autoencoder models were struggling to model this data more so than the UIUC airfoil database. The optimized airfoil coordinates do not appear to lie on a lower dimensional design manifold. Of course, further studies are necessary to confirm this given that

not all modeling methods and activation functions were tested.

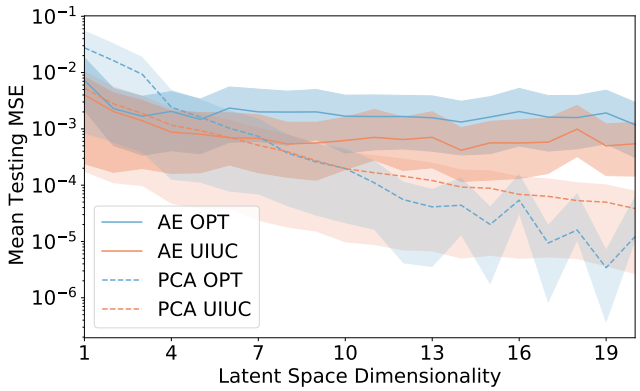

Figure 3.4: MEAN PCA AND AUTOENCODER TESTING MSE VALUES VS. DIMENSION ON OPTIMIZED AND UIUC AIRFOIL DATASETS

3.3.3 Including Parameters

The input parameters of Reynold's number, Mach number, angle of attack, and coefficient of lift were introduced as data set features. The optimized airfoil coordinates were generated using these conditions. Since the original features are dependent on these four inputs, we examine their effect on the latent space dimensionality in Fig. 3.5.

There are no significant differences in performance and dimensionality between the data sets. The greatest contrast is between the selected models. Once again, the autoencoders perform considerably better than PCA in lower dimensional latent spaces until five dimensions.

After adding the input parameters, the autoencoders took over four times as long to run as they did on the original optimized airfoil data set. This computational cost is great for insignificant performance benefits. Input parameters were found to be not only unnecessary for model fitting, but to hinder it. As discussed in Section 2.7, previous work has found larger feature sizes to take longer to run. We attribute the increased run time to increased feature size and to the

model selection. Extensions of this work using various autoencoder architectures and activation functions may yield different results.

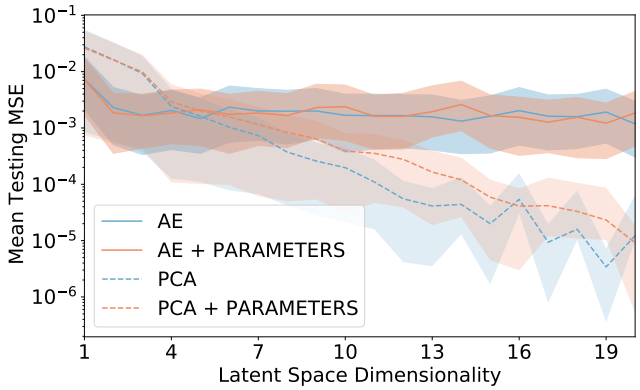

Figure 3.5: EFFECT OF INPUT PARAMETERS ON MEAN PCA AND AUTOENCODER TESTING MSE VALUES VS. DIMENSION

3.4 Discussion and Summary

This section first discusses how geometry optimality and performance parameters affect reconstruction and latent space dimensionality. We then digress to an overview of the limitations of this study such as autoencoder architecture, activation functions, and the issue of autoencoders getting stuck in local optima.

3.4.1 How Does Geometry Optimality Affect Latent Space Dimensionality?

Using various methods, the UIUC data set intrinsic dimensionality was found to be between 2 and 6 with a greater probability around 4 dimensions [43]. Using PCA and autoencoders, we found that these methods had similar performance around 4-6 dimensions for the UIUC and the optimized airfoil data set. While our data suggest that optimized geometry does not necessarily lie in a lower dimensional latent space than the original geometry, the model variety used was

not comprehensive. Alternative generative models such as variational autoencoders or generative adversarial networks may exhibit different behavior.

3.4.2 How Does Including Performance Parameters Affect Reconstruction?

We found that including input parameters as data set features from which the airfoil coordinates were generated does not necessarily decrease the latent space dimensionality. Since additional features only increase the computational cost, this finding suggests that there is no benefit to including additional features in the data set on which the rest of the airfoil geometry is dependent on. It also could be the case that the range of physical parameters over which the geometries were optimized, such as the range of Reynolds numbers, Mach numbers, angles of attack and lift coefficients, may not have been helpful to the dimensionality reduction. Extensions of this work into broad aerodynamic regimes or alternative flow models may alter our findings.

3.4.3 How Might the Autoencoder Architecture Alter Our Results?

Autoencoder architecture may additionally impact reprojection accuracy. For uniformity, all autoencoder models were given the same architecture in the results shown. A separate study was conducted to validate that this architecture was appropriate. Deeper neural networks from 2 to 10 hidden layers were additionally investigated to confirm that there was no significant performance increase for various architectures using the optimized airfoil data set. It was found that shallow networks were sufficient and that increasing the number of hidden layers actually results in worse performance for deep models. Moreover, as the network depth grew, so did the training time. For this reason and for uniformity, the original model with one hidden layer was

used for all experiments. Autoencoder architecture was also validated by repeating some of the trials after removing the nonlinear activation functions. Since the loss metric was already MSE, this newly-constructed autoencoder represents the same linear transformation as PCA, resulting in the same solution as in the PCA trial.

3.4.4 How Might the Activation Functions Alter Our Results?

The ReLU activation function was used throughout this study because it is fast, does not typically have the problem of vanishing gradients, and it generates sparse representations. We also assumed that the since PCA performs worse on the optimized airfoils than on the UIUC airfoils that the optimized data set is more nonlinear. While we did make this assumption, the curvature of the manifold was not thoroughly studied. We plan to investigate this phenomenon further and assign explicit manifold curvature metrics. Methodologies to better investigate the manifold curvature would likely need to rely on other, more differentiable activation functions.

Additional activation functions were used in combination with the architecture study. Sigmoid, Tanh, ELU, and CELU were tested in models ranging from 1-10 hidden layers. Sigmoid and Tanh were expected to have poor performance as they can often have vanishing gradients. This was observed when Sigmoid activation was used. On the other hand, Tanh, ELU, CELU, all performed reasonably similar to ReLU throughout all architecture sizes. Further study of these activation functions is necessary for future curvature investigations.

3.4.5 How Might Autoencoders Be Forced to Global Optima?

Throughout this study, one of the major issues observed was autoencoders losing their way with local optima and not converging to global minima. A method to solve the problem is to learn their optimal representations using gradient-based optimizers. This method was recently developed and applied to linear autoencoders [95]. Gradient-based approaches and regularization methods show promise in learning the optimal representation of principal components.

Chapter 4: Non-Linear Latent Spaces for Airfoil Design using Rotation Augmented Gradients

4.1 Overview

In this chapter, we introduce a method creating non-linear latent spaces with ordered principal directions via the use of autoencoders with Rotation Augmented Gradients. We investigate the ability of this single architecture, once trained, to provide similar reconstruction Mean Squared Error (MSE) to those of multiple individual autoencoders trained for separate latent space sizes. We then demonstrate that the proposed model stabilizes autoencoder training, particularly in autoencoders with extraneous nodes, as measured by a reduction in variance in the reconstruction MSE.

4.2 Theory

The Rotation Augmented Gradient method was shown to recover the correct representation more efficiently than gradient update rules such as nested dropout or non-uniform L_2 weight regularization [12]. It does this by applying a rotation to the encoder and decoder weight matrices, alongside the standard gradient descent update. The rotation and gradient descent update are separate terms, meaning that the model learns the rotation and PCA subspace independently. The

rotation augmentation term conserves reconstruction loss and its detachment from the gradient descent update increases training stability in linear models. Intuitively, learning the rotation and performing the gradient descent independently allows the gradient descent step to achieve faster convergence in the rotation direction, given the shallow gradients in the rotation direction once the autoencoder reaches a plateau of reasonable reconstruction MSE. This is only possible because the rotation augmentation term conserves reconstruction loss. The RAG algorithm additionally has strong symmetry-breaking properties that empirically expedite convergence over methods such as non-uniform weight regularization or deterministic nested dropout, which often suffer from ill-conditioning. For proofs and a more detailed theoretical analysis, we direct the reader to the original formulation of the RAG algorithm given by Bao et al. [12].

In comparison to the existing approaches, our work is most similar to RAG, with slight alterations to the algorithm to assess its performance on non-linear autoencoders. This is useful because linear models inherently struggle to model nonlinear and complex data. The primary advantage of using autoencoders over linear methods such as PCA, where the principal components can be directly established, stems from the use of nonlinear activation functions. By extending the RAG algorithm to nonlinear models, we show that this method of inducing latent space orderings has a wider range of applications.

In the original formulation, the encoder and decoder were each composed of one linear layer. We modified the architecture, adding a minimum of one nonlinear activation function and additional linear layer in both the encoder and decoder. With this new architecture, multiple weight matrices are created. The rotation was applied to the last weight matrix of the encoder and the first weight matrix of the decoder.

4.3 Autoencoder Hyperparameter Tuning

The hyperparameters used in this study included the learning rate, number of epochs, number of nodes in each layer, and the activation functions. The autoencoder architecture was additionally investigated to ensure our models were not overly simplified. All hyperparameter values could potentially impact the model accuracy, and thus are critical to determine.

For the synthetic problem, the AE hyperparameters were manually fine tuned such that the regular AE performed well. These parameters were then fixed those for the comparison between the naive AE and the RAG models. The simple encoder structure of input layer, ReLU activation, and bottleneck layer, was sufficient in modeling the synthetic data. The learning rate and number of epochs were set to 1.0×10^{-3} and 1×10^{4} respectively. RAG trained in 3 dimensions performed the best in one dimension when the output of the input layer and the input to the bottleneck layer were set to 1×10^{3} nodes. This is an artifact of reducing dimensionality by averaging out latent components which is discussed in more detail in Section 4.6.

