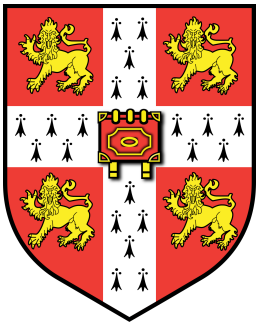
MPhil in Machine Learning and Machine Intelligence
Cambridge University Department of Engineering



Continuity of autoencoders, unsupervised anomaly detection and Deep Atlases

Dissertation submitted by

Charles A. Arnal
Homerton College

August 2021

Under the supervision of Mihaela van der Schaar and Fergus Imrie

Many thanks to Mihaela van der Schaar, Fergus Imrie, James Jordon and Changhee Lee for their guidance, and to Vincent Divol for helpful discussions.

Declaration

I, Charles Arnal of Homerton College, being a candidate for the MPhil in Machine Learning and Machine Intelligence, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.
I further declare the software that was used for this thesis: all computing experiments were carried out in Python. I relied on standard libraries such as Numpy, as well as two specialized libraries: Tensorflow and Scikit-learn. All plots were made using either the Matplotlib library or MATLAB. None of the above software was modified and no other third-party software was used.

Word count: 14'876

Charles Arnal
August 2021

Continuity of autoencoders, unsupervised anomaly detection and Deep Atlases

Abstract

Dimensionality reduction techniques aim to encode data in a compact way, usually by embedding it in a lower dimensional space, while preserving crucial information. They are often applied to the data as a pre-processing step for some downstream task. While there are many such methods, those based on the use of neural networks organised as autoencoders have enjoyed great success and popularity in recent years.

Among many other applications, the idea arose to use autoencoders to perform unsupervised anomaly detection, the task of distinguishing anomalies from normal points in entirely unlabelled data sets, by using the difference between the original data point and its reconstruction by the autoencoder as a measure of anomaly. This method is very dependent on the quality of the reconstruction of normal data points and relies on several key assumptions regarding the nature of anomalies.

In this thesis, we examine those assumptions, as well as certain limitations of classical autoencoders. In particular, we show that the fact that the mappings that autoencoders learn are usually continuous makes them unable to accurately encode certain data sets (specifically those related to closed manifolds), which in turn impacts in certain situations their usefulness for unsupervised anomaly detection and dimensionality reduction in general.

Hoping to solve some of those problems, we also propose a piecewise continuous autoencoder, named Deep Atlas, designed to avoid the shortcomings of classical autoencoders. We test it thoroughly, and analyze its strengths and limitations.

Contents

Introduction **11**

I **13**

1 Dimensionality reduction **15**
1.1 Definition, motivation and existing techniques 15

2 Limitations of continuous dimensionality reduction algorithms **19**
2.1 Theoretical aspects 19
2.2 Experimental illustration 22

II **27**

3 Unsupervised anomaly detection **29**
3.1 Definition and motivation 29
3.2 Unsupervised anomaly detection and autoencoders 30

4 Anomalies and structure **33**
4.1 Introduction 33
4.2 What is an anomaly? 33
4.2.1 Menagerie of examples 35
4.3 How do we identify anomalies? 40

III **43**

5 Deep Atlas **45**
5.1 A geometric intuition 45
5.2 The main algorithm 46
5.3 Training 47
5.3.1 Losses 48

5.3.2 Chart splitting algorithm 51
5.4 On the minimal number of charts required 53

6 Comparison to other algorithms **55**
6.1 Autoencoders 55
6.2 Mixture of experts 55
6.3 Other piecewise continuous manifold learning techniques 57

7 Experiments **59**
7.1 Introduction 59
7.2 Illustration of the main mechanisms 60
7.3 Hyperparameter testing 64
7.4 Classification 70
7.5 Unsupervised anomaly detection 70
7.6 Discussion 71

Conclusion **73**

Bibliography **75**

A Additional details on the experimental protocols **81**
A.1 Datasets 81
A.1.1 Torus union sphere 81
A.1.2 2-dimensional sphere 83
A.1.3 Swiss roll 83
A.1.4 40-dimensional sphere 83
A.2 Model configuration and training 83
A.2.1 Torus union sphere 84
A.2.2 Swiss roll 84
A.2.3 40-dimensional sphere 84
A.2.4 MNIST 84

Introduction

Reducing the dimensionality of data in a way that preserves crucial information can be of great help, either to make visualisation easier, or as a first step towards some other task, such as regression, classification or anomaly detection (see [Engel et al., 2012] for a general survey), as high-dimensional data present unique challenges to most algorithms (see [Johnstone and Titterington, 2009]).

