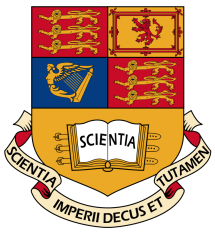

Imperial College London

Imperial College London

Faculty of Engineering

Department of Earth Science and Engineering

Unsupervised Machine Learning: An Application in Seismic Signal Analysis

Author:
Hugo Benjamin Coussens

Supervisors:
Lukas Mosser
Professor Olivier Dubrule

A thesis submitted for the degree of

MSc Applied Computational Science and Engineering

August, 2019

Contents

Code metadata description	Specification
Technical platform of implementation	PyPi and Jupyter
Permanent link to code/repository used for this code version	`https://github.com/msc-acse/acse-9-independent-research-project-coush001`
Compilation requirements and dependencies	PyTorch>=1.0, Jupyter
Programming Languages	Python3
Current code version	v1.2
Link to developer documentation	`https://github.com/msc-acse/acse-9-independent-research-project-coush001/blob/master/README.md`

Abstract

A Python package (SeismicReduction) has been developed to enable efficient exploratory analysis with dimensionality reduction on seismic datasets. The package was co-developed alongside an accompanying graphical user interface (GUI). The software supports SEG-Y formatted seismic datasets. Seismic amplitude traces serve as an input to an array of unsupervised machine learning algorithms. PCA, UMAP, VAE and β-VAE have been investigated for optimal configuration to perform dimensionality reduction and cluster analysis on an example seismic dataset. The package and GUI are designed to be accessible for geophysicists and geologists regardless of Python programming fluency. The aim is to provide automation to the workflow enabling quick exploratory investigations in this new tranche of seismic analysis.

keywords: **Seismic, Unsupervised Learning, AVO Analysis, VAE, β-VAE, UMAP**

1 Introduction

1.1 Motivation

Recent advances in unsupervised machine learning algorithms provide powerful tools for the representation of high dimensional data in a low dimensional space. Initial studies into the application of these algorithms on seismic data provide promising results Mosser et al. (2019); Mosser (2018); Desai (2019). The algorithms have exhibited an ability to learn and represent meaningful geophysical features. Unlike conventional techniques the models learn parameters from the full seismic trace, revealing possibly previously un-exposed insights. The study and utilisation of dimensionality reduction in this domain is still in its infancy. Investigations, software tools and use cases are limited. To further understand the utility and transferability of this analysis, it is vital to have a highly accessible and optimised tool to deliver an array of unsupervised dimensionality reduction investigations.

1.2 Background

1.2.1 Machine Learning in Geoscience

An understanding of the physical structure and composition of our planet, and other planets, is of extreme importance for many reasons. These range from the discovery of essential natural resources to the prediction of natural hazards. Study by direct observation is occasionally possible via borehole and well measurements. However these are only rarely financially feasible, and consistently unobtainable below certain depths. Therefore occurrence of these direct observations are sparse and only locally defined. It is a key challenge to callibrate these direct observations with the various sources of indirect observations. Indirect observations are gained via surface measurements obtained by an array of different geophysical surveying techniques. Unfortunately this data is at best a noisy non-linear representation of material properties. Models are therefore generated from ill-posed, poorly conditioned inversion systems (Backus and Gilbert (1970); Mora (1986)) . Despite the advancement

of precision and quality in recording equipment and performance of inversion algorithms, an exact solution to this problem is not possible. Instead we must look to develop technologies that continue the incremental progress to higher accuracy and reliability of models. The geosciences community (both scientific and commercial) has been slow to get on board the machine learning revolution, Karpatne et al. (2019). However, the review by Bergen et al. (2019) highlights the positive outlook for solid earth geosciences to undergo a machine learning transformation. The prospect is particularly exciting due to the existence of large and diverse datasets associated with geophysical and geological investigations. These range from signal based seismic surveys to satellite imagery and 3D laser scanning (Lidar). Bergen et al. (2019) outlines three particular categories in which machine learning research in geosciences will be focussed:

1. **Automation in the execution of complex tasks, which are unable to be defined by a set of explicit commands.** These tasks may aim to make predictions, or automate steps in data pre-processing and data analysis workflows. In either case they may add value in situations where the task is too complex or time consuming for humans, or machine learning exceeds human performance. Relevant examples include use of support vector regression for seismic data interpolation of missing traces, and supervised learning for lithofacies recognition using support vector regression, Hall (2016).
2. **Modelling or representation of characteristic features from datasets.** In supervised scenarios the algorithms aim to reproduce the fundamental physics of systems. The aim is often to find the underlying physical model or image from the observed data. In the un-supervised case, the objective may be simple clustering of similar data points, or additionally the extraction of physical attributes into a lower-dimensional representation. For example: McCann et al. (2017) reviews the utilisation of convolutional neural networks for inverse problems in imaging. The advantages in domain transferability and potential speed-ups over traditional methods are encouraging. Mosser et al. (2018) present a reduction in the computational expense of seismic inversion via the combination of a Generative Adversarial Network (GAN) and numerical solution of seismic inversion using the adjoint method.
3. **New discoveries of features, patterns or properties un-achievable via conventional methods.** It is argued that the functional space provided by machine learning, and deep learning in particular, provides new areas for discovery previously unobtainable by conventional methods. For example analysis of fault friction from seismic signals presented by Rouet-Leduc et al. (2018).

1.2.2 Analysis of Seismic Data

Intepretations on seismic data have traditionally been dominated by qualitative visual interpretation, via the observation of reflector shape and position. The task of a geologist being to identify key features such as sedimentary sequences, horizon events or fault planes. This is done via visual identification of characteristic structures and features observed in reflection events known as 'horizons'. These may be linked back to direct observation from surface outcrops or petrophysical well measurements. Important to note this is a subjective process, and in the absence of direct observation, is very prone to error and uncertainty.

The advent of improved processing routines and higher resolution in seismic data have facilitated quantitative interpretation techniques to be implemented with higher confidence. These techniques arise from the sensitivity of seismic velocities to key petrophysical parameters, such as pore-fluid presence, porosity and pore pressure.

A particular example of quantitative interpretation is amplitude versus offset analysis (AVO). AVO analysis looks at the relationship between amplitude and source-receiver offset, Avseth et al. (2005). This is underpinned by the variance of the acoustic reflection coefficient with angle of incidence. The AVO technique enables geophysicists to determine pore-fluid presence, porosity and seismic velocity. Unlike classical 'bright-spot' interpretations (Hammond (1974), AVO has the ability to distinguish lithology variations from fluid changes, Chiburis et al. (1993).

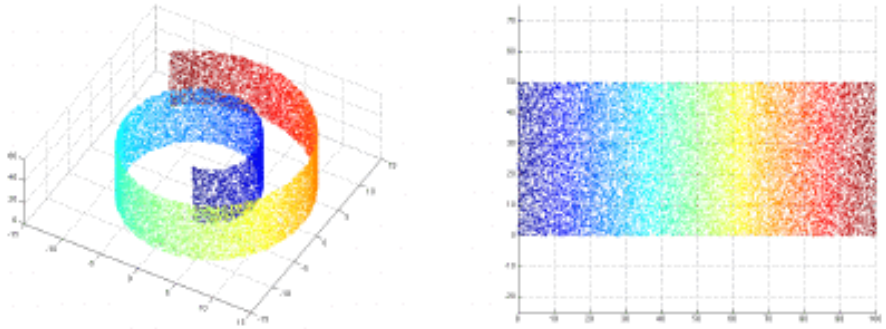

Figure 1: The Swiss Roll dataset.
On the left: the data is presented in its original form. On the right is a two-dimensional embedding of the data performed by the non-linear ISOMAP algorithm.

1.2.3 Key Advances in Unsupervised Learning

Unsupervised learning algorithms and techniques are the subset of machine learning tasked to deal with unlabelled datasets, Sathya and Abraham (2013). In absence of data labels, or target variables, the task is to draw insight from the intrinsic and hidden structure in the data. There are a handful of different genres of algorithms which tackle unsupervised learning via different approaches. The two covered here are auto-encoding based neural networks and manifold approximation algorithms. Broadly, the desired outcome of unsupervised learning is either clustering, data compression, or extraction of meaningful features. Data in the real world often possesses high dimensionality, for example; there are many pixels within a single image; many discrete values to a signal recording and many words to a magazine article. In order to identify meaningful patterns and similarities within datasets it may be useful to reduce data dimensionality. Doing so will facilitate visualisation and clustering of the high-dimensional data. Initially dimensionality reduction was confined to linear techniques such as Principal Component Analysis (PCA) or Linear Discriminant Analysis (Wold et al. (1987); Ye et al. (2005)). The obvious limitations of these linear techniques is the inability to project on non-linear manifolds. There have been great advances in the performance of non-linear techniques recently. The Swiss roll data-set is an excellent illustrative example for the benefit of non-linear techniques. Figure 1 presents the non-linear isometric mapping (ISOMAP) study by Burgoyne and McAdams (2007). The points in this example lie on a three-dimensional manifold representing a swiss roll. Linear methods like PCA are unable to represent the full structure of the manifold and effectively squash the layers of the Swiss roll together. In contrast non-linear methods have been shown to find an embedding that maintains structure, in effect un-rolling the Swiss roll onto a two-dimensional surface.