Since our goal was to compare only the effects of the RAG modification, we used optimized hyperparameter tuning results from Chapter 3 on the airfoil databases. The original encoder was composed of an input layer, ReLU activation, and a bottleneck layer while the decoder mirrored this structure. We selected this existing design as a proof-of-concept that RAG could be extended to nonlinear models.

To guard against model overfitting, we applied L_2 regularization for numerous weight decay parameters ranging from 1×10^{-4} to 0.5. We found that increasing the weight decay also resulted in increased test MSE in all airfoil database trials. This suggests that minimal overfitting occurred. Slight overfitting of the naive AE was observed in the synthetic trial when 1000 nodes

were used. However, this overfitting showcases an advantage of using RAG. Where the naive AE overfit on this wide architecture, the RAG models were more robust.

4.4 Experiment 1: Comparison of Reconstruction Accuracy Between NAEs and RAG

This experiment answers the question: Does aligning the latent coordinates with key principal components result in a multi-fidelity model with similar performance to models trained on individual latent space sizes? We introduce an approach to ordering non-linear autoencoder latent spaces using the RAG algorithm. Furthermore, we show that this model provides similar reconstruction MSE to those of multiple traditional autoencoders, each trained on a separate latent space size.

4.4.1 Reconstruction Error Methodology

Our methodology is best explained by taking the initial toy problem as an example: First, we train one model with the rotation augmented gradient on a 3-dimensional latent space. Assuming the coordinates are indeed ordered, the first latent coordinate should account for the greatest amount of explained variance. The second coordinate should account for less and the third coordinate should account for even less. Having constructed this 3-dimensional ordered latent space, imagine we omit the last coordinate, filling it with some constant placeholder value such as the mean value of this component. This model would essentially become two-dimensional without needing to train a new model. We may simply take the newly formed coordinates and pass them through the decoder to project the results of the two-dimensional model back into the original

space. This process can then be iterated again to create a one-dimensional representation. We refer to this method of reducing dimensionality of the RAG model by averaging out the components with least information as latent averaging.

Using a vanilla autoencoder, we can then generate models the traditional way: constructing three separate models and comparing them to the multi-fidelity RAG results at each dimension. Since only one model needs to be created and trained, rather than individual autoencoder models for each latent dimension, we examine if the ordering of latent coordinates can be leveraged to reduce training time. We extend this idea to the UIUC airfoil database as well as to applied airfoil optimization examples. The latter optimization examples necessitate an expanded methodology which is detailed in 5.1.1.

4.4.2 Results

We first examine the synthetic data set and then proceed to the UIUC airfoil database example. Figure 4.1 depicts the MSE of three model types: vanilla autoencoders trained on each individual latent space size, RAG models trained on a three dimensional space and edited to reduce the latent dimensionality, and PCA. The solid lines depict the average MSE over 20 cross validation trials whereas the shaded areas display the bootstrapped empirical 95% confidence interval over mean testing MSE.

In our synthetic problem, the standard deviation of Gaussian noise added to the function was 0.02. For this reason, we expected the autoencoders to have MSE on the order of 4×10^{-4}. In Fig 4.1, we find that our intuitive estimation is well aligned with our results and the autoencoder performance hovers around this value. Since the synthetic data set was composed of a 1D data

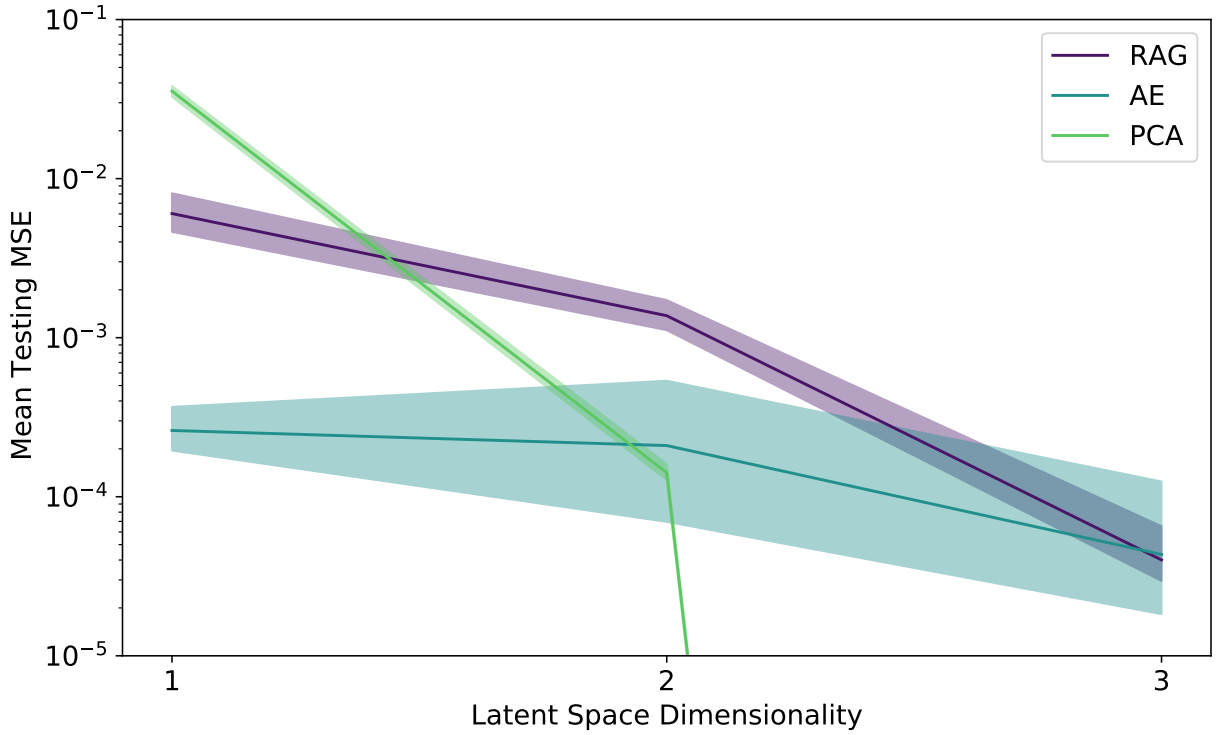

Figure 4.1: COMPARISON OF RECONSTRUCTION ERROR BETWEEN ONCE-TRAINED RAG AUTOENCODERS AND INDIVIDUALLY TRAINED AUTOENCODERS FOR EACH LATENT DIMENSION SIZE ON SYNTHETIC DATA SET

set in 3D space, the autoencoder with bottleneck layer of 1 dimension does a reasonable job approximating this, yielding on average a MSE of 2.61×10^{-4}. Increasing the dimension of this layer adds only slight improvements.

RAG models had marginally lower reconstruction error in 3 dimensions when compared to naive AE models. When the dimensionality is reduced further, the RAG performance drops. This was expected as information is lost in the latent editing process. The difference in MSE between RAG and naive autoencoders in 2 dimensions is approximately 1.16×10^{-3} and the difference in 1 dimension is 5.77×10^{-3}. Although the RAG models only needed to be trained once, the error between models is significant.

PCA performs the worst out of all models when trained on 1D. As displayed in Fig. 4.2, a

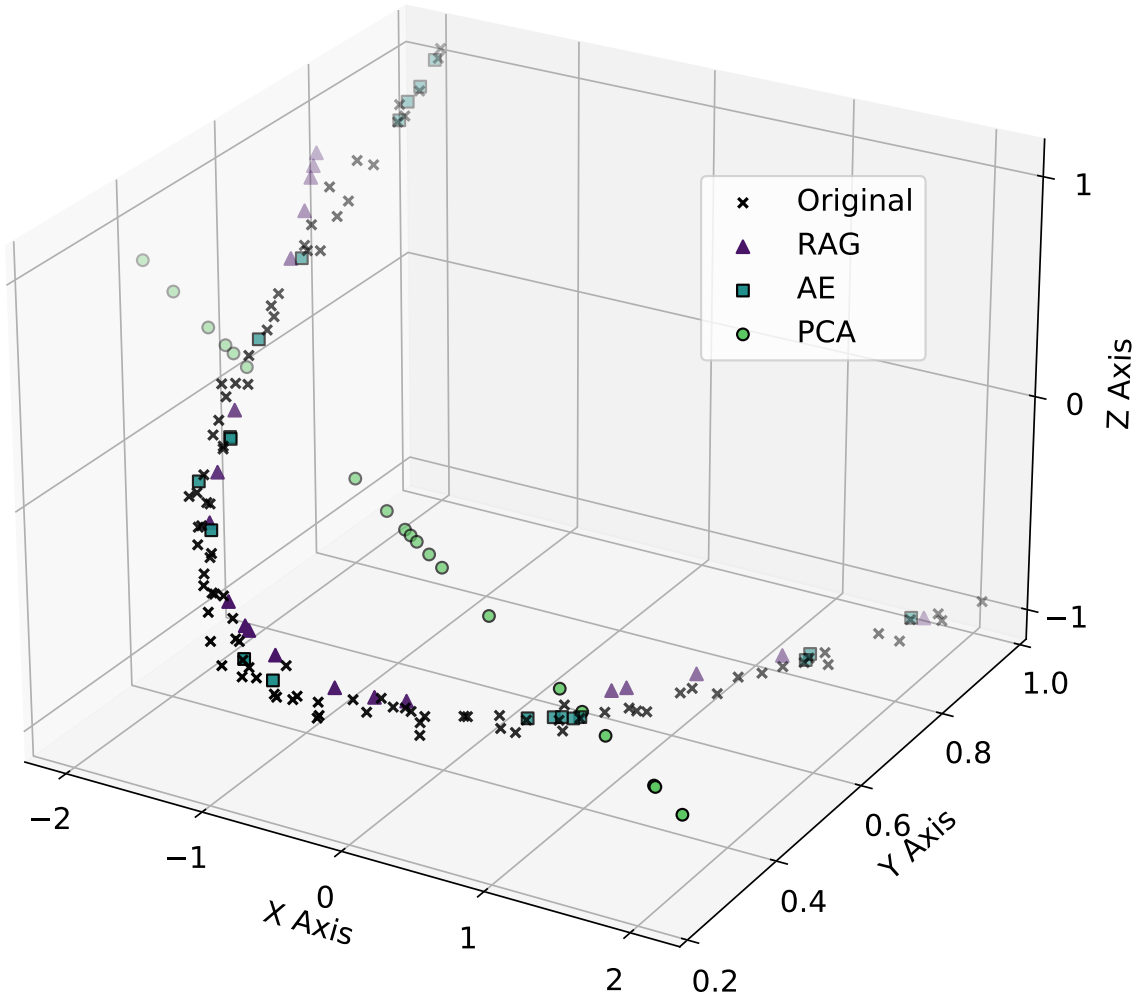

Figure 4.2: ORIGINAL SYNTHETIC COORDINATES PLOTTED ALONGSIDE 1D REPROJECTED COORDINATE PREDICTIONS USING RAG, AE, AND PCA