While many techniques exist, a relatively recent trend (see [Hinton and Salakhutdinov, 2006]) is to use deep neural networks organized as autoencoders to perform dimensionality reduction (though the idea dates from the 90s, see [Kramer, 1991]). An autoencoder is a system composed of two neural networks, an encoder and a decoder. The encoder maps data to a lower-dimensional state, while the decoder aims to reconstruct the initial input from the encoding.

Autoencoders benefit from the flexibility and power of deep neural networks, and can often be trained conjointly with another algorithm tasked with some downstream task to learn a compact representation that preserves the information relevant for said task. They have enjoyed great success in a variety of contexts (image classification [Geng et al., 2015], self-supervised learning on tabular data [Yoon et al., 2020], etc.).

One particular field of application of autoencoders is Unsupervised Anomaly Detection (UAD), which is the task of separating anomalies from normal points in an entirely unlabelled data set (see [Goldstein and Uchida, 2016] for a survey). Depending on the circumstances, anomalies can represent illnesses, computer intrusions, financial frauds, manufacturing flaws, etc. Unsupervised approaches are made necessary in many contexts in which obtaining large amounts of labelled data proves impractical, either due to costs or to intrinsic difficulty, while unlabelled data abound.

A variety of UAD techniques based on autoencoders recently emerged (see [Chalapathy and Chawla, 2019] for a survey). The core idea of most of these is to train an autoencoder on the unlabelled data set, then to use some function of the reconstruction error as a measure of anomality. The intuition is that as the normal points are more numerous in the data set and less noisy than the anomalies, the autoencoder should be much better at reconstructing them after training. The method is consequently very dependant on the quality of the encoding and the reconstruction of the normal points. Though a partially faulty encoding can be sufficient for other tasks (when classifying images, a blurry and partially distorted picture of a dog can often still be correctly distinguished from that

of a cat), even small imprecisions can be perceived as anomalies.

While autoencoders and UAD are well-studied subjects on their own, the combination of the two is a new and promising development which combines the potentials of deep learning and unsupervised learning. We feel that though clever algorithms have already been developed (such as in [Yoon et al., 2021]), there is a lack of theoretical reflection regarding the underlying premises. The family of autoencoder-based UAD algorithms summarily described above relies in particular on two implicit assumptions:
The first, which also applies to general dimensionality reduction, is that autoencoders are a flexible encoding system that is a priori suitable to any kind of data. The second is that measuring how well a given dimensionality reduction algorithm (which need not be an autoencoder) trained on the data set reconstructs a data point yields a good measure of anomality.

In this thesis, we question both these assumptions. In the first part, we define dimensionality reduction and briefly review the existing methods in Chapter 1, with a particular focus on manifold learning methods and autoencoders. In Chapter 2, we give a theoretical exposition of some of the limitations of classical autoencoders: we prove that learning continuous functions, which might a priori seem like a desirable quality, keeps them from learning accurate encodings of certain types of data - specifically data that lie on a close manifold. We also illustrate our claims with a few qualitative experiments. As far as we know, only the authors of [Batson et al., 2021] have made a similar observation, with a much less detailed analysis.

In the second part of the thesis, we define in Chapter 3 the task of unsupervised anomaly detection and give some relevant background, particularly on autoencoder-based methods. We also discuss the consequences for these methods of the limitations exposed in the preceding chapter. In Chapter 4, we reflect more generally on the assumptions regarding the nature of anomalies that underlie most UAD methods. Again, the angle of analysis adopted seems to have been previously mostly unexplored.

In the final part, we describe in Chapter 5 a new type of piecewise continuous autoencoder inspired by a mathematical intuition from di□erential geometry, which we nicknamed Deep Atlas. We designed it to overcome some of the shortcomings of classical autoencoders detailed in earlier chapters. We compare Deep Atlas to related algorithms in Chapter 6, and we test it with many di□erent hyperparameter configurations in Chapter 7. We also apply it to classification and UAD tasks in comparison to simple autoencoders.