Variational Auto-Encoders (VAE)
To understand VAEs, Kingma and Welling (2013), it is helpful to first understand the structure and usage of autoencoders, Hinton and Salakhutdinov (2006). They are comprised of artificial neural networks (ANN's) which work in two parts: An encoder network reduces inputs into meaningful representations at a low dimensional 'bottleneck', referred to as the latent space. A decoder network aims to reconstruct the input back out from the latent representations. The goal is for the network to learn patterns or features from the input data that are summarised by the latent space variables. The network is usually trained on a simple cross-entropy loss function which penalises the network for inaccurate reconstruction from input to output. VAEs follow a very similar structure to autoencoders, but with different capabilities. VAEs can act as generative models, whereby learnt latent space distributions of the data can be sampled stochastically in order to generate brand new synthetic outputs. The key difference in a VAE is the presence of the stochastic latent space. VAE latent space representations are penalized for divergence from a Gaussian prior. The objective function of a VAE

is known as the evidence lower-bound (ELBO). ELBO is comprised of two constituent terms. The first: reconstruction likelihood quantifies the models ability to create an accurate reconstruction of the input through the information bottleneck. The second: KL-divergence aims to spread the range of latent vectors to match a Gaussian prior. A modified version of the VAE, known as β-VAE, aims to learn a latent representation in which the parameters of the latent vector have been observed to 'disentangle' features from the input, Burgess et al. (2018); Higgins et al. (2017). The β-VAE makes a modification to the ELBO by multiplying the KL-divergence term by a factor, β.

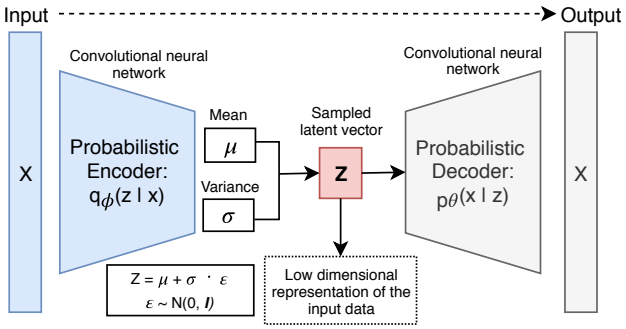

Figure 2: VAE architecture diagram. The reparameterisation trick to sample the latent space for model training is specified in the equations in the bottoom left corner. Z represents the latent vactor that we extract for visualisation of data clustering.

Manifold Embedding/Neighbour Mapping

An alternative approach to dimensionality reduction comes in the form of neighbour maps or manifold embedding algorithms. These algorithms are particularly useful in the visualisation of high-dimensional datasets, however the latent variables may not represent direct feature extraction. Significant progress has been made recently in this field with t-distributed stochastic neighbour embedding (t-SNE) by van der Maaten and Geoffrey E. (2008) and with uniform manifold approximation and projection (UMAP) by McInnes et al. (2018). The t-SNE algorithm finds a low dimensional embedding that minimises the KL-divergance between two distributions, the distribution that measures pairwise similarity in the input data and the distribution which does the same in the low-dimensional embedding.

UMAP brings a similar approach to t-SNE but with a few extra advantages. Both algorithms are efficient in successfully preserving local structural relationships in data. However, UMAP has been shown to better preserve the global structure of the data set. UMAP is able to transform data into a representation of any chosen dimension, where t-SNE is computationally limited to 2 or 3. UMAP has also been shown to operate with a superior run-time performance to t-SNE and other similar algorithms. A common camplaint with t-SNE is that it is non-parametric, and therefore is unable to add new data points to an existing embedding. It is true that t-SNE does not learn a function which projects individual samples from ambient space to embedding space. However the objective function is well defined and new samples can be added to an existing embedding by the optimisation of a new data-point position with respect to the existing embedding. The openTSNE implementation, Poličar et al. (2019), presents the only algorithm capable of this functionality. In contrast UMAP inherently learns the mapping $f(x) : \mathbb{R}^n \rightarrow \mathbb{R}^2$, allowing new data to be added to existing embeddings without the need to re-run the algorithm at all.

1.2.4 Application of Unsupervised Learning to Seismic Data

The investigations by Mosser et al. (2019) apply unsupervised learning techniques to a seismic data set for visualisation in 2-D. Calibration with AVO anomalies reveal successful clustering of traces with low fluid-factor values.

The investigation uses the following workflow:

1. Data Processing:
    - Horizon Flattening
    - Data Normalisation
2. Calculation of fluid factor for every amplitude sample using an AVO anomaly approximation
3. Dimensionality reduction applied using UMAP to 2-D
4. Training and running VAE, reducing to latent space of 8-dimensions, visualised in 2d by the UMAP algorithm
5. Both are visualised with AVO fluid factors overlain as attributes.

Desai (2019) provides an investigation into the robustness and sensitivity of a VAE model trained on seismic data. In this study geophysically motivated variations are applied to seismic trace inputs with the response in the latent space observed. Variations to the input data were amplitude, time shifting, Gaussian noise and far offset amplitude increase. Results show the responses of latent variables correlate well with the geophysically motivated modifications. Indicating that the VAE model is extracting certain physical properties into the latent space. The possible uses/interpretations that can be drawn from the studies of unsupervised learning applied to seismic are as follows:

- VAE's can be used as a tool for the de-noising of seismic data.
- VAE's can be used as generative models to create synthetic seismic data.
- Data clustering calibrates well with AVO anomaly values.
- Latent variables plotted in physical space, highlight model ability to cluster spatial relationship of data.
- Dimensionality reduction can be used as an anomaly detection tool for seismic traces.

1.3 Project Objectives

The principal project objective was to design and develop a tool or a set of tools to perform the full dimensionality reduction workflow on seismic datasets. The key focus was on high usability and accessibility of the tools. The tool automates the previously studied workflows for UMAP and VAE. SeismicReduction additionally incorporates PCA and β-VAE into the array of models.

There is currently little experience about the use of these algorithms on seismic data. It is reasonable to assume there is significant potential in the possible applications, however far more research is required before we can build confidence in specific use cases. The scope of this project is not to deliver production ready software delivering a specific use case, it is to provide a tool that delivers a comprehensive yet efficient exploration of the array of models and data processing techniques. It is the aim that this tool provides a platform that inspires further interest and research into specific use cases, model development and investigation into the feature extraction exhibited by the models.

Investigations to optimise the models on seismic data were carried out. Optimal model configurations have been documented, these account for hyper-parameters such as the VAE neural network architecture, optimisers, learning rates and required training epochs.

1.4 Software Requirements

The key requirements for the software are defined in Table 1. These were defined as the minimum functionality to be delivered in the minimum viable product.

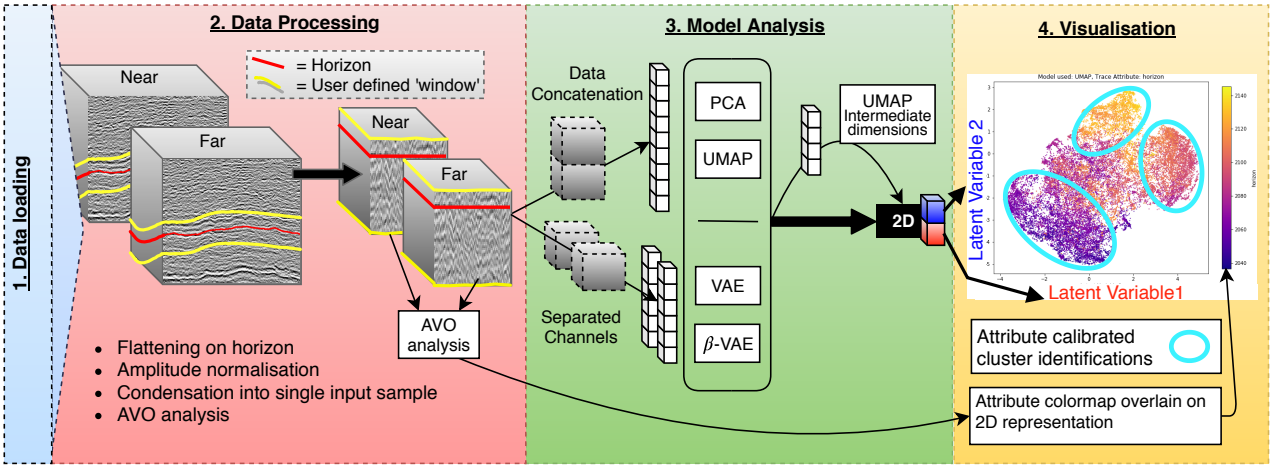

Figure 3: Full analysis workflow from data loading to two dimensional model output. Grey cubes represent the whole seismic cube. White cubes represent the number of dimensions to each individual sample of the data at each stage.

Requirement
Encapsulate following features into single Python package API:
- Load SEG-Y format seismic data into Python environment
- Run data processing routines with user specified paramaters
- Run PCA, UMAP, VAE, and β-VAE with user specified parameters
- Save and load trained neural network models
- Plot live ELBO values from VAE training
- Visualise model outputs with trace attributes as the colour map
- Save model visualisation plots to file
Present full software functionality in a graphical user interface

Table 1: Requirements

2 Analysis Methodology

Before describing the software structure and implementation it first important to describe the details and rational behind the methodology of each stage of the analysis workflow. The full workflow is summarised in Figure 3. Each main component will be described in further detail in the following sections.

2.1 Data Source

The workflow requires provision of two 3D seismic data cubes, one for near-offset amplitudes and one for far-offset amplitudes. It is expected that both cubes represent the exact same subsurface locality. In order to perform horizon flattening (explanation in Section 2.2) a horizon depth interpretation dataset is necessary. The specific technical requirements for the format of these files is covered in the code documentation. The loading of these datasets into the Python environment is performed by SegyPy, Hansen (2018).