line is not an ideal approximation for this synthetic data, and in fact the RAG models on average outperform the PCA models by a factor of 5.88 in one dimension. However, PCA can theoretically reconstruct the model perfectly in 3 dimensions since the input space matches the output space. For this reason and because the synthetic data set was quite simple, PCA outperforms the autoencoder models in 3 dimensions. In practice, one would not use dimensionality reduction

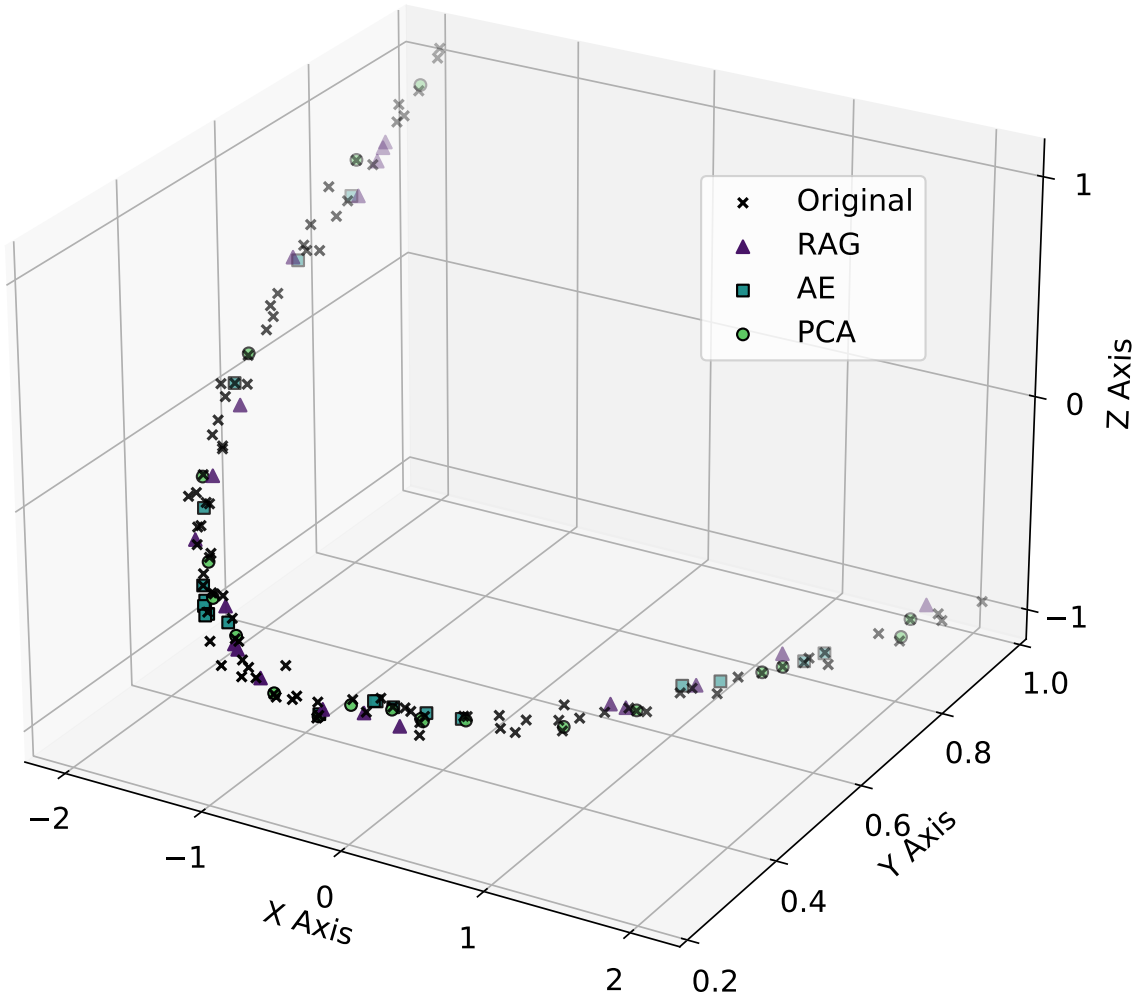

Figure 4.3: ORIGINAL SYNTHETIC COORDINATES PLOTTED ALONGSIDE 2D REPROJECTED COORDINATE PREDICTIONS USING RAG, AE, AND PCA

methods to maintain the original dimensional space, and thus we focus on analysis of the traditional autoencoders and the RAG models in lower dimensional space, using PCA merely as a reference point.

A typical example of the benefits of using RAG over PCA is visualized in Fig. 4.2 where the ability of RAG to model nonlinear functions is evident. Fig. 4.3 displays that models are

comparable in 2 dimensions, in part dues to a lack of 3D twist in the synthetic data set. This toy case serves as a proof-of-concept that the RAG algorithm works reasonably well in reducing training time while minimizing loss.

This same study was then conducted on the UIUC airfoil database as depicted in Fig. 4.4. We found that in comparison to the vanilla autoencoder trained on each individual dimension, the once-trained RAG model performs the similarly on the four dimensional space both models were trained on, and decreases in performance with latent space size. On average, the difference between RAG and naive autoencoder MSE was 1.78×10^{-4} in four dimensions, 3.22×10^{-3} in three dimensions, 1.10×10^{-2} in two and 9.61×10^{-3} in one dimension. These are substantial errors, and we find that the nonlinear RAG model cannot compete with individually trained AE models or PCA.

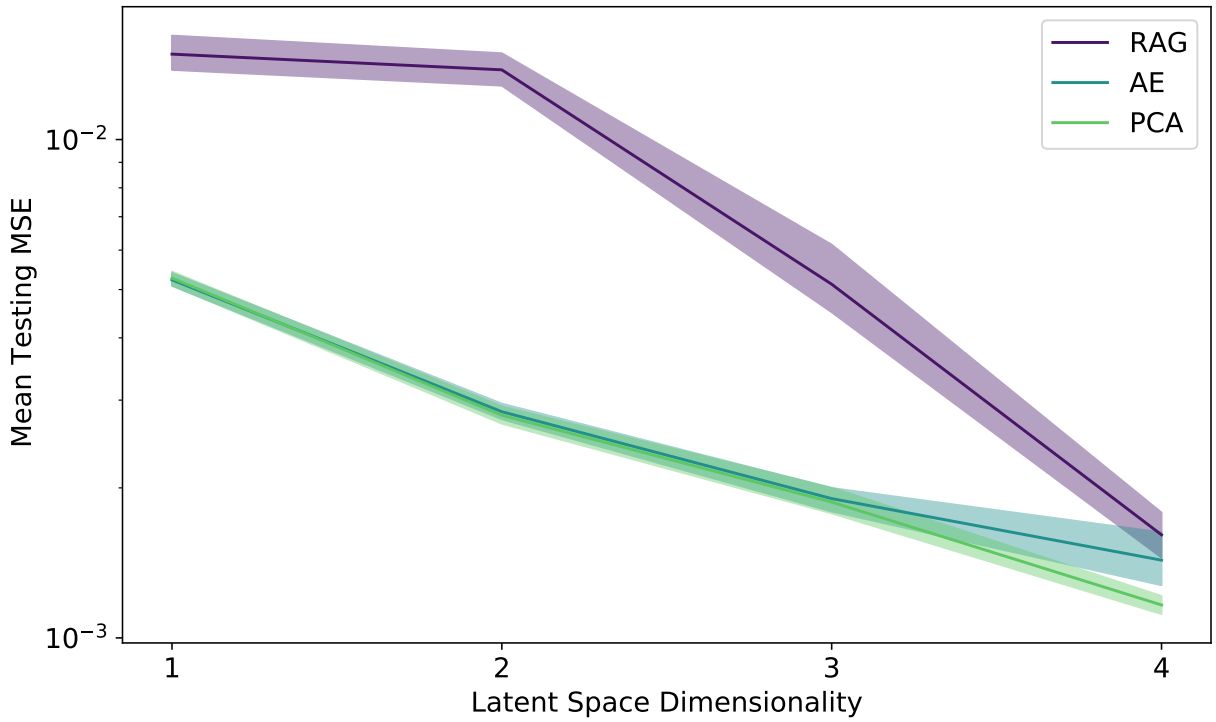

Figure 4.4: COMPARISON OF RECONSTRUCTION ERROR BETWEEN ONCE-TRAINED RAG AUTOENCODERS AND INDIVIDUALLY TRAINED AUTOENCODERS FOR EACH LATENT DIMENSION SIZE ON UIUC AIRFOIL DATABASE

4.5 Experiment 2: Effect of RAG on Fitting Stability of Non-Linear Autoencoders

We examine the training stability of non-linear autoencoders to answer the question: to what extent does aligning the latent coordinates with key principal components alleviate fitting stability? We demonstrate that the proposed model stabilizes autoencoder training, as measured by a reduction in variance in the reconstruction MSE.