We observe that while Deep Atlas is proficient at faithfully encoding and reconstructing data from simple closed manifolds (where it significantly outperforms classical autoencoders), it enjoys much more limited success with more complex data sets - we analyse why, and discuss possible improvements.

We finally summarize our findings and suggest further avenues of research in the Conclusion. All of our code is available on: `https://github.com/CharlesArnal/Deep_Atlas`

Part I

Chapter 1

Dimensionality reduction

We briefly define dimensionality reduction and mention some current techniques.

1.1 Definition, motivation and existing techniques

We call *dimensionality reduction algorithm* any method that aims to reduce the dimensionality of data while preserving as much information as possible. It is often crucial to reduce the dimensionality of data, either to better visualize it in two or three dimensions, or to prepare it for some downstream task, such as classification, clustering or anomaly detection (see [Engel et al., 2012] or [Sorzano et al., 2014] for surveys). Many methods used for such tasks are poorly suited to high and very high-dimensional data, for a variety of reasons (see [Johnstone and Titterington, 2009] or [Steinbach et al., 2003]): they might struggle when the number of samples is relatively small compared to the dimension (risk of overfitting), their computational cost might scale poorly with the dimension of the input (as is the case for Support Vector Machines, see [Bishop, 2006]), distance-based methods face counter-intuitive phenomena[1], etc.

Dimensionality reduction techniques can be applied independently from any downstream task, in which case the aim is to get a more compact representation of the data that preserves as much information as possible, though one cannot hope for a universally good representation: as an example, a dataset might have very little variance along one of the dimensions of the input space, yet that dimension might have the most predictive power with regard to some dependent variable that we would like to predict. Most dimensionality reduction methods, such as Principal Component Analysis (PCA, see [Jolli□e, 2002]), would flatten the data along that dimension, hence losing crucial information. Some techniques can also be applied in conjunction with a downstream task to insure that the

[1] As an illustration, it is easy to see that the mass of a normal multivariate distribution in dimension n is concentrated (when n is large enough) close to a sphere of radius proportional to $\sqrt{n}$. This means that two such distributions whose means are far apart can nonetheless have common regions of high probability density.

preserved information is that which is needed for said task (e.g., training an autoencoder with a weighted sum of a reconstruction loss and the loss of the downstream task).

Some methods, such as PCA or Autoencoders (see [Kramer, 1991] and below), learn a mapping from the input space to a lower-dimensional encoding space using the training data, which can then be applied to embed new data points. Others do not, such as Locally Linear Embeddings (LLE, see [Roweis and Saul, 2000]), Multidimensional Scaling (MDS, see [Kruskal, 1964]) and its refinement Isomap (see [Tenenbaum et al., 2000]), or t-distributed Stochastic Neighbor Embedding (t-SNE, see [van der Maaten and Hinton, 2008]): out-of-sample points cannot be encoded. See Table 1.1 for a partial list of dimensionality reduction algorithms, with a short description of their core idea and whether they learn mappings.

Many of these methods can be described as *manifold learning techniques*, i.e. that they work on the assumption that the data has a particular structure - more specifically, that it lives (up to some noise) on some low-dimensional manifold in the high-dimensional feature space (see Section 2.1 for more on manifolds, and [Melas-Kyriazi, 2020] for a general reference on the theory of manifold learning). Though the line between manifold learning and non-manifold learning dimensionality reduction algorithms is not set in stone (as most methods can be applied to any dataset), manifold learning methods often explicitly aim to make use of some local structure of the data and can thus be expected to perform better when the mass of the underlying distribution is indeed concentrated on a manifold of sorts. This is for examples the case for Principal Manifold (as the name suggests, see [Gorban et al., 2008]), or Isomap, where the local structure of the manifold is meant to be captured through the neighborhood graph.

All the methods mentioned above have limitations: for example, PCA is quite crude and misses any non-linear information (though kernel PCA does not), MDS relies on Euclidean distances instead of taking into account the intrinsic geometry of the data, Isomap does take the structure into account, but is very sensitive to hyperparameter selection and lacks robustness (the “short-circuit" problem, see [Balasubramanian and Schwartz, 2002]), which is also true of LLE (as they both rely on a k-nearest neighbours subroutine), etc. Additionally, all methods that do not learn a mapping cannot be cross-validated, as there is no mapping to be applied to the cross-validation set[2], which complicates hyperparameters selection and performance evaluation.