2.2 Data Processing

Data processing covers the flow of data from raw seismic cubes to individual model inputs. The key focus in this context is the method in which we engineer individual data samples from the two

different 3D seismic cubes. It is important this process is explained with verbosity as the origination of model inputs is vital for defining the scope and context of what our model outputs mean.

A seismic cube can be understood as the low frequency - filtered representation of acoustic impedance contrasts between different subsurface layers. A cube is simply a three dimensional array of amplitudes, however it is important to keep in mind the genesis of this data. A cube is the composition of many individual vertical components: seismic traces. A single seismic trace is a recording of signal amplitudes, representative of subsurface reflection layers, in time. The amplitudes at greater time correlate to greater depth.

A key consideration is the distribution of the primary target/variable that we are investigating. In this context, this is hydrocarbon reservoir presence: approximated via the AVO anomaly indicator. In supervised machine learning it is preferred to train models on a dataset with even distribution of data classes. In this context we can think of data samples as falling in the crude classification of hydrocarbon presence or absence within the sample. However as we are dealing with unsupervised learning there is no explicit data classification. Nonetheless it is still desirable to attempt to engineer the data in the most efficient way to create an even distribution. The data partitioning/sampling method chosen was vertical array sampling. Each data sample is akin to a single trace (or some amalgamation from near and far offset traces) and represents the vertical extent of subsurface information for a given locality in map view. This intuitively means data points are also representative spacial locations. Model analysis variables can be retracted back to the spacial domain and leveraged for further interpretations. Importantly this scheme of data sampling supports the most even distribution of data sampling the target reservoir.

At this stage we implement the ability to choose from two different methods for conditioning the length of seismic trace. Both methods involve a form of 'cropping' of the vertical extent of the data. In 'horizon flattening' data is extracted from a set window above and below the horizon position. In 'plain cropping' data is extracted from a set window across the whole dataset, regardless of horizon depth, this could be utilised where there is no available horizon data, or can be applied to an already flattened dataset. The reason behind horizon flattening or cropping the data is for the concentration of data around a key target region/horizon.

The last consideration is the engineering of a single input from two separate seismic cubes. This was dealt with in two alternative ways specific to individual models. The PCA and UMAP models are only compatible with single channel data. In this case near and far offset traces are concatenated into a single channel vector input. For the VAE and β-VAE models a single input is constituted by two independent channels. This process is highlighted visually by Figure 4.

2.3 Trace Attributes

For further interpretation of outputs it is important to have a set of 'per-sample' attributes. These attributes are represented in the colour maps in model output visualisation. This allows for clear identification of data clustering of each attribute. Dependent on the attribute this could indicate that the model has 'learnt' to represent the features within a low dimensional space.

AVO anomaly

In standard AVO analysis the full common mid-point (CMP) gather is utilised. The amplitude vs $\sin^2(offset)$ plot returns a linear relationship. The gradient and intercept of this relationship contains the information of the AVO anomaly. In a cross plot of gradients vs intercepts we expect the samples from hydrocarbon bearing interval boundaries to exhibit a larger offset from the background trend. This phenomena is highlighted in Figure 5 and 6, taken from Russell (2014).

However in our workflow we only have access to two different offsets for each trace. Instead we use a simplified approximation to the AVO anomaly. This is done in the following way: Firstly calculating the near-far amplitude difference for each sample. We then find a linear regression co-efficient to near-far difference against the near offset amplitude. Finally fluid factor for each sample is calculated by near-far difference minus the regression line. This calculation is summarised by Equations 1 and 2. Negative fluid factor to correspond to increased likelihood of hydrocarbon presence. This is an

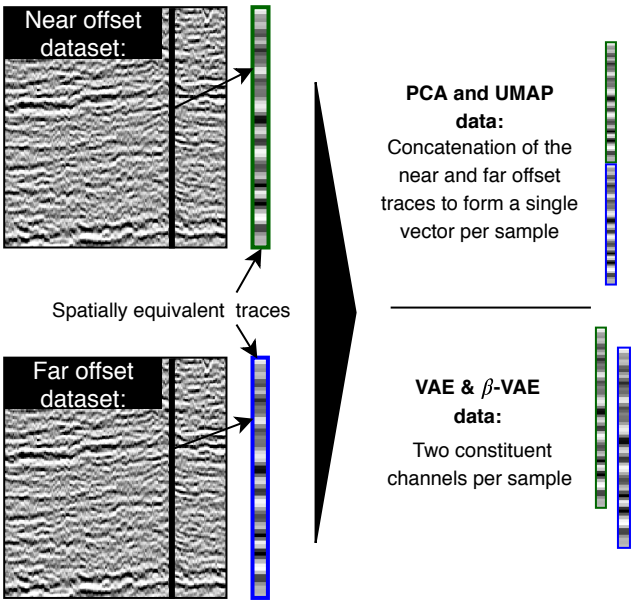

Figure 4: Visual presentation of flow of data from seismic cube to an individual sample of data via two different methods, dependant on model destination.

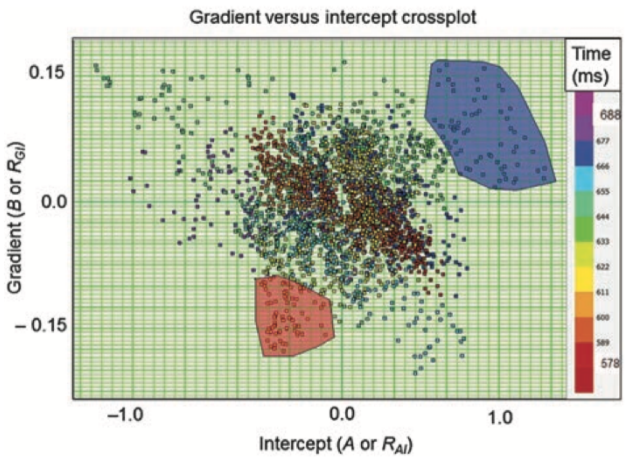

Figure 5: AVO intercept versus gradient crossplot taken from a 100-ms window around the target horizon between CMPs 300 and 360. The regions captured by the red and blue polygonal zones represent large AVO anomalies and are are shown in Figure 6

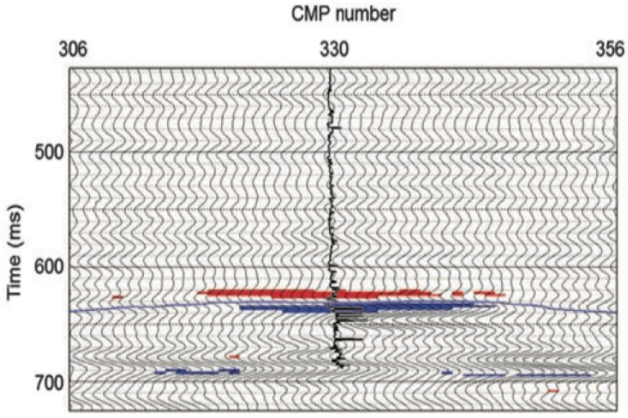

Figure 6: Red and blue polygonal zones from Figure 5, where the red zone shows the top of the gas sand and the blue zone shows the base of the gas sand

approximation only and allows for relative interpretation of fluid factors.

$$y_{avo} = \alpha * x_{avo} \tag{1}$$

$$f_{fluid} = y_{avo} - \alpha * x_{avo} \tag{2}$$

x_{avo} - near offset sample amplitude
y_{avo} - difference between the far offset sample amplitude and the near offset amplitude.
α - linear regression co-efficient
f_{fluid} - fluid factor attribute (AVO anomaly)

Horizon depth
After interpolation of the horizon interpretation data we have a value for the horizon position (in time) for every seismic trace. This is useful to see if models have organised/represented data in the latent space dependent on the horizon 'depth' of each trace.

2.4 Model Analysis

Principal Component Analysis:

- The PCA algorithm was implemented with the scikit-learn 'decomposition' library.
- The algorithm allows for any dimensional input to be transformed into any number of principal components akin to output dimension.

Uniform Manifold Approximation and Projection:

- The UMAP algorithm is implemented from the umap-learn package, McInnes et al. (2018)
- 'n_neighbours' parameter defines how UMAP balances local versus global structure. This parameter defines the local neighbourhood UMAP uses when learning the manifold of the data.
- 'min_dist' parameter controls how tightly UMAP is able to pack points together. It refers to the minimum distance permitted between points in the low dimensional representation.

Variational Auto Encoder:

- The VAE model is implemented via use of the PyTorch package, Paszke et al. (2017).
- The architecture is based upon the initial works by Mosser (2018) visualised in Figure 7.
- The VAE is modified to be able to handle an adaptable size of input dimension based on the processed data input. Due to intricacies arising with convolutional layers the input dimension must be a factor of 4.
- The models are able to be adapted with an arbitrary latent dimension, and therefore output dimension.
- There is user control for the amount of epochs and learning rate to be run in model training
- Training/validation dataset initialisation, model training and model running are all automated.

Beta Variational Auto-Encoder:

- The β-VAE model is exactly the same as the VAE except for the added β parameter as discussed previously.

All models are capable of reducing dimension from any input dimension to any user specified size. However the software only facilitates visualisation of two dimensional outputs. Therefore we employ use of an intermediary UMAP step to condense any oversized outputs into two dimensions. For example we may want to run an analysis where we use a VAE for $\mathbb{R}^n \rightarrow \mathbb{R}^8$, we then would use UMAP for $\mathbb{R}^8 \rightarrow \mathbb{R}^2$. The key point here is that the user can specify any output dimension for the models. If however, this is not $\mathbb{R}^2$ we opt to use UMAP to perform a final reduction step to $\mathbb{R}^2$. UMAP was chosen as it is the model that combines the best clustering performance with the quickest runtime.