4.5.1 Methodology

Training stability of traditional autoencoders and RAG autoencoders are examined by first, performing cross validation to estimate the metrics of MSE on they toy problem and CL/CD on the airfoil optimization problems in an unbiased manner. This was conducted by shuffling the data and taking 20 splits of 80% for training and 20% for testing. A model was trained on each split and variance in reconstruction MSE was then calculated across model for each method.

For each phase, we repeated these calculations to determine the average training stability of our models. We additionally implement a bootstrapped empirical 95% confidence interval over mean testing MSE as another metric of stability.

4.5.2 Results

The variance in each latent dimension representation on both the synthetic problem and on the UIUC airfoil database are given in Table 4.1 along with the bounds of the bootstrapped

		Variance	CI Bounds
3D Synth	AE	8.460e-9	[1.834e-5, 1.224e-4]
	RAG	1.256e-9	[2.941e-5, 6.551e-5]
2D Synth	AE	1.999e-7	[6.871e-5, 5.256e-4]
	RAG	4.504e-7	[1.118e-3, 1.721e-3]
1D Synth	AE	3.676e-8	[1.947e-4, 3.700e-4]
	RAG	1.447e-5	[4.665e-3, 8.162e-3]
4D UIUC	AE	1.550e-7	[1.274e-3, 1.620e-3]
	RAG	1.483e-7	[1.456e-3, 1.795e-3]
3D UIUC	AE	5.551e-8	[1.794e-3, 2.002e-3]
	RAG	3.121e-6	[4.491e-3, 6.078e-3]
2D UIUC	AE	5.402e-8	[2.747e-3, 2.951e-3]
	RAG	5.720e-6	[1.281e-2, 1.494e-2]
1D UIUC	AE	1.209e-7	[5.089e-3, 5.395e-3]
	RAG	7.363e-6	[1.375e-2, 1.619e-2]

Table 4.1: VARIANCE OF MEAN TESTING MSE AND BOUNDS OF BOOTSTRAPPED 95% CONFIDENCE INTERVAL BETWEEN ONCE-TRAINED RAG AND NAIVE AE

empirical 95% confidence intervals over mean testing MSE. In the synthetic problem where both models were trained on using equivalent bottleneck layers of three dimensions, the variance of the RAG models were on average over 5.84 times times smaller. On the airfoil database, the average variance of RAG models were found to be slightly smaller (4.32%), however they are close enough where this difference is likely statistically insignificant.

As the RAG models trained on 3 dimensional spaces in the synthetic problem and 4 dimensional spaces in the airfoil database were used to lower dimensional spaces, the variance increased. This was expected as a model trained in 1D will outperform a model trained in three dimensions on 1D representations. However, it is important to note that this increase in variance was often similar to that of the naive AE. While the variance did increase significantly in the 1D synthetic case, in the UIUC 1D case the RAG model's variance is less than 2 orders of magnitude greater than that of a naive AE. Even though the naive AE model was fit using 1D and the RAG model was fit using a 4D bottleneck layer, this performance demonstrates that the RAG algorithm

is leveraging ordering to create a stable fit without retraining.

The bounds of the bootstrapped empirical confidence interval over mean testing MSE on the synthetic and UIUC airfoil databases are also displayed in Table 4.1. The most important cases for comparing the stability of RAG and AE models are when both methods were trained on the same dimension. In the 3D synthetic case, the bounds of the naive AE confidence interval were 2.88 times wider than that of the RAG models. Conversely, the confidence interval of the RAG models were on average 1.02 times as wide in the 4D UIUC case. These results suggest that the RAG algorithm may increase training stability in simple cases. However, when the data becomes more complex, the rotation augmentation term may be have as much impact on model training stability. Further experimentation is necessary in order to validate this.

4.6 Discussion and Summary

4.6.1 Limitations

We found that the representations of AE and RAG were similar when trained and tested on the same dimension. The discrepancy between models stems from our approach of reducing dimensionality by setting the latent coordinates with least information to their average values. With this latent space editing, information captured in these latent coordinates is not only lost, but new information is created that is detrimental to the reconstruction. This method of reducing the dimensionality of the RAG model only worked well when the model had many nodes. Take the RAG models in Fig 4.1 and Fig 4.2 for instance. Although hyperparameter tunings showed a great fit to the original data points could be obtained with a RAG model with 10 nodes and a bottleneck layer of 1 dimension, using our scheme to reduce the dimensionality of a RAG

model trained in 3 dimensions to 1 dimension necessitated 1000 nodes. Even then, the end behavior of the RAG model predictions is visually imperfect as displayed Fig 4.2. Overall, the nonlinear version of the RAG model does not extend well to modeling lower dimensional spaces. Nevertheless, the RAG algorithm does appear to increase training stability and reduce overfitting in autoencoder models with many nodes.

The fundamental limitation is that while our method uses orderings to reduce loss in dimensionality reduction compared to traditional methods, this process is not lossless. Just as in PCA where each component constitutes a potion of the explained variance, each latent dimension in the RAG model contains some amount of information about the data. By averaging out a dimension to create a lower dimensional model, this information is effectively removed. The more dimensionality is reduced from the original model the greater this loss will become, regardless of the methodology behind reducing dimensionality from the original trained model.

4.6.2 How Can Models Be Fairly Compared?

For consistency in our experiments, we used identical architectures in both AE models, only changing whether the RAG terms were included or not. However, it is possible that the best architecture for a standard AE model might be different than best one for one that uses RAG. In this case, RAG models may be put at a disadvantage. We selected this comparison because it is a conservative estimate. The RAG model seems to require additional epochs when compared to a naive AE. We did not thoroughly investigate methods to accelerate RAG training because we wanted to keep the comparison between models straight-forward.

Chapter 5: In-Progress Ordered Components for Design Space Preconditioning

5.1 Overview

In this chapter we attempt to answer the question: to what extent does inducing an ordering on the latent dimensions improve the optimization convergence rates on various airfoil optimization problems? We evaluate the ability of RAG to pre-condition the design space for gradient-free optimization, as measured by accelerating Bayesian Optimization convergence on 2D RANS optimization. This chapter is preliminary, and there are several issues with the current implementation which are discussed in Section 5.2

5.1.1 Applied Optimization Methodology

We perform Bayesian Optimization on the airfoil representations learned in cases 2 and 3. This is done by first generating a tensor of random initial points that are then passed through the decoder used as latent coordinates to generate 2D airfoil coordinates as depicted in Fig 5.1. These airfoil designs are then run through a CFD simulation using the SU2 adjoint solver to generate CL/CD performance metrics. After the initial latent coordinates and their respective performances once reprojected back into the original design space are established, they are then fed into the Bayesian Optimize to generate new latent coordinate candidates as shown in Fig 5.2.

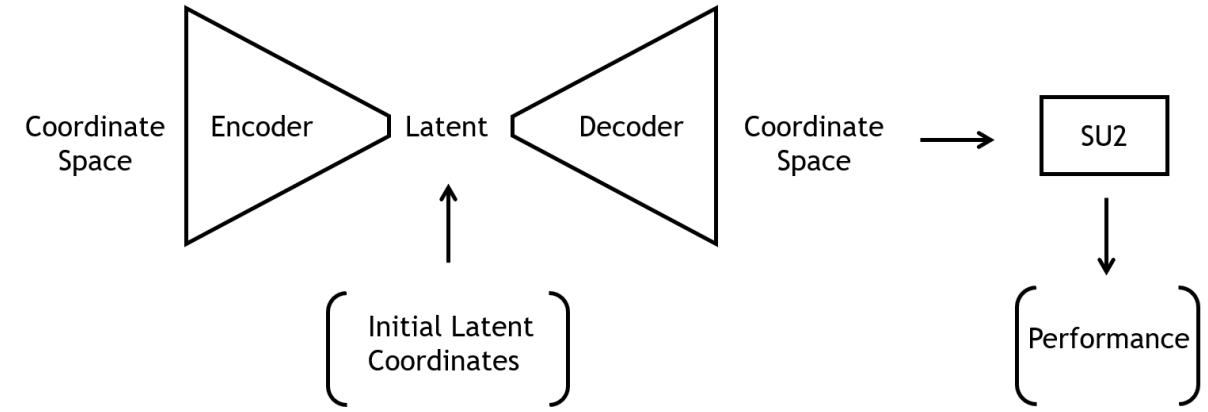

Figure 5.1: INITIALIZATION OF LATENT COORDINATES AND RESPECTIVE PERFORMANCES

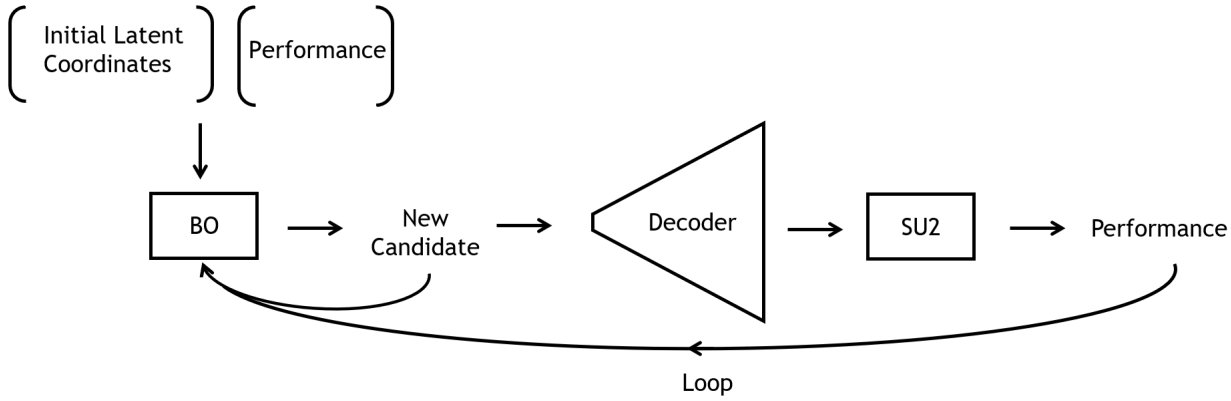

Figure 5.2: OPTIMIZATION WORKFLOW

Each new candidate, gets passed through the decoder and the SU2 solver to determine that candidates performance, and the cycle iterates until an optimal design is found. This process is also using a vanilla autoencoder and PCA for comparison of convergence speeds and optimal design performance.