[2]Unless the algorithm was trained on the union of the training set and the validation set, which goes against the very principle of cross-validation.

A few dimensionality reduction algorithms			
Name	Reference	Core idea	Learns a mapping
PCA	[Jolli□e, 2002]	Project the data onto the axes of greatest variance.	✓
Kernel PCA	[Schölkopf et al., 1998]	Apply PCA to non-linear features of the data using a kernel trick	✓
Autoencoder	[Kramer, 1991]	Train a neural network to encode the data in low-dimension, then reconstruct it	✓
LLE	[Roweis and Saul, 2000]	Find an embedding that aims to preserve local linear relations between points	7
MDS	[Kruskal, 1964]	Find an embedding that aims to preserve (renormalized) Euclidean distances between points	7
Isomap	[Tenenbaum et al., 2000]	Find an embedding that aims to preserve (renormalized) distances between points computed using a neighborhood graph	7
t-SNE	[van der Maaten and Hinton, 2008]	Find an embedding that aims to preserve a function of the Euclidean distances between points using a probabilistic framework	7
Principal Manifold	[Gorban et al., 2008]	Approximate the data set with a parameterized manifold and project the data points onto it	∼

Figure 1.1 – A few dimensionality reduction algorithms, with either the original reference or a more accessible and up-to-date one. We also briefly summarize their core idea, and indicate whether they learn a mapping that can be used to encode out-of-sample points (the ∼ symbol indicates that learning a mapping can be more or less practical depending on the exact implementation of Principal Manifold).

In this thesis, we are particularly interested in autoencoders, a relatively old idea (see [Kramer, 1991]) that has gained a lot of traction in the last ten years due to the popularization of deep learning (see [Hinton and Salakhutdinov, 2006]; note also that they are not even mentioned in this survey [Sorzano et al., 2014] on dimensionality reduction dating from 2014). Fundamentally, an autoencoder is composed of two neural networks, an encoder and a decoder. The encoder maps high-dimensional data to a low-dimensional encoding (or latent representation), which the decoder uses to reconstruct as faithfully as possible the initial input, which forces the autoencoder to preserve crucial information in the encoding. There exist many refinements of this basic idea (probabilistic approaches,

regularization mechanisms, etc. - see [Bank et al., 2020] for a survey).

Autoencoders combine many qualities (they learn a mapping, can be trained conjointly with a downstream task, etc.) and are usually perceived as a flexible, all-purpose dimensionality reduction technique, well-suited to most data and tasks, including manifold learning (see e.g. [Vincent et al., 2008]). In what follows, we question that assumption and analyze a limitation of autoencoders, which is in fact shared by most dimensionality reduction techniques that learn a mapping: the fact that this mapping is usually **continuous**. We explore the consequences of it in Chapter 2.

Chapter 2

Limitations of continuous dimensionality reduction algorithms

In this chapter, we give a formal definition of manifolds and show that there is a theoretical limit on how well all dimensionality reduction techniques that apply continuous mappings to the data, among which autoencoders, can hope to encode closed manifolds.

2.1 Theoretical aspects

An n-dimensional *topological manifold* is a topological space that is locally similar to $\mathbb{R}^n$ (see for example [Munkres, 2000] for an introduction to basic topology). More formally:

Definition 2.1.1. *A completely separable and Hausdor [1] topological space M is a topological manifold of dimension n if every point $x \in M$ admits an open neighborhood that is homeomorphic to an open subset of $\mathbb{R}^n$.*

Consider a problem in which n-dimensional data comes from a distribution whose mass is concentrated on a d-dimensional manifold $M \subset \mathbb{R}^n$, with $d << n$. We choose to focus on those dimensionality reduction techniques that aim to learn from the data a map $f : \mathbb{R}^n \to \mathbb{R}^m$, where $m < n$, such that the restriction $f|_M : M \to \mathbb{R}^m$ yields a representation as "faithful" as possible. What is meant by faithfulness depends on the situation and downstream task, but it usually involves preserving part of the structure of M: the images of points that are close in M (according to some reasonable distance on M, which need not coincide with the Euclidean distance on $\mathbb{R}^n$) should also be close in $\mathbb{R}^m$ and vice-versa. In particular, $f|_M$ usually needs to be "as injective" as possible.