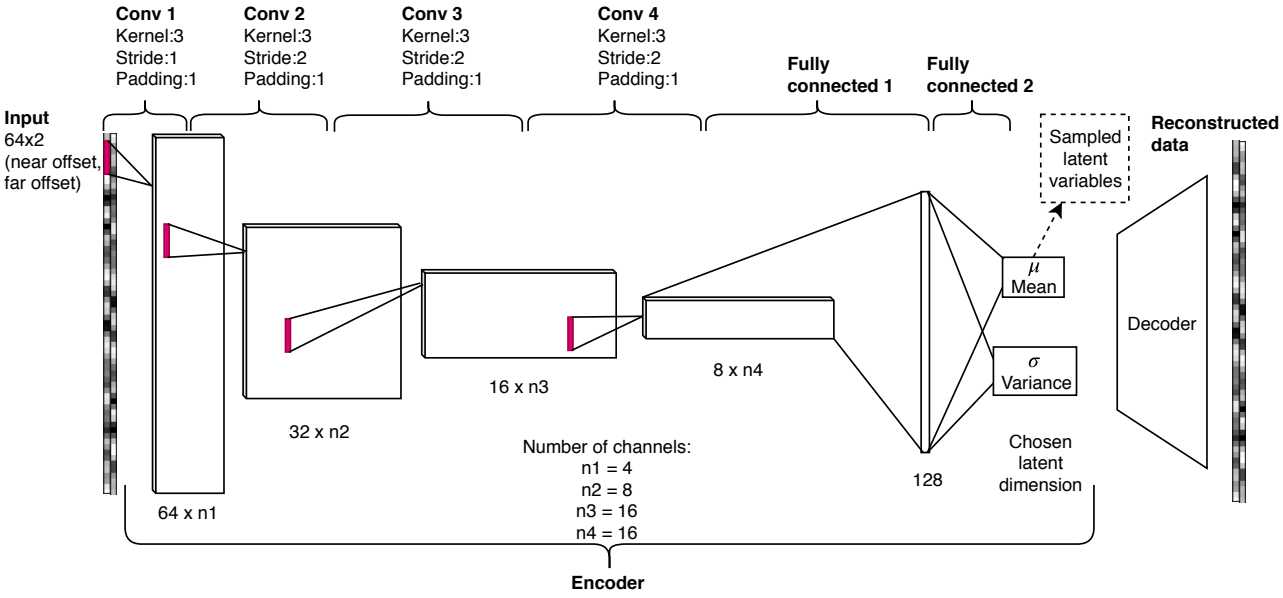

Figure 7: VAE Encoder architecture. Decoder architecture follows the same regime of channel number as the encoder with the opposite direction. Pink rectangles represent the convolutional kernel. Decoder architecture is not exhibited explicitely but follows the same architecture as the encoder in reverse order.

2.5 Visualisation

2D model output is represented in a scatter plot of the latent variables for each sample. The outputs are overlain with colour maps in order to asses whether there has been clustering of these attributes. The use of different matplotlib 'colormap' schemes and value ranges for the colour maps reveals slight nuances in the clustering of different attributes.

3 Software Development Life Cycle

3.1 Development Methodology

The project was split into three distinct phases: Initial research, project planning and software development. With the majority of the time spent in the development phase. The initial research consisted of a literature review of the broader application of machine learning in geosciences and seismic data (Section 1.2). A project plan was constructed around fundamental software requirements and the proposed development methodology. A full agile framework is not required in this setting with a single developer, however the key themes were followed. Notably the goal of developing the minimum viable product (mvp) as quickly as possible and an iterative approach based on feedback from the key stakeholders thereafter. The planned objectives to each 'sprint' or phase was as follows:

1. **1st sprint goal:** develop the minimum viable product which delivers functionality to perform the essential requirements. This included set up of the development environment: working with Git version control and a GitHub repository alongside initial unit testing and test integration with Travis. On completion of the mvp an alpha release of the software was uploaded to PyPi.
2. **2nd sprint goal:** Perform review and feedback process with the project stakeholders (project supervisors), create an agenda of modifications and new additions with specified relative importance. This ensured the most important tasks were completed first, leaving less critical features to the back end of the development window. The focus was on extending the feature capacity of the tool both in the SeismicReduction package and also the GUI.

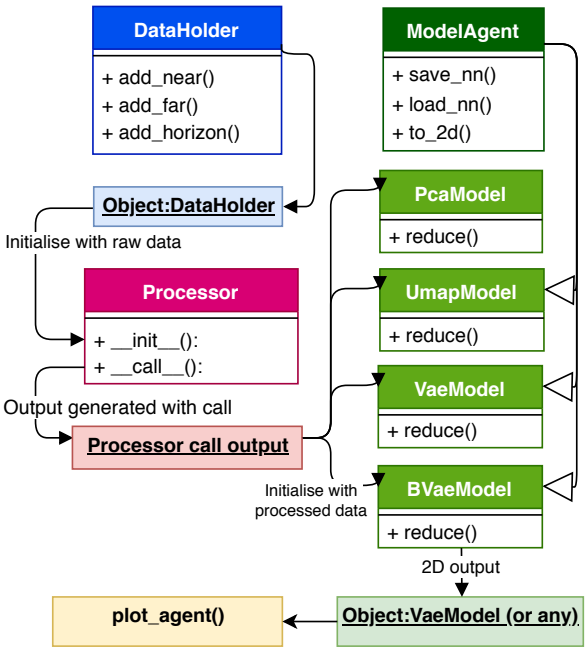

Figure 8: Architectural diagram of class relationships, key outputs, and interactions.

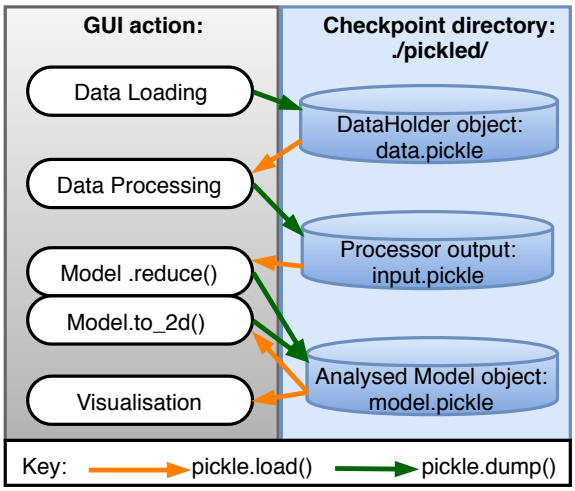

Figure 9: Graphical user interface checkpointing schematic. Highlighting the assigned directory and placeholder name for each key checkpointing component.

3. **3rd sprint goal:** Carry out investigation and optimisation of the specific model algorithms and analysis workflow stages. This task came last as it was far more efficient to carry out on developed software than running on scratch code components. The results are highlighted in Section 5.

3.2 Software Description

3.2.1 Architecture

The architectural overview of SeismicReduction is presented in Figure 8, note only the public methods of each class are presented. The architecture was designed to provide four modules each with the single responsibility for a key stage in the workflow. All modules are written in standard Python3 syntax. The architecture allows for easy extensibility to the software.

3.2.2 Module Overviews

1. **Data loading** is handled by the DataHolder class. The class is designed to have the single responsibility to load and save raw seismic data into a Python object. The expected data to be loaded are near and far offset SEG-Y formatted files and a horizon depth specification file. The conversion of SEG-Y data into standard Python data structures is supported by the segypy Python package.

2. **Pre-processing** is handled by the Processor class. Processor follows the 'factory' design pattern from the Gang of Four (GoF) OOP design pattern paradigms Gamma et al. (1993). This design pattern allows a Processor to generate many different processed outputs from one instantiation of a raw dataset. A useful feature for a user who wishes to sample a range of pre-processing routines and parameters from a single data set. The possible processing routines are described previously.

3. **Model analysis** is handled by an array of daughter classes (PcaModel, UmapModel, VaeModel and BVaeModel) all inheriting from the ModelAgent parent class. ModelAgent hosts the general functionality for object initialisation, data formatting and model loading/saving. There is also a method used to reduce any intermediate dimension into 2D, called to_2d(). This method utilises the UMAP algorithm. Each daughter class has the responsibility of implementing a single unsupervised learning algorithm. After object initialisation the reduce() method is used with all classes as the key operator to run the models. The specific parameters for reduce() vary for each class dependent on the model implemented.

4. **Visualisation** is performed by the plot_agent() function. The function formats and plots two dimensional model outputs with a specified data attribute as a colormap, for example minimum AVO anomaly or horizon depth in trace. The specific colour map and the range of this colour map can also be specified.

3.3 GUI Tool

A key component of the project was the development of a graphical user interface tool. The interface is embedded within a Jupyter notebook using the graphical widgets package for Jupyter: ipywidgets, IPython Development Team (2019). The interface provides access to all of the module routines through a straightforward series of data entry widgets and activation buttons. The interface makes use of dynamic output windows to provide seem-less transition between different ouputs. Each stage of the workflow is check-pointed via the use of the Python package 'pickle' which saves Python objects as binary files. This allows for far quicker running of the analysis as time taken to load a pickle object is far less than the operation of re-loading raw SEG-Y files and re-training the neural networks. Diagrammatic representation of this process is presented in Figure 9.