The objective of our optimization is to maximize the lift-drag ratio under the following conditions: Reynold's number: 7×10^6, Mach number: 0.7, target lift coefficient: 0.35, and angle of attack: 0 degrees. These flow conditions were used in the midbench suite tutorial and

were anticipated to therefore be stable flow conditions for the inverse design. Our specific model uses the Botorch SingleTaskGP model, a single-task exact Gaussian process with inferred homoskedastic noise. The Matern kernel was used to compute the covariance matrix. Expected improvement was used as the acquisition function to evaluate the posterior without the need to perform Monte-Carlo sampling. Bounds were set by determining the maximum and minimum of each latent coordinate. A scalar of 0.001 was subtracted from the minimum values and added to the maximum values as a buffer to ensure that a larger area would be sampled in case the optimal solution lies just outside the range of explored coordinates. This is a relative and problem dependent scalar. Larger values expand the domain, and were typically found to generate less optimal candidates. However, this could be useful to explore if the data does not cover the design space well. This scalar is a hyperparameter that requires independent optimization and fine tuning. In our experiments on the UIUC airfoil database, values under 0.01 typically yielded the best results. Optimization using the RANS solver was computationally expensive, so fewer cross-validation trials were performed. In total, 5 trials were initialized using 20 randomly generated samples within the latent coordinate bounds of each model prior to 480 iterations of optimization.

5.1.2 Autoencoder Hyperparameter Tuning

We found the autoencoders used in Chapter 3 were insufficient in generating smooth airfoils and the performance of the Bayesian Optimizer was hindered. The addition of a hidden layer in the encoder and decoder was observed to ameliorate those optimization difficulties. The final autoencoder architecture for the airfoil optimizations was symmetric. The encoder consisted of three linear layers separated by ReLU activation functions. The first linear layer had 384

nodes, the same number as data set features. The second layer had 10 nodes, and the final layer was composed of 4 nodes to match the latent dimension. The learning rate was increased to a value of 1.0×10^{-3}, and the number of epochs to 15,000 as the increased model complexity necessitated additional training time. A further study was conducted testing the use of other activation functions including Tanh, Sinh, and CELU. We found no discernible benefit from using these activations over ReLU on the studied autoencoder architectures and elected to continue using ReLU.

5.1.3 Results

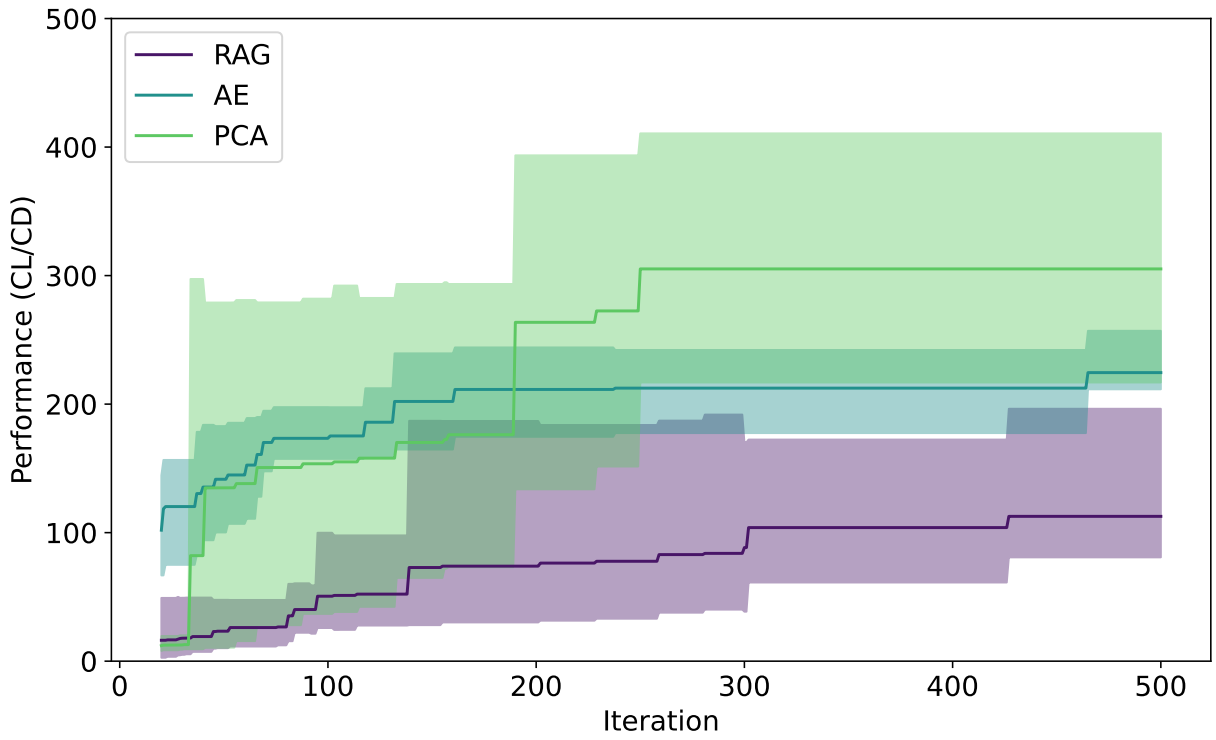

Figure 5.3: OPTIMIZATION HISTORY USING RAG AND NAIVE AE MODELS ON THE UIUC AIRFOIL DATABASE

First, we examine model performance the UIUC airfoil database and then proceed to the optimized airfoil database results. The number of optimization iterations is plotted against gen-

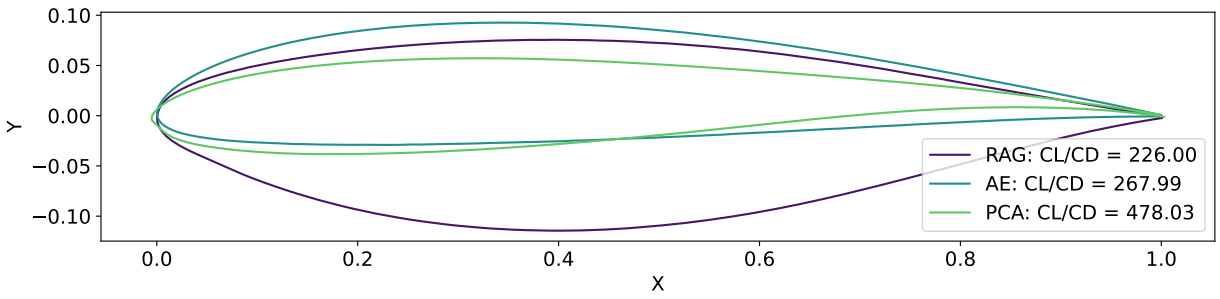

Figure 5.4: BEST PERFORMING BO-GENERATED AIRFOIL GEOMETRIES USING RAG AND NAIVE AE MODELS ON THE UIUC AIRFOIL DATABASE

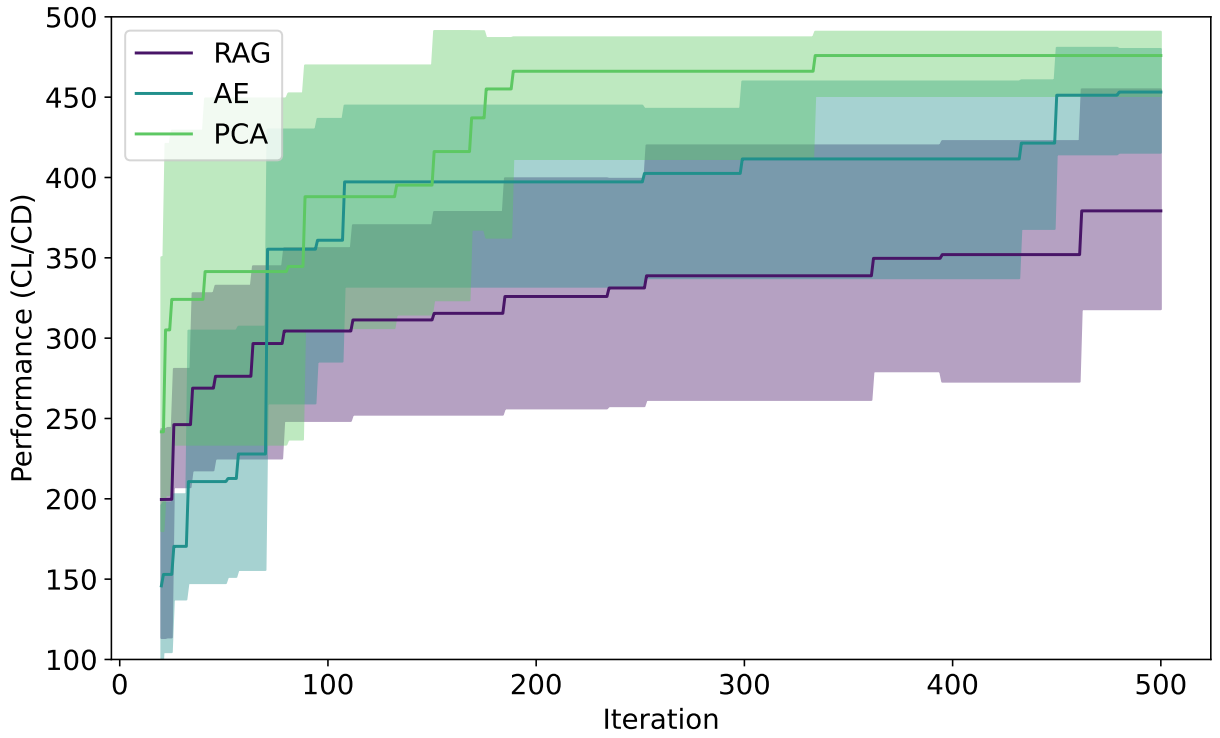

Figure 5.5: OPTIMIZATION HISTORY USING RAG AND NAIVE AE MODELS ON THE OPTIMIZED AIRFOIL DATABASE

erated airfoil performance in Fig. 5.3. The bootstrapped empirical 95% confidence intervals over mean testing MSE are wide, in part because the number of cross validation trials was limited to 5 models in each method. However, even with the limited trials, some clear trends appear.