There is a trade-o between encoding as much information as possible, and obtaining a compact, low-dimensional representation. Image[2] dimension m can be smaller than d,

[1] Those two conditions are automatically satisfied for any subspace of the finite-dimensional Euclidean space.

[2] In the sense of mathematical image, not picture.

in which case some information is necessarily lost, or greater, in which case the image $f(M)$ will generically be a submanifold of measure 0 of $\mathbb{R}^m$. In the case of an unknown downstream task, the ideal scenario would be for m to be equal to d and for $f|_M : M \to \mathbb{R}^d$ to be a homeomorphism, i.e. a bijective, bicontinuous application from M to $\mathbb{R}^d$ - such a mapping would capture all, and only, the relevant information. As is discussed below, we generally cannot hope for such a mapping to exist. Autoencoders also learn a decoding function $g : \mathbb{R}^m \to \mathbb{R}^n$, which is meant to be a parametrization of sorts of M; the composition $g \circ f|_M : M \to \mathbb{R}^n$ is usually constrained to be close to the inclusion $i_M : M \hookrightarrow \mathbb{R}^n$ for some measure of similitude (such as the average square distance $|g \circ f|_M(x) - i_M(x)|^2$ over all sample points x).

As mentioned in Chapter 1, the encodings f learnt by most manifold learning techniques (in particular autoencoders) are continuous. While this may seem like a desirable quality, it also has negative consequences: we claim that under mild hypotheses, it is impossible to injectively and continuously map the d-dimensional manifold M to $\mathbb{R}^d$, a fact that was alluded to but neither explained nor proved in [Batson et al., 2021]. This can be easily visualized in simple cases: it is impossible to embed the circle in the Euclidean line or the sphere in the Euclidean plane without “flattening" or “folding" them (see Figure 2.1).

It is possible to show that this is a more general phenomenon. Assume first that M is connected, compact and without boundary. As the n-th homology group of $\mathbb{R}^n$ is trivial, the degree of the application induced in homology by $f|_M : M \to \mathbb{R}^n$ is trivial (see [Hatcher, 2005] for a general introduction to algebraic topology, and [Outerelo and Ruiz, 2009] for more on the degree of continuous maps in particular).

With most techniques (such as neural networks with usual activation functions), the set $\Sigma \subset M$ of points on the neighborhood of which the encoding f is not smooth[3] is of measure 0 in M; consequently, $f(M)$ is also of measure 0. Consider now $f|_{M \setminus \Sigma} : M \setminus \Sigma \to \mathbb{R}^n$: it is a smooth application. According to Sard’s theorem (see [Michor et al., 2008, Chapter 1]), the set of singular values of $f|_{M \setminus \Sigma}$ is of measure 0 in $\mathbb{R}^n$. Hence all points y of $f(M)$ except a subset of measure 0 admit an open neighborhood U such that $f|_M$ is a smooth local di□eomorphism on $f^{-1}(U)$. For such a $y \in f(M)$ and using the definition of the degree in terms of local $\mathbb{Z}_2$ orientations (as is explained in [Outerelo and Ruiz, 2009]), the cardinality of the preimage $f^{-1}(y) \neq \varnothing$ must be equal to the degree of f modulo 2, i.e. 0: in other words, y has a non-zero and even number of preimages. This means that $f|_M : M \to \mathbb{R}^n$ is dramatically non-injective: either $f(M)$ is itself of measure 0 (which means that f is crushing M into something of a smaller dimension), or most points of $f(M)$ are such that they have at least two preimages.

If M is not compact or has boundaries, things are not so clear-cut: for example, a two dimensional closed ball (i.e. a disk) can be embedded in the Euclidean plane, but a

[3] One can also replace f by an arbitrary close smooth function $\tilde{f} : M \to \mathbb{R}^n$ for the rest of the discussion, as smooth functions from M to $\mathbb{R}^n$ are dense in the space of continuous functions from M to $\mathbb{R}^n$.