3.4 Software Testing

Software is tested in two ways: unit tests and integration tests. These tests were implemented into the 'Travis' platform for continuous integration of updates into the software.

Unit tests are used to assess the functionality of individual functions or methods. These tests cover all of the software modules. The specification of tests often came prior to implementation of the actual software functionality. This was not possible in the early stages of the prototype development as it would have hindered development speed. However as the software matured, this form of test driven development was found to be extremely useful in ensuring stable and robust additions to the software were developed.

The second form of testing was integration testing. These integration tests validate the full analysis workflow from the current state of the software and test against a pre-determined verified model output. The integration test is run with the exact same routine and parameters documented when creating the base case. The base case results are created from a verified stable version. These tests are very important to assert that the full analysis is working as expected. Unfortunately if these tests fail, we are not given the exact location of the bug. However they will catch any bugs in the software that somehow do not break the unit tests, but still provide deviation from expected output.

4 Case Study

4.1 Glitne Dataset

The example seismic dataset was acquired from the commercial Glitne oil field. The Glitne field is located on the Norwegian continental shelf in blocks 15/5 and 15/6 in the Sleipner area of the North Sea. The decommissioned field produced 8.9 million Sm^3 of oil in its production lifetime. The reservoir was a deep marine fan system sandstone of Paleocene age, part of the 'Heimdal' formation residing at 2,150 metres depth (Norsk Petroleum (2019)).

The total dataset is composed of two 3D seismic cubes, near offset and far offset. An interpretation for the 'Top Heimdal' horizon reflector is also used. This interpretation specifies the two-way time (twt) of the horizon in each seismic trace. This is useful for specific data processing to focus on the area reservoir and therefore the known presence of hydrocarbons.

The seismic data cubes are composed of 251 cross lines, 101 inlines and 250 samples in twt.

4.2 Results

The results from this dataset are obtained with the following workflow:

- Data processing: Flattening on horizon (12 samples above, 52 below), normalisation to mean 0 and std 1.
- UMAP and PCA: $\mathbb{R}^{128} \rightarrow \mathbb{R}^2$
- VAE and β-VAE: $2 \times \mathbb{R}^{64} \rightarrow \mathbb{R}^2$
- UMAP parameters : n_neighbours:50, min_dist: 0.001
- VAE parameters: Learning rate = 0.0005, epochs= 300
- B-VAE: same as VAE with β=7.

5 Investigation and Optimisation

5.1 Investigation of VAE Training

As previously covered the VAE networks are trained via minimisation of the (negative) ELBO objective function. Minimisation of this function does not explicitly mean latent variables will represent meaningful geophysical features. ELBO simply references a lower-bound to the log-likelihood, often the actual log-likelihood is used instead for model comparisons. However for simplicity in this study we stick with the ELBO. Minimisation of the (negative) log-likelihood or ELBO function alone is not enough to ensure the model is successfully extracting physically meaningful features. In the absence of other intrinsic data attributes we use cross-referencing of model outputs with the AVO anomaly and horizon depth as a method to assess the ability of our model to extract geophysical features.

A series of VAE investigations to optimise the model training and gauge an understanding on how factors such as dataset size and model architecture effect ELBO values.

5.1.1 Optimisation of Training Parameters

Machine learning model parameters are often interdependent and therefore to find the global minimum it is necessary to perform a grid search of all possible parameter space. The process followed in this optimisation was as follows: 1. Comparison of two popular optimisers finding the optimal parameters for each. 2. Use the best optimiser on array of model architectures and test again with different learning rates.

Optimiser

The task of the optimiser in machine learning model training is to minimise the objective function. This is achieved via the update of model parameters based on gradients calculated at each epoch. Two main algorithms were assessed for the use of training the VAE neural network: mini-batch stochastic gradient descent with momentum (SGD) and adaptive moment estimation (Adam).

SGD is one of the most straightforward optimisation algorithms available. Model parameters are updated after every mini-batch (segments of the whole training set) based on the learning rate and gradients. The addition of momentum acts to dampen the stochastic oscillations in SGD by steering updates with 'momentum'. The momentum term refers to the addition of a specified fraction of the previous parameter update value which is added to the current parameter updates.

The Adam algorithm finds adaptive learning rates for each parameter of the model. The learning rates are based on an exponentially decaying average of past squared gradients and an exponentially decaying average of past gradients. An advantage is that best results can often be achieved without the necessity to tune learning rate.

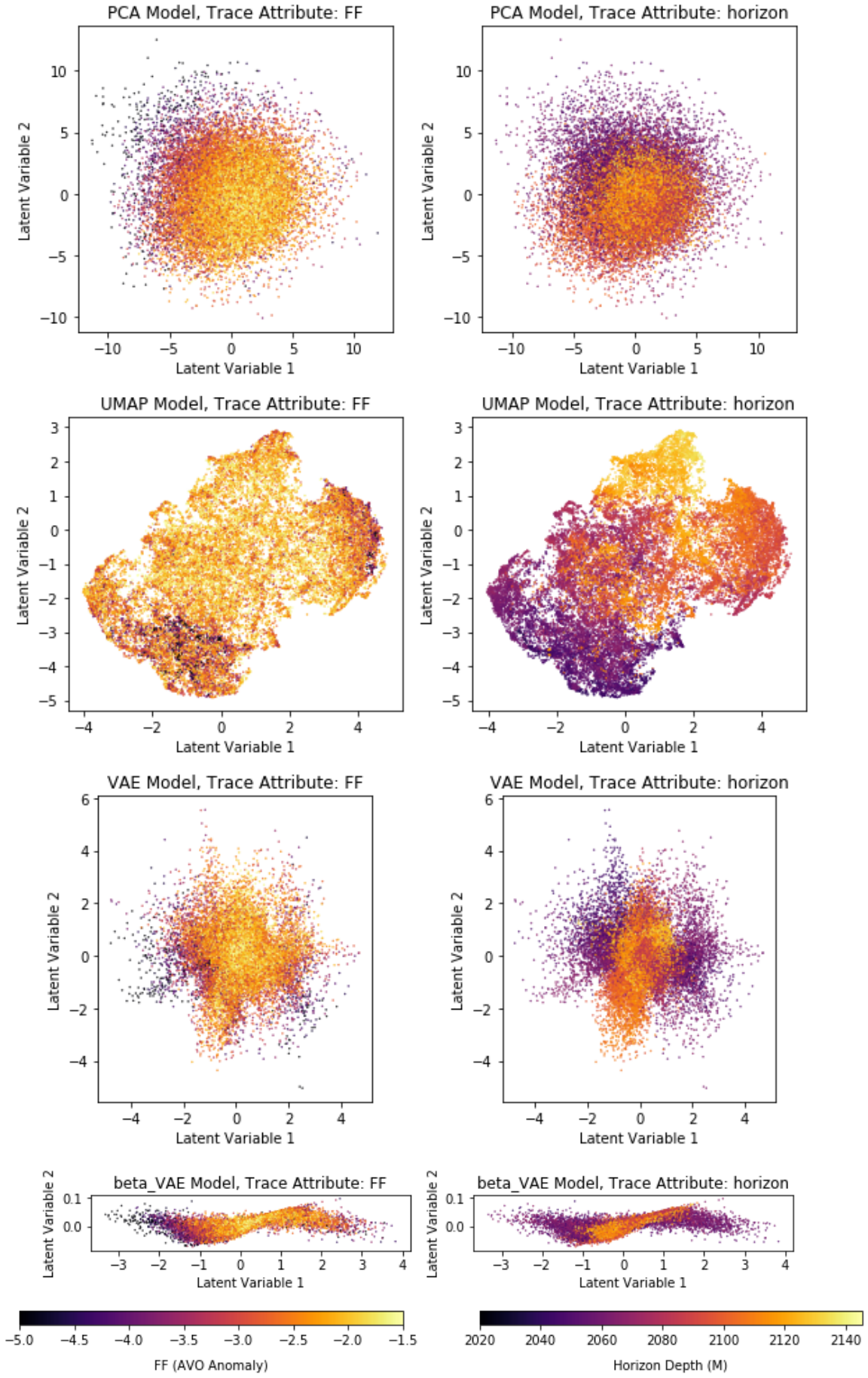

Figure 10: Results from the case study of the Glitne field seismic data set run on all models. Column 1 refers to models with the AVO fluid factor (FF) attribute plotted, Column 2 presents the outputs with the horizon depth in twt (horizon) attribute plotted

The Adam method has often been shown to have superior performance in terms of speed and accuracy in training, however the review by Wilson et al. (2017) states, "We observe that the solutions found by adaptive methods generalise worse (often significantly worse) than SGD, even when these solutions have better training performance. These results suggest that practitioners should reconsider the use of adaptive methods to train neural networks." It is therefore not a foregone conclusion that Adam should always be the optimiser of choice. Both algorithms were tuned to find optimal parameters and then compared against each other. For Adam this involved a pure line search for the optimal learning rate. For SGD this involved a grid search of the learning rate and momentum.

For Adam the optimal learning rate was found to be 0.0005, delivering a minimimum ELBO of 90.1. Across the search the values varied by 10% indicating tuning the learning rate on Adam was far more important than predicted. The optimum ELBO for SGD (90.4) was found with a learning rate of 0.00005 and a momentum of 0.7. The difference between the two optimisers is negligible but Adam was chosen for its marginal performance advantage. A caveat for both optimisers: the lowest ELBO values were achieved with relatively small learning rates where the model did appear to slightly overfit. This over-fitting was recognised by a divergence of the validation and training losses. With progressive epochs the training loss continued decreasing with the validation loss starting to increase after reaching a minima around 100 epochs.