The model that consistently generated the highest performing airfoils was PCA, followed by naive AEs and RAG models. It is also interesting to note that the AE models have a much

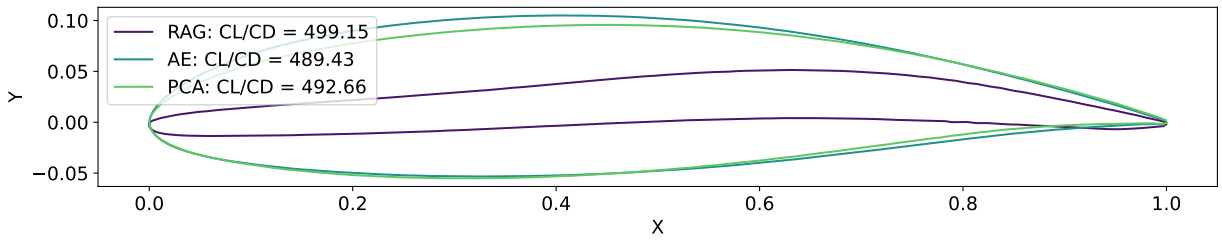

Figure 5.6: BEST PERFORMING BO-GENERATED AIRFOIL GEOMETRIES USING RAG AND NAIVE AE MODELS ON THE OPTIMIZED AIRFOIL DATABASE

larger performance after initialization than either models that induce latent orderings. Additionally, the confidence interval of the naive AE models are much tighter than that of either of order-inducing models.

The best performing airfoils generated using each method are overlaid in Fig. 5.4. All models developed smooth representations, without any specific constraints such as enforcing internal geometric layers [1]. The best PCA airfoil not only learned to be smooth, but also learned to decrease its width to reduce drag, and to develop a downward-pointing trailing edge to increase generated lift.

We then run the experiment on the SU2 optimized airfoil database as visualized in Fig. 5.5. Again, both PCA and the naive AE models had superior performance over the RAG models by the end of the optimization history. However, on this data set, RAG model performance was on average greater than the AE after initialization.The performance of these models are much greater than those trained on the UIUC airfoil database.

The best performing airfoils at the end of this optimization are displayed in Fig. 5.6. Although the airfoil generated using the naive AE looks similar to its UIUC database counterpart, the RAG algorithm took a vastly different approach, greatly reducing its width to lessen drag. The rotation term did effect the results considerably by narrowing the search to another part of

the manifold. While the RAG algorithm did provide a reasonable design representation for optimization, the models that excluded the RAG term outperformed it on average. We interpret this to mean that the RAG algorithm did not precondition the design space better and did not help the model avoid local minima.

5.2 Discussion and Summary

The results displayed in this section are unrealistic. Lift to drag ratios of nearly 500 are unreasonably high and should not be feasible. We believe that these performance values are erroneous, and are caused by failure of the solver to converge. We do not know if the observed trends will be the same once the solver is fixed and limit our discussion of the current results.

5.2.1 Limitations

The limitations of this are primarily due to problems using the solver and suboptimal use of the case 3 database.

5.2.1.1 Solver Difficulties

Our experiments were computationally expensive. The largest component of this expense was the CFD required to evaluate the performance of the airfoil geometry. To bound this, SU2 adjoint solver was limited to 1250 iterations. This stopping criteria resulted in some airfoils not reaching the convergence criteria. In such cases, the resulting lift to drag ratio may not be accurate. This was especially an issue when the optimized data set was used. On this data set, the final results required filtering to eliminate these case which generated extremely unrealistic

results (CL/CD $>$ 500). Even so, some of the performances may be artificially inflated if they did not fully converge in these 1250 iterations. Raising this limit would yield more reliable results but would greatly add to the computational cost of experimentation. It is also possible that some airfoils still not converge even if this cut-off is raised. To address this, we are currently investigating either fixing the tolerance of convergence in the existing inverse design pipeline, or pivoting to using a computationally cheaper, but less accurate solver such as XFOIL for which previous work has developed a workflow for [1]. It is essential to remedy this before the results are considered.

Additionally, there were iterations in which optimization of the acquisition function of the naive AE yielded an ill-conditioned, non positive definite matrix. In these cases the candidate selection process was altered to create a random replacement tensor of equivalent size and within the bounds.

Finally, even though each latent coordinates are constrained within a range depending on the model, many coordinate combinations yield shapes with intersecting geometries, ultimately resulting in meshing failures. To avoid this we implemented a system to check if the geometry had any intersections prior to meshing. If intersections were detected, the optimization was skipped and the performance was set to a value of 0. Unfortunately, this raised a slight issue. If in the initialization process all geometries have intersections, the Bayesian optimizer is not given much useful information on where to sample to increase performance. Methods that directly model feasible versus in-feasible regions, rather than trying to factor this into the BO objective, may ameliorate such problems [96, 97].

5.2.1.2 Optimized Data Set

To originally create the optimized data set, Bézier-GAN used the input parameters of Reynolds number, Mach number, and target lift coefficient. The output of this model was both optimized airfoil coordinates and optimized angles of attack [48]. Since the models in this section used only the coordinates, and not the angles of attack they were optimized for, the airfoils were not actually optimized for the flow they were evaluated on. Since many of the airfoils in this data set were similar but had different angles of attack, the shape variability was also low. To fix this problem, there are two potential workflows. The first solution is to assign the angle of attack as an additional feature vector. From the results in Chapter 2.11, the addition of a feature vector by itself is unlikely to significantly impact the results. In the case that it does not, weights could be applied to this parameter such that the model focuses more on correctly determining the angle of attack, as getting the wrong value of this parameter would result in a greater loss than any of the other parameters. This weight would need to be optimized. The second possible approach would be to factor the angle of attack into the coordinate values. In this case the feature vectors could be kept the same, and the coordinates would change. Both options are currently under investigation.

5.2.2 How Well Does the RAG Algorithm Extend to More Complex Autoencoder Structures?

We ran our airfoil optimization workflow on autoencoder architectures with one and with two hidden layers. If our results hold, we found that increasing the number of layers and nonlinear activation functions did not have much effect on the RAG algorithm's ability to precondition the

design space. Evaluating deep architectures was deemed outside the scope of our investigation.

5.2.3 In What Situations Is the RAG Algorithm Useful?

We expect that RAG may be more valuable in specific design situations when it is known that shallow gradients are causing gradient descent to converge slowly in AE training. In such cases, the additional rotation term may speed up convergence and has been proven to have local linear convergence to the optimal representation in linear autoencoders [12]. In the case of the two airfoil databases we tested so far, this rotation term did not meaningfully accelerate convergence.

Chapter 6: Conclusion

In this thesis we first investigated three characteristics of airfoil manifolds. The impact of latent dimension size and sample size on the optimized airfoil data set reconstruction error was explored. We found that autoencoders perform better in low dimensional spaces but PCA is more stable with extremely low or high fractions of training samples. Next, we investigated the intrinsic dimensionality between the UIUC airfoil database and a subset of these airfoils optimized using CBGAN and CEBGAN models. From this study we determined that optimized geometries do not necessarily live in lower dimensional manifolds. Finally, we investigated the effect of including target optimization conditions as data set features on reconstruction error over a swath of latent space sizes. We found that adding features that all data is dependent on does not necessarily reduce data set dimensionality. We discovered throughout these trials, autoencoders would often lose their way, getting trapped within local optima rather then converging to global minima.

Our work then pivoted to examining a recently proposed method to solve the problem by learning optimal representations using rotation augmented gradient autoencoders [95]. We extended this algorithm to nonlinear autoencoders to increase its utility on complex and nonlinear data sets. Since this algorithm induces latent component orderings, we investigate a method of leveraging this ordering to create approximations in lower dimensions without the need to retrain the model. We find our simple approach is functional but requires wide autoencoder

models to construct the best low dimensional representations, which still have significant error. We then examine the training stability of nonlinear RAG autoencoders, finding that it is largely data-dependent and is most beneficial in simple cases. Finally, we study the ability of nonlinear RAG autoencoders to precondition the design space. Our preliminary results show that RAG latent ordering does not guarantee faster convergence and increased performance in gradient-free Bayesian Optimizations; however, there are still fundamental issues in the current implementation.