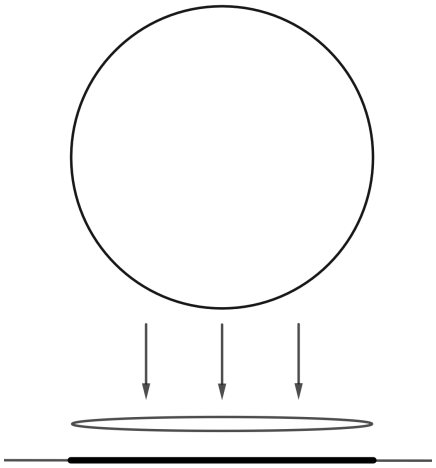


Figure 2.1 – It is impossible to continuously embed a circle into $\mathbb{R}$ - the mapping will necessarily be non-injective.

torus from which an open ball has been removed cannot. The only theoretical guarantee in general is that one can always embed a d-dimensional manifold in $\mathbb{R}^m$ for $m \geqslant 2d$, as stated by the strong Whitney embedding Theorem (see [Michor et al., 2008]). This bound is essentially sharp[4].

In real life applications, the data rarely live on a true topological manifold, though it can for example be the case for data from physics (as in [Batson et al., 2021]) or computer vision (as in [Lee et al., 2004]); it is more common to see "manifold-like" sets, similar to manifolds in some regards but less regular (due to noise, self-intersections, discontinuities, etc.). It would in fact be an interesting research question to first somewhat formalize this notion of "manifold-like", then to try to measure how "manifold-like" typical data is in various fields of application. Nonetheless, the discussion above suggests that one should not expect to always be able to embed the data into a space corresponding to its true dimensionality, or even of a slightly higher dimension. In the case of an autoencoder, similar arguments can be made regarding the decoding function g to show that it can generally not be onto M without being non-injective.

In practice, this problem can be partially overcome by existing methods: M can for example be embedded in a space of dimension $m \geqslant 2d$, though it has the disadvantage of making the encoding less compact, and most of the points in the encoding space do not correspond to points of M. One could also hope to obtain "almost everywhere injective" embeddings. Take the example of the sphere: it cannot be embedded in $\mathbb{R}^2$, but the sphere from which one of the pole has been removed can, as it is homeomorphic to a disk[5].

[4] In the sense that for any $k \in \mathbb{N}$, there exists a manifold M of dimension $d = 2^k$ (the real projective space) that cannot be embedded in $\mathbb{R}^{2d-1}$. When d is not a power of 2, slightly better bounds can be found.

[5] Hence why the documentation of the scikit-learn library [Pedregosa et al., 2011] demonstrates its man-

Similarly, for any d-dimensional manifold M, there are subsets of M of arbitrarily small measure such that their complement in M can be embedded in $\mathbb{R}^d$; an autoencoder should be able to learn such an embedding, that would be faithful on most of M except for a small subset (which it would typically map to wildly nonsensical values). It is however unclear how one should train the autoencoder to do so.

Note that for simplicity, we have focused on purely topological questions; equally important are more subtle problems related to metrics, as we would usually like some notions of distance between points to be approximately preserved by the encoding.

2.2 Experimental illustration

In this section, we run a few qualitative experiments and give illustrations of the phenomena described in Section 2.1. We consider a dataset of points densely covering the surface of a 2-dimensional sphere, without any noise added, and apply various dimensionality reduction techniques to it. More quantitative and systematic experiments are described in Chapter 7.

In Figure 2.2, we show an encoding obtained using Isomap (the scikit-learn implementation, see [Pedregosa et al., 2011]), one of the dimensionality reduction techniques mentioned in Chapter 1 which aims to preserve distances (as computed using a neighborhood graph) between points. At the top, the original data points as well as their encodings in the plane are represented together. The encodings are colored, and the original data points are given the same color as their encodings, so as to make the mapping visually apparent. We see that Isomap flattens the sphere into a disk (along a vertical plane of symmetry of the sphere), then embeds it very regularly into the plane. As a result, almost all encoded points have two preimages, one in each of two halves of the sphere. This is illustrated at the bottom of the figure, where the points of those two halves are represented, along with a black segment that connects them to their embedding in the plane. Each of the two halves have roughly the same image.

Other classical methods (also their scikit-learn implementations) perform in di□erent ways; for example, MDS produces an encoding similar to that of Isomap, while Spectral Embedding and LLE