Neural Network Architecture

A VAE can be viewed as two distinct neural networks, the encoder and the decoder. The only constraint on these networks is that the encoder input dimensions matches the decoder output dimensions. The encoder output must also match the decoder input (consistent latent space dimension). The way each network achieves this transformation is un-constrained and can be seen as a free parameter. Therefore the number of convolutional layers and channels therein can be varied as needed. Model capacity can be approximated as a function of the total number of model parameters. The larger the number of parameters the larger the capacity of the model and therefore the greater total function space available. If a model possesses too many parameters for the task it is at risk of over fitting and loss of model generality.

As seen in the optimiser investigation, the model was overfitting when running on small learning rates. We show that by using a model architecture with fewer channels per convolutional layers in the encoder, and decoder, the model is able to achieve the same optimimum ELBO values, but with no signs of overfitting into later epochs. To provide a structured approach to investigating VAE architectures, only encoders and decoders as exact inverses were considered. Meaning the decoder matched the number of encoder convolutional layers with deconvolutional layers with a mirrored channel scheme.

Three different VAE model architectures were investigated. The number of convolution/deconvolution layers was kept constant and only the scheme of channel numbers was altered. These are described in Table 2, the channel number specifications refer to the same layers as in Figure 7. The three VAEs were run on two different learning rates, 0.01 and 0.0005. With a learning rate at 0.01 none of the models show over fitting, the medium and larger models enable a marginally lower ELBO value. Using a learning rate of 0.0005 the medium and large models both show over fitting. The 'Small' model does not show any over fitting within the 300 epochs run. The 'Small' model also achieves a comparable minimum ELBO (90.3) with a considerably lower final ELBO (90.5) compared with the other two models.

Network:	**Channels**				**Fully connected layer Parameters**	**Learning Rate 0.01**		**Learning Rate 0.0005**		**Time to train full 300 epochs**
	n1	n2	n3	n4		Final ELBO	Min ELBO	Final ELBO	Min ELBO	**(seconds)**
Small	4	8	16	16	128	96.9	96.2	90.5	90.3	2679
Medium	3	32	32	32	128	96.4	95.5	92.6	90.1	4133
Large	8	32	64	128	128	96.3	95.1	106.4	90.1	6147

Table 2: VAE neural network architecture schemes and associated results run with two different learning rates. Timing was taken for the full 300 epochs with the models running on learning rate of 0.0005

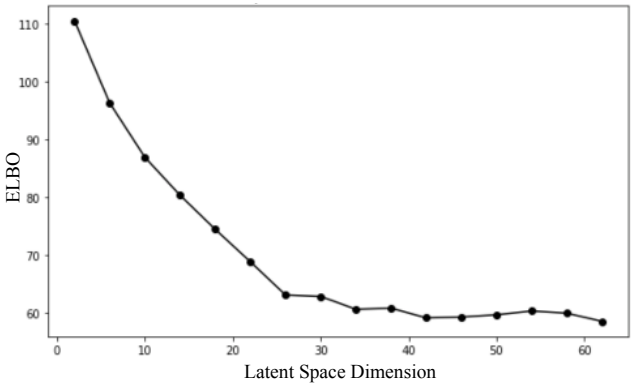

Figure 11: Effect of latent space dimension on final validation ELBO values

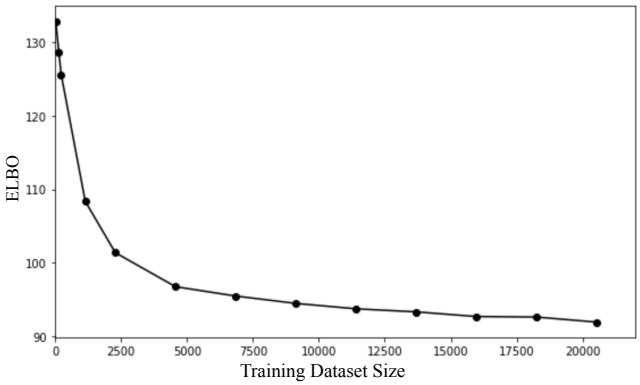

Figure 12: Effect of the training dataset size on ELBO values

Training time
The extra parameter calculations of the 'larger' models mean computational cost results in training taking considerably longer to train compared with the smaller scheme.

Optimisation conclusions
Based on the optimisation tests it was discovered that a superior performance is achieved using the Adam optimiser with a learning rate of 0.0005. To combat the observed overfitting with the benchmark model, the 'small' model with reduced channels and therefore parameters was used. The additional benefit is the reduction in runtime alongside performance increase.

5.1.2 Dimension of Latent Space

The latent space dimension or 'hidden size' of the VAE network is an adjustable parameter of the model. A study of the effect of latent dimension on ELBO values will provide relevant insight into neural network training sensitivities. Importantly documentation of these values will provide a valuable bench mark for users to check the model training results against.

It was expected that increase in the hidden size would enable lower values of the (negative) ELBO function. A larger latent dimension translates to less of an information bottle neck in the encoder and therefore reduced reconstruction error achieved via the decoder. We observe an exponential decrease in the (negative) ELBO from 2 to 35 latent dimension size. At this point the ELBO values appear to flatten off. The likely reason for this is that the ELBO value has become fully dependent on the KL divergence term, as the reconstruction error has converged to zero.

5.1.3 Training Dataset Size

The volume of training data available is relevant to gain an understanding for how models perform with different sized datasets. Hopefully this investigation can also act as a benchmark for future studies to asses performance against. A test set was initially sampled randomly from the data for use as the validation ELBO score in each of the runs. Different sized training sets (of systematically increasing size) were then extracted regularly from the remaining pool of data.

As expected the (negative) ELBO value decreases with an increased training dataset size. The trend is exponential and appears to flatten off at around 7000 samples. This gives us confidence that for the proposed VAE model a dataset of size 7000 traces or larger would be adequate to achieve a comparative model performance. However this is not certain as model performance may vary dependent on the specific dataset.

5.2 Parameter Effects on Cluster Outputs

One of the key purposes of this tool is to enable an efficient exploration of the parameter space of models used. In this section we provide a systematic parameter investigation of two of the models to provide insight into the effects on model output clustering. We follow the standard processing workflow, specified in Section 4, and only vary the parameters specified.

5.2.1 β Parameter

In this investigation the effect on varying the β parameter of the β-VAE is tested on the model output with a latent space of 2. The result of this investigation are presented in Figure 13.

A β-VAE trained with a β parameter of 1 is equivalent to the standard VAE. When we decrease the β value we observe the latent space variables increase in range from a magnitude of 5 to 20. There is no significant change observed in the appearance of the clustering of the data. When the beta value is increased past a certain threshold, the distribution of data in the latent space moves from a near gaussian, centred at the origin, to a flat elongated shape. The transition between the gaussion to flattened output happens in a small <1 β window. Interestingly if the model training is stopped at different epoch numbers, we see the transition at different β values. The higher the epoch (the more each model is trained) the higher the beta value required to observe the transition. The investigation was run on a learning rate of 0.0005 for 300 epochs.

High β-value models have effectively condensed the data into a single dimension. Further to this it appears the model has still preserved a one-dimensional version of sorting/clustering observed in the standard VAE. High AVO values are placed near zero with low AVO values on the periphery of the mass. This could prove to be a very useful feature if the magnitude of the dominant latent variable alone could prove to be a direct indicator of AVO value.

5.2.2 Umap Parameters

The two key parameters for the Umap algorithm are n_neighbours and min_dist as described previously. These parameters were investigated with a grid search approach. The results are presented in Figure 14. With a low n_neighbours parameter the model produces a near normal distribution of datapoints. With larger values of this parameter the data forms a more irregular pattern. This is due to the algorithm capturing a nuance of the global relationships in the dataset, as apposed preservation of only local relationships. This has forced a departure from the overall normal distribution to a more non-homogeneous one.

6 Discussion and Conclusion

6.1 Results Discussion

The results discussed are those run by the standard workflow visualised in Figure 10.

PCA
The spatial distribution of the outputs from the PCA model appear to follow a gaussian like distribution in both latent variables. The AVO colour map plot shows reasonable sorting of both AVO and horizon attribute values. As this is simply a linear technique we would not expect complex clustering structures within the output. This result is indicative of PCA on high dimensional data. The output is able to capture some level of clustering of classes however the cluster boundaries overlap and are far less distinct when compared with the non-linear model clustering.