Airfoil manifolds remain a complex area of study with numerous factors impacting latent space dimension and reprojection accuracy. Our approach provides several experiments to help understand the implicit effect of data and model selection on the latent space. Beyond that, our developments have explored potential methods to reduce model training time, the number of models needed to be trained, and number of iterations required in airfoil optimization. Although many of these investigations did not result in improvements to the current state of the art, they had potential to do so and merited further analysis. If our work extends, we have learned some generalities about the nonlinear RAG method of ordered dimensionality reduction, and have determined that using the naive approach of removing the last latent component is often insufficient in creating a robust through several dimensional spaces. We have additionally established that the RAG algorithm impacts the gradient-free optimization results, and drives the optimal airfoil search to another part of the design manifold. With our contributions, we hope to shed some light on various characteristics of airfoil manifolds and ordered nonlinear models, and to lay groundwork for future studies in this field.

6.1 Future Work

6.1.1 Optimization Implementation Issues

The primary area of future work is to fix the implementation problems associated with evaluating the ability of RAG to precondition the design space for gradient-free optimization. As discussed in Section 5.2, the principal issues were poor solver convergence and the fact that our current workflow used strictly the optimized airfoil coordinates and not the angles of attack they were optimized for. Action is currently being taken on both of these items. We are investigating: a.) fixing the current convergence criteria, b.) using another solver, c.) using the weighted angle of attack as a feature vector, and d.) using the angle of attack to calculate new coordinate values. Once rectified, it may also be prudent to examine the performance of the original RAG algorithm in our inverse-design workflow to establish a comparison to our nonlinear rendition.

6.1.2 Vary Latent Space Size

After fixing the optimization problems, a useful test would be to repeat these experiments with various latent space sizes. As discussed in Chapter 3, this latent dimensionality greatly impacted model performances. All experiments on airfoil databases in Chapter 5 were conducted using a latent size of four dimensions as this had previously been determined to be about the intrinsic dimensionality of the data set [43]. As this latent dimension decreases, the performance of PCA should degrade faster than that of the autoencoders. Although our results show that PCA is sufficient to model these datasets in 4 dimensions, a further reduction would likely require more complex models and would showcase the strengths of RAG and naive AE models.

6.1.3 Examine Other Methods of Reducing Dimension of a Trained Model

Another area of future work is to investigate other alternatives for reducing dimensionality of a trained and ordered autoencoder model. While our approach of averaging out the component that accounts for the least information did work, it was a lossy process that only outperformed PCA in wide autoencoders. Although some amount of information will always be lost, further research in this area could decrease such loss.

6.1.4 Decrease Number of Feature Vectors

Additionally, autoencoders require a large number of samples to generate the best and most useful results. In our study, the UIUC airfoil data set had 1042 samples and 384 feature vectors, which corresponds to approximately 2.71 times as many samples as feature vectors. While we have demonstrated that this can still result in a usable representation for optimization, it would be beneficial to investigate this as it could potentially effect the model utility. To do this there are two potential avenues: increasing samples or decreasing feature vectors. Increasing samples would be an arduous process, but decreasing the number of feature vectors appears to have merit. Previous work has explored the possibility of representing airfoils as functions, rather than as coordinates. This could greatly cut down on the number of feature vectors [98]. Other work used 20 design variables to generate airfoil splines [99]. By cutting down the number of feature vectors in these manners, the performance of both naive and RAG AE models in learning a lower dimensional representation would likely increase, provided the geometry remains unaltered.

Bibliography

[1] Wei Chen, Kevin Chiu, and Mark D Fuge. Airfoil design parameterization and optimization using bézier generative adversarial networks. *AIAA journal*, 58(11):4723–4735, 2020.

[2] Garret N Vanderplaats. Approximation Concepts for Numerical Airfoil Optimization. Technical report, NASA Technical Paper 1370, 03 1979.

[3] Asha Viswanath, Alexander IJ Forrester, and AJ Keane. Dimension reduction for aerodynamic design optimization. *AIAA journal*, 49(6):1256–1266, 2011.

[4] QIU Yasong, BAI Junqiang, LIU Nan, and WANG Chen. Global aerodynamic design optimization based on data dimensionality reduction. *Chinese Journal of Aeronautics*, 31(4):643–659, 2018.

[5] Rafael Gómez-Bombarelli, David Duvenaud, José Miguel Hernández-Lobato, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *CoRR*, abs/1610.02415, 2016.

[6] Juan Antonio Delgado-Guerrero, Adrià Colomé, and Carme Torras. Sample-efficient robot motion learning using gaussian process latent variable models. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 314–320. IEEE, 2020.

[7] Jeffrey C Regier and Philip B Stark. Mini-minimax uncertainty quantification for emulators. *SIAM/ASA Journal on Uncertainty Quantification*, 3(1):686–708, 2015.

[8] RICHARD Bellman. Dynamic programming. *Press Princeton, New Jersey*, 1957.

[9] Johan Larsson and Qiqi Wang. The prospect of using large eddy and detached eddy simulations in engineering design, and the research required to get there. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 372(2022):20130329, 2014.

[10] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *arXiv preprint arXiv:2003.05991*, 2020.

[11] Quentin Fournier and Daniel Aloise. Empirical comparison between autoencoders and traditional dimensionality reduction methods. In *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pages 211–214. IEEE, 2019.

[12] Xuchan Bao, James Lucas, Sushant Sachdeva, and Roger B Grosse. Regularized linear autoencoders recover the principal components, eventually. *Advances in Neural Information Processing Systems*, 33:6971–6981, 2020.

[13] Quoc V Le et al. A tutorial on deep learning part 2: Autoencoders, convolutional neural networks and recurrent neural networks. *Google Brain*, 20:1–20, 2015.

[14] Yonghyeon Lee, Hyeokjun Kwon, and Frank Park. Neighborhood reconstructing autoencoders. *Advances in Neural Information Processing Systems*, 34:536–546, 2021.

[15] Anush Sankaran, Mayank Vatsa, Richa Singh, and Angshul Majumdar. Group sparse autoencoder. *Image and Vision Computing*, 60:64–74, 2017.

[16] Ali Ghodsi. Dimensionality reduction a short tutorial. *Department of Statistics and Actuarial Science, Univ. of Waterloo, Ontario, Canada*, 37(38):2006, 2006.

[17] Alan Julian Izenman. Introduction to manifold learning. *Wiley Interdisciplinary Reviews: Computational Statistics*, 4(5):439–446, 2012.

[18] Laurens Van Der Maaten, Eric Postma, Jaap Van den Herik, et al. Dimensionality reduction: A comparative review. *J Mach Learn Res*, 10(66-71):13, 2009.

[19] Jonathon Shlens. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*, 2014.

[20] Steven M Holland. Principal components analysis (pca). *Department of Geology, University of Georgia, Athens, GA*, pages 30602–2501, 2008.

[21] Saïd Ladjal, Alasdair Newson, and Chi-Hieu Pham. A pca-like autoencoder. *arXiv preprint arXiv:1904.01277*, 2019.

[22] Roberto Fernández, Pierre-Yves Louis, and Francesca R Nardi. Overview: Pca models and issues. *Probabilistic cellular automata*, pages 1–30, 2018.

[23] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[24] Rasmus Bro and Age K Smilde. Principal component analysis. *Analytical methods*, 6(9):2812–2831, 2014.

[25] Lindsay I Smith. A tutorial on principal components analysis. 2002.

[26] Christopher Bishop. Bayesian pca. *Advances in neural information processing systems*, 11, 1998.

[27] Chunming Li, Yanhua Diao, Hongtao Ma, and Yushan Li. A statistical pca method for face recognition. In *2008 Second international symposium on intelligent information technology application*, volume 3, pages 376–380. IEEE, 2008.

[28] Jean Hélder Marques Ribeiro and William Roberto Wolf. Identification of coherent structures in the flow past a naca0012 airfoil via proper orthogonal decomposition. *Physics of Fluids*, 29(8):085104, 2017.

[29] Patrick LeGresley and Juan Alonso. Airfoil design optimization using reduced order models based on proper orthogonal decomposition. In *Fluids 2000 conference and exhibit*, page 2545, 2000.

[30] Tan Bui-Thanh, Murali Damodaran, and Karen Willcox. Aerodynamic data reconstruction and inverse design using proper orthogonal decomposition. *AIAA journal*, 42(8):1505–1516, 2004.

[31] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[32] Bharath Sriperumbudur and Nicholas Sterge. Approximate kernel pca using random features: Computational vs. statistical trade-off. *arXiv preprint arXiv:1706.06296*, 2017.

[33] Karl Ezra Pilario, Mahmood Shafiee, Yi Cao, Liyun Lao, and Shuang-Hua Yang. A review of kernel methods for feature extraction in nonlinear process monitoring. *Processes*, 8(1):24, 2019.

[34] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[35] Yasi Wang, Hongxun Yao, and Sicheng Zhao. Auto-encoder based dimensionality reduction. *Neurocomputing*, 184:232–242, 2016.

[36] Wei Wang, Yan Huang, Yizhou Wang, and Liang Wang. Generalized autoencoder: A neural network framework for dimensionality reduction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2014.

[37] Elad Plaut. From principal subspaces to principal components with linear autoencoders. *arXiv preprint arXiv:1804.10253*, 2018.

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

[39] Danny Liu and Lei Tang. Proper orthogonal decomposition (pod)/response surface methodology (rsm) methodology for low reynolds number aerodynamics on micro aerial vehicle (mav). Technical report, ZONA TECHNOLOGY INC SCOTTSDALE AZ, 2006.

[40] Noriyasu Omata and Susumu Shirayama. A novel method of low-dimensional representation for temporal behavior of flow fields using deep autoencoder. *Aip Advances*, 9(1):015006, 2019.

[41] Kazuo Yonekura, Kazunari Wada, and Katsuyuki Suzuki. Generating various airfoil shapes with required lift coefficient using conditional variational autoencoders. *arXiv preprint arXiv:2106.09901*, 2021.

[42] Michael S Selig. Uiuc airfoil data site. 1996.