UMAP
The UMAP algorithm has created a non-homogeneous 2D spatial distribution of points. There are small isolated clusters observed in the AVO attribute plot, these are distributed across the latent space. Interestingly it is the horizon depth attribute which shows a much more distinct pattern to

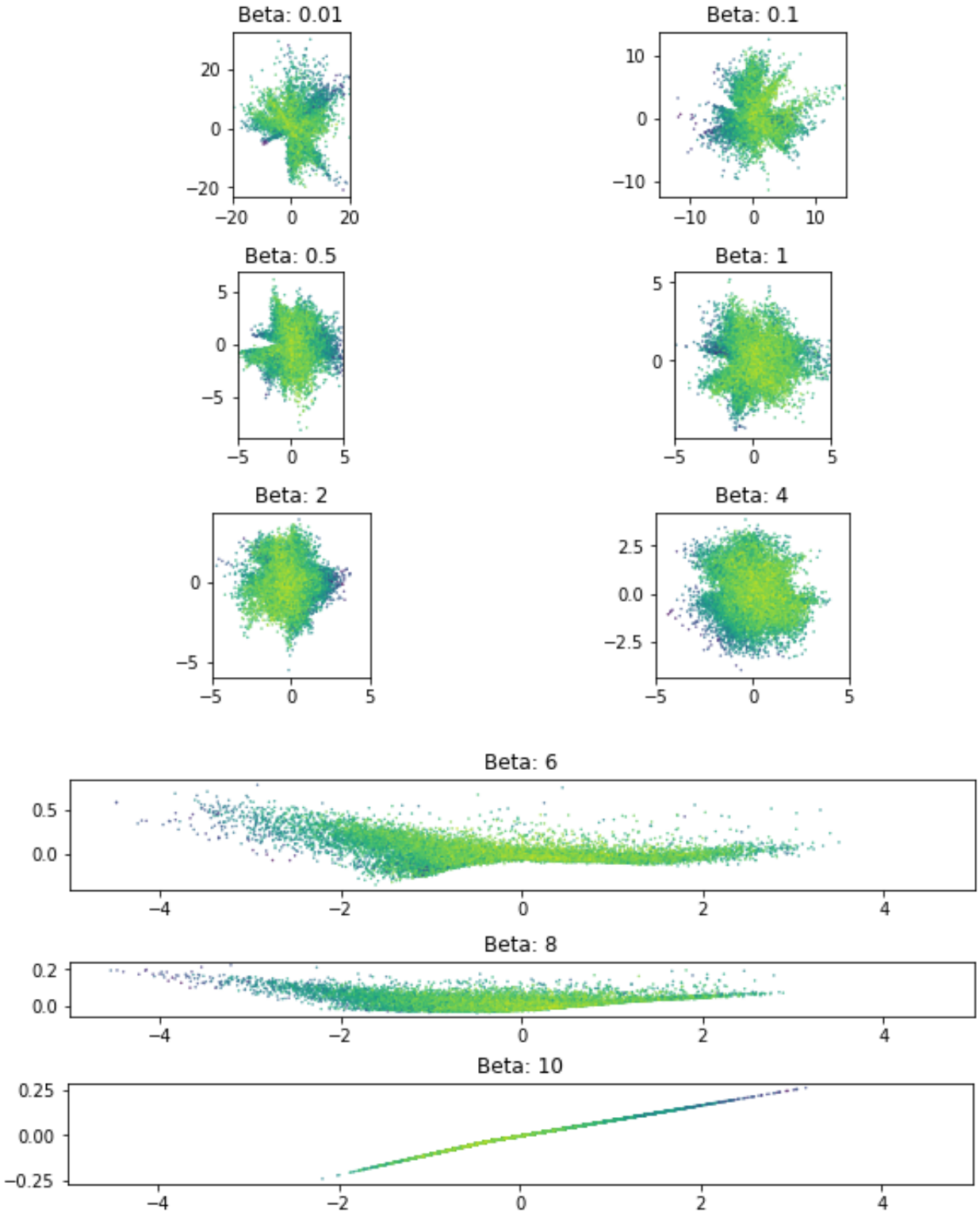

Figure 13: β-parameter investigations. Run with a learning rate of 0.0005, for 300 training epochs. The vertical axis for the Beta: 8 and Beta: 10 plots have both been expanded by a factor of 3 to allow for clear visualisation of the plots.

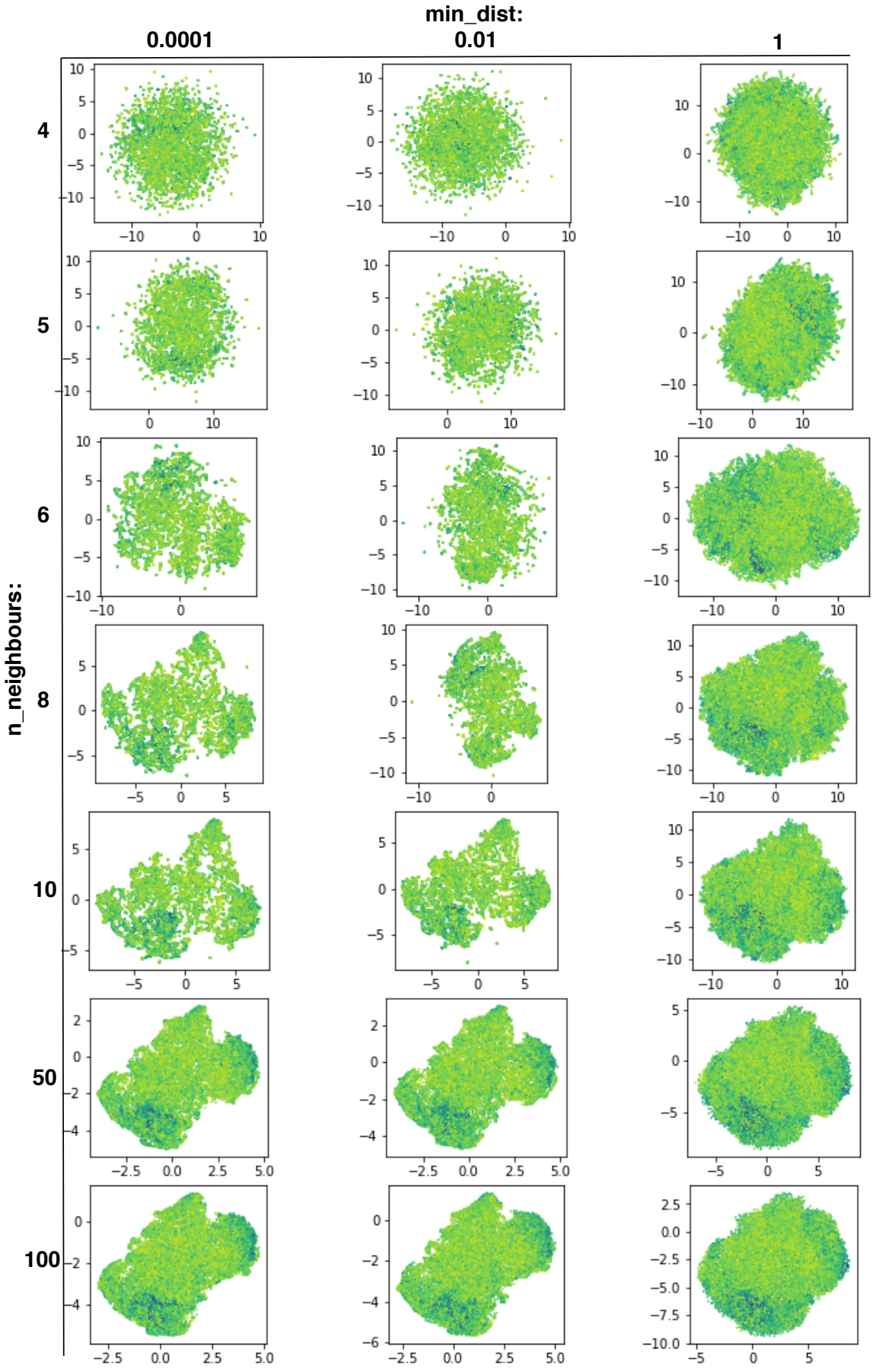

Figure 14: UMAP parameter investigation. The rows increment through the n_neighbours parameter, the columns increment the min_dist parameter. The plotted attribute is the AVO anomaly fluid factor.

its distribution. The horizon depth appears to be strongly sorted with large regions of similar depth points gathered together. The transition between these regions also appears fairly continuous. This can be explained fairly intuitively as the UMAP algorithm is searching for a manifold approximation. It is reasonable to assume the seismic data is laterally continuous and therefore that adjacent traces share similar amplitude signals and also horizon depth attributes. Therefore we can infer that the manifold the UMAP algorithm has approximated is heavily influenced by the spacial attributes of data. And therefore spatial relationships in the physical domain are preserved in the low dimensional domain.

VAE
The spatial distribution of the VAE output is Gaussian-like with concentrations of data in elongated features. The minimum AVO colourmap shows a concentration of the low values on the periphery of the latent space with high AVO anomaly concentrated in the centre. Due to the VAE fitting distributions to a prior normal distribution we expect continuity to the latent variables. A very useful feature if these variables show strong correlation to a particular attribute of the data. It appears that the horizon plot shows a sharp transition in the variable between clusters. However it appears the transition of the AVO attribute is far more gradual and continuous.

β-VAE
The model has created outputs with a large variation in one latent variable compared to the other. This results in the spatial distribution of the β-VAE output as a flat elongated structure. The AVO attribute shows strong clustering within the representation. The high AVO anomaly values are centred over the origin with increases in magnitude in latent variable translating to decrease of the AVO anomaly. The same features of the continuity in variables as seen in VAE is seen here.

AVO v Horizon Attribute
There are some key points to be made in comparison of the plots of the AVO and horizon depth attributes that could not be made from either one in isolation. Firstly, we can observe that both attributes exhibit at least some form of sorting or clustering from all of the models. We can also observe that there are similarities in the clusters of the AVO anomaly and the clusters of horizon depth. This observation suggests it is inherent in the data that horizon depth and AVO anomaly are coupled to some extent. On closer inspection it is revealed that the AVO anomaly values suggestive of hydrocarbon presence are matched with values of shallow horizon. This makes sense intuitively as the Glitne oil field is known to trap oil in a structural high. Clearly the coupling of these attributes is not a perfect linear relationship otherwise both plots would look exactly the same. We did not need machine learning models to make the connection between AVO anomaly and horizon depth. But there seems to be evidence of independent clustering of each attribute by the models. This is highlighted by instances of clustering of one attribute where there is no equivalent clustering in the opposing attribute. This is observed for both attributes and is particularly evident in UMAP. This suggests that models are somehow picking up on nuances of AVO and horizon depth independently of each other.

However for the most significant clustering regions in each model the AVO anomaly closely follows the horizon depth. Therefore it is not clear whether the models have learnt the low dimension representations based on the nuances in the data describing AVO or Horizon depth. One way to gain a better understanding of this would be to run the analysis on two specific datasets: one with a topologically varying horizon but an absence of hydrocarbon; the other with a near flat horizon and hydrocarbon presence.

6.2 Challenges in Development

Generality of parameters
A key requirement of the software was the ability to run the analysis with a wide array of variable user defined parameters. There is opportunity for free parameters to be provided at nearly every stage in

the workflow. A prime example is the variability in dimension of input samples dependent on data processing parameters chosen. This variability was initially incompatible with the statically defined neural network architecture of the VAE. The solution was the addition of dynamic variability to the internal model architecture at model initialisation. This was facilitated by passing the expected input dimension at VAE initialisation. Although the convolutional layers are independent of dimension, it was necessary to define the size of the fully connected linear layers which follow. The linear dimension is calculated by the dimension of each channel multiplied by the number of channels. The input dimension is constrained to a factor of four. This is to ensure the input dimension reduced by the encoder matches the dimension reconstructed from the decoder.

6.3 Limitations to Workflow and Analysis

AVO as a Hydrocarbon Indicator
The AVO anomaly calculation in this software is a simplified model compared to the industry standard approach. However one of the challenges in the industry is a lack of dissemination of this expertise. This tool provides the opportunity for quick initial research on a dataset, that if showing promising results could be promoted for the full AVO analysis. The AVO anomaly here provides only an approximation to the hydrocarbon content, where low values correspond to increased likelihood of hydrocarbon presence. We have not calibrated the AVO to the dataset to define a discrete 'cut-off' value signifying the anomaly value which corresponds to a transition from hydrocarbon absence to presence. This would be a relatively straightforward operation requiring the original maps of the oil field extent at time of seismic survey. If there were publicly available records of the physical extent of the Glitne field, we could also engineer a new attribute which specified explicitly if a trace was within the field or not. The output would be plotted with a binary colour map to highlight explicit clustering of the 'hydrocarbon' traces. With this we could place a higher degree of confidence in the models ability directly recognise hydrocarbon presence in seismic traces.

Horizon data
Interpretation of horizon depths is carried out as a manual, somewhat subjective process. Therefore it is not always 100% accurate for identifying the correct depth. If we are flattening the data on a horizon which is not necessarily accurate or worse not laterally consistent, we may be jeopardising the opportunity of our model to learn efficiently. Small variations away from the true horizon may not be an issue for the convolutional models (VAE and β-VAE) they are however for PCA and UMAP. Models benefit from key features, such as the target horizon amplitude, to be present in the same dimension for all of the sample vectors. If the actual position of this key feature is varying sample to sample, this will limit the models ability to learn an efficient mapping from the dataset.

Dataset
Only a single dataset was used in the calibration and optimisation of the machine learning models. Therefore it is unknown how the models will perform on new datasets. Despite the normalisation of data processing, seismic data from one survery will vary significantly to that of another. The difference in geological setting and seismic processing will both provide provide disparity between the data distributions. PCA and UMAP are expected to be adaptable to new datasets without any modifications necessary to the code. However, the deep neural network based VAE and β-VAE are very brittle, even small shifts in data distribution can lead to disparate results. New datasets will certainly require models to be re-trained. However, it is expected that the neural network architecture provided here will be able to generalise well to different datasets. The hyperparameters from this case study should also provide a reasonable first guess, however specific tuning on new datasets may be required for optimum performance.

6.4 Conclusion and Outlook

The software developed has extended and automated the initial investigations carried out by Mosser et al. (2019). The β-VAE model has been applied to seismic data for the first time. The overlay of certain 'per-trace' attributes reveal all models are able to sort or cluster the data in relation to these attributes, however the exact underlying physical features the models are learning is still unclear.

This tool provides a qualitative analysis on the ability and degree of clustering low dimensional space. However, the potential utility of these model outputs if far greater. For example Mosser et al. (2019) exhibits potential further applications: clusters from the analysis output can be mapped back to the physical domain revealing how the latent space variables show similar distribution in the physical domain. Put simply: the models cluster spatially similar traces in the latent space. This highlights the models ability in finding spatial relationships in the data. This happens despite an absence of explicit spatial information being being provided to the models. Inclusion of the automation of this further analysis would be a valuable addition to the software's overall utility.

We propose the most relevant steps to be made in furthering the utility of this tool and the understanding of the analysis are as follows:

New features:

- Add compatibility for other seismic data file formats
- Automation of latent space cluster visualisation back in physical domain
- Addition of new unsupervised models/ algorithms:
  - Plain auto-encoder
  - K-means clustering for automatic model output clustering

Further investigations:

- Exploration of new datasets with this tool.
- Calibration of clusters with unexplored 'per-trace' data attributes.
- Sample alternative methods to dataset partitioning, horizontal or cubic volume sampling of the dataset.
- Comprehensive study with the industry standard AVO analysis techniques.

References

Avseth, P., Mukerji, T., and Mavko, G. (2005). *Quantitative Seismic Interpretation*.

Backus, G. and Gilbert, F. (1970). Uniqueness in the Inversion of Inaccurate Gross Earth Data. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*.

Bergen, K. J., Johnson, P. A., de Hoop, M. V., and Beroza, G. C. (2019). Machine learning for data-driven discovery in solid Earth geoscience. *Science*, 363(6433):eaau0323.

Burgess, C. P., Higgins, I., Pal, A., Matthey, L., Watters, N., Desjardins, G., and Lerchner, A. (2018). Understanding disentangling in β-VAE. *arXiv e-prints*.

Burgoyne, J. A. and McAdams, S. (2007). Non-linear scaling techniques for uncovering the perceptual dimensions of timbre. In *International Computer Music Conference, ICMC 2007*, pages 73–76.

Chiburis, E., Leaney, S., Skidmore, C., Franck, C., and Mchugo, S. (1993). Hydrocarbon detection with AVO. *Oilfield Review*.

Desai, S. (2019). Application of Deep Learning Techniques for Amplitude Variation with Offset Analysis. *Imperial College MSci Project*.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1993). Design patterns: Abstraction and reuse of object-oriented design. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*.

Hall, B. (2016). Facies classification using machine learning. *The Leading Edge*.

Hammond, A. L. (1974). Bright spot: Better seismological indicators of gas and oil. *Science*, 185(4150):515–517.

Hansen, T. M. (2018). A python module for reading/writing of seg-y formatted files. https://github.com/cultpenguin/segypy.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2017). beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR*, 2(5):6.

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

IPython Development Team (2019). Ipython html widgets for jupyter. https://github.com/jupyter-widgets/ipywidgets.

Karpatne, A., Ebert-Uphoff, I., Ravela, S., Babaie, H. A., and Kumar, V. (2019). Machine Learning for the Geosciences: Challenges and Opportunities. *IEEE Transactions on Knowledge and Data Engineering*, 31(8):1544–1554.

Kingma, D. P. and Welling, M. (2013). Auto-Encoding Variational Bayes. *arXiv e-prints*, page arXiv:1312.6114.

McCann, M. T., Jin, K. H., and Unser, M. (2017). Convolutional neural networks for inverse problems in imaging: A review. *IEEE Signal Processing Magazine*.

McInnes, L., Healy, J., and Melville, J. (2018). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *ArXiv e-prints*.

Mora, P. (1986). Nonlinear elastic inversion of multioffset seismic data. In *1986 SEG Annual Meeting, SEG 1986*.

Mosser, L. (2018). Anomaly screening applying pseudo-wells. https://github.com/LukasMosser/ASAP.

Mosser, L., Dubrule, O., and Blunt, M. (2018). Stochastic seismic waveform inversion using generative adversarial networks as a geological prior. In *1st EAGE/PESGB Workshop on Machine Learning*.

Mosser, L., Monte, A. A., Avseth, P., Draege, A., and Macgregor, L. (2019). Investigating the AVO Signature of a North Sea Oil-Field Using Pseudo-Wells and Unsupervised Deep Learning. In *81st EAGE Conference and Exhibition 2019 - AI/Digitalization for Interpretation - AVO Application*.

Norsk Petroleum (2019). Glitne field fact-sheet. https://www.norskpetroleum.no/en/facts/field/glitne/.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*.

Poličar, P. G., Stražar, M., and Zupan, B. (2019). opentsne: a modular python library for t-sne dimensionality reduction and embedding. *bioRxiv*.

Rouet-Leduc, B., Hulbert, C., Bolton, D. C., Ren, C. X., Riviere, J., Marone, C., Guyer, R. A., and Johnson, P. A. (2018). Estimating Fault Friction From Seismic Signals in the Laboratory. *Geophysical Research Letters*.

Russell, B. H. (2014). Prestack seismic amplitude analysis: An integrated overview. *Interpretation*.

Sathya, R. and Abraham, A. (2013). Comparison of supervised and unsupervised learning algorithms for pattern classification. *International Journal of Advanced Research in Artificial Intelligence*, 2(2):34–38.

van der Maaten, L. and Geoffrey E., H. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 164(2210):10.

Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., and Recht, B. (2017). The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems*.

Wold, S., Esbensen, K., and Geladi, P. (1987). Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2(1):37 – 52. Proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists.

Ye, J., Janardan, R., and Li, Q. (2005). Two-dimensional linear discriminant analysis. In *Advances in neural information processing systems*, pages 1569–1576.