[43] Wei Chen, Mark Fuge, and Noa Chazan. Design manifolds capture the intrinsic complexity and dimension of design spaces. *Journal of Mechanical Design*, 139(5):051102–051102–10, 2017.

[44] Qiuyi Chen, Phillip Pope, and Mark Fuge. Learning airfoil manifolds with optimal transport. In *AIAA SCITECH 2022 Forum*, page 2352, 2022.

[45] Alec Van Slooten and Mark Fuge. Effect of optimal geometries and performance parameters on airfoil latent space dimension. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 86229, page V03AT03A005. American Society of Mechanical Engineers, 2022.

[46] Amit Kumar and Nagabhushana Rao Vadlamani. Inverse design of airfoils using convolutional neural network and deep neural network. In *Gas Turbine India Conference*, volume 85536, page V001T02A001. American Society of Mechanical Engineers, 2021.

[47] Emre Yilmaz and Brian German. Conditional generative adversarial network framework for airfoil inverse design. In *AIAA aviation 2020 forum*, page 3185, 2020.

[48] Qiuyi Chen, Jun Wang, Phillip Pope, Wei Chen, and Mark Fuge. Inverse design of two-dimensional airfoils using conditional generative models and surrogate log-likelihoods. *Journal of Mechanical Design*, 144(2):021712, 2021.

[49] Nutan Chen, Maximilian Karl, and Patrick Van Der Smagt. Dynamic movement primitives in latent space of time-dependent variational autoencoders. In *2016 IEEE-RAS 16th international conference on humanoid robots (Humanoids)*, pages 629–636. IEEE, 2016.

[50] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[51] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.

[52] Jasem Almotiri, Khaled Elleithy, and Abdelrahman Elleithy. Comparison of autoencoder and principal component analysis followed by neural network for e-learning using handwritten recognition. In *2017 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, pages 1–5. IEEE, 2017.

[53] Phillip Pope, Chen Zhu, Ahmed Abdelkader, Micah Goldblum, and Tom Goldstein. The intrinsic dimension of images and its impact on learning. *arXiv preprint arXiv:2104.08894*, 2021.

[54] Elizaveta Levina and Peter J Bickel. Maximum likelihood estimation of intrinsic dimension. In *Advances in neural information processing systems*, pages 777–784, 2005.

[55] Thiago Rios, Bernhard Sendhoff, Stefan Menzel, Thomas Bäck, and Bas van Stein. On the efficiency of a point cloud autoencoder as a geometric representation for shape optimization. In *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 791–798. IEEE, 2019.

[56] Thiago Rios, Bas van Stein, Patricia Wollstadt, Thomas Bäck, Bernhard Sendhoff, and Stefan Menzel. Exploiting local geometric features in vehicle design optimization with 3d point cloud autoencoders. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, pages 514–521. IEEE, 2021.

[57] Dunja Mladenić. Feature selection for dimensionality reduction. In *International Statistical and Optimization Perspectives Workshop" Subspace, Latent Structure and Feature Selection"*, pages 84–102. Springer, 2005.

[58] Xudong Jiang. Linear subspace learning-based dimensionality reduction. *IEEE Signal Processing Magazine*, 28(2):16–26, 2011.

[59] R. Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks*, 5(4):537–550, 1994.

[60] Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In *International Conference on Machine Learning*, pages 2649–2658. PMLR, 2018.

[61] Wei Chen and Mark Fuge. Synthesizing designs with inter-part dependencies using hierarchical generative adversarial networks. *Journal of Mechanical Design*, 141(11):111403, 2019.

[62] Paul G Constantine. *Active subspaces: Emerging ideas for dimension reduction in parameter studies*. SIAM, 2015.

[63] Paul G Constantine, Eric Dow, and Qiqi Wang. Active subspace methods in theory and practice: applications to kriging surfaces. *SIAM Journal on Scientific Computing*, 36(4):A1500–A1524, 2014.

[64] Trent W Lukaczyk, Paul Constantine, Francisco Palacios, and Juan J Alonso. Active subspaces for shape optimization. In *10th AIAA multidisciplinary design optimization conference*, page 1171, 2014.

[65] Chunfeng Cui, Kaiqi Zhang, Talgat Daulbaev, Julia Gusak, Ivan Oseledets, and Zheng Zhang. Active subspace of neural networks: Structural analysis and universal attacks. *SIAM Journal on Mathematics of Data Science*, 2(4):1096–1122, 2020.

[66] Jichao Li, Jinsheng Cai, and Kun Qu. Surrogate-based aerodynamic shape optimization with the active subspace method. *Structural and Multidisciplinary Optimization*, 59(2):403–419, 2019.

[67] Danial Khatamsaz, Abhilash Molkeri, Richard Couperthwaite, Jaylen James, Raymundo Arróyave, Ankit Srivastava, and Douglas Allaire. Adaptive active subspace-based efficient multifidelity materials design. *Materials & Design*, 209:110001, 2021.

[68] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[69] Oren Rippel, Michael Gelbart, and Ryan Adams. Learning ordered representations with nested dropout. In *International Conference on Machine Learning*, pages 1746–1754. PMLR, 2014.

[70] Artur Bekasov and Iain Murray. Ordering dimensions with nested dropout normalizing flows. *arXiv preprint arXiv:2006.08777*, 2020.

[71] Chelsea Finn, Lisa Anne Hendricks, and Trevor Darrell. Learning compact convolutional neural networks with nested dropout. *arXiv preprint arXiv:1412.7155*, 2014.

[72] Paul Bertens. Rank ordered autoencoders. *arXiv preprint arXiv:1605.01749*, 2016.

[73] Daniel Kunin, Jonathan Bloom, Aleksandrina Goeva, and Cotton Seed. Loss landscapes of regularized linear autoencoders. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3560–3569. PMLR, 09–15 Jun 2019.

[74] Chi-Hieu Pham, Saïd Ladjal, and Alasdair Newson. Pca-ae: Principal component analysis autoencoder for organising the latent space of generative networks. *Journal of Mathematical Imaging and Vision*, pages 1–17, 2022.

[75] Terence D Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural networks*, 2(6):459–473, 1989.

[76] Karthik Mani, Brian A Lockwood, and Dimitri J Mavriplis. Adjoint-based unsteady airfoil design optimization with application to dynamic stall. In *Annual Forum Proceedings-AHS International*, volume 3, pages 1940–1952, 2012.

[77] Nicolas R Gauger, Andrea Walther, Carsten Moldenhauer, and Markus Widhalm. Automatic differentiation of an entire design chain for aerodynamic shape optimization. In *New Results in Numerical and Experimental Fluid Mechanics VI*, pages 454–461. Springer, 2007.

[78] Joshua Hodson, Douglas F Hunsaker, and Robert Spall. Wing optimization using dual number automatic differentiation in machup. In *55th AIAA Aerospace Sciences Meeting*, page 0033, 2017.

[79] Siva Nadarajah and Antony Jameson. A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization. In *38th Aerospace Sciences Meeting and Exhibit*, page 667, 2000.

[80] Apoorv Agnihotri and Nipun Batra. Exploring bayesian optimization. *Distill*, 5(5):e26, 2020.

[81] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[82] Riccardo Moriconi, Marc Peter Deisenroth, and KS Sesh Kumar. High-dimensional bayesian optimization using low-dimensional feature spaces. *Machine Learning*, 109(9):1925–1943, 2020.

[83] Jonathan Wenger, Geoff Pleiss, Philipp Hennig, John Cunningham, and Jacob Gardner. Preconditioning for scalable gaussian process hyperparameter optimization. In *International Conference on Machine Learning*, pages 23751–23780. PMLR, 2022.

[84] Jan J Gerbrands. On the relationships between svd, klt and pca. *Pattern recognition*, 14(1-6):375–381, 1981.

[85] Dheeraj Bokde, Sheetal Girase, and Debajyoti Mukhopadhyay. Matrix factorization model in collaborative filtering algorithms: A survey. *Procedia Computer Science*, 49:136–146, 2015.

[86] Wei Sun. *Stability of machine learning algorithms*. PhD thesis, Purdue University, 2015.

[87] Richard O Duda, Peter E Hart, and David G Stork. Pattern classification 2nd ed. *John Willey & Sons Inc*, 2001.

[88] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[89] Daniel Berrar. Cross-validation., 2019.

[90] Larry Wasserman. *All of statistics: a concise course in statistical inference*, volume 26. Springer, 2004.

[91] Sharath M Shankaranarayana and Davor Runje. Alime: Autoencoder based approach for local interpretability. In *International conference on intelligent data engineering and automated learning*, pages 454–463. Springer, 2019.

[92] Masataro Asai and Hiroshi Kajino. Towards stable symbol grounding with zero-suppressed state autoencoder. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, pages 592–600, 2019.

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

[94] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An

imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[95] Xuchan Bao, James Lucas, Sushant Sachdeva, and Roger B Grosse. Regularized linear autoencoders recover the principal components, eventually. *Advances in Neural Information Processing Systems*, 33:6971–6981, 2020.

[96] Wei Chen and Mark Fuge. Beyond the known: Detecting novel feasible domains over an unbounded design space. *Journal of Mechanical Design*, 139(11), 2017.

[97] Wei Chen and Mark Fuge. Active expansion sampling for learning feasible domains in an unbounded input space. *Structural and Multidisciplinary Optimization*, 57:925–945, 2018.

[98] Wang Xudong, Wang Licun, and Xia Hongjun. An integrated method for designing airfoils shapes. *Mathematical Problems in Engineering*, 2015, 2015.

[99] Brian R Jones, William A Crossley, and Anastasios S Lyrintzis. Aerodynamic and aeroacoustic optimization of rotorcraft airfoils via a parallel genetic algorithm. *Journal of aircraft*, 37(6):1088–1096, 2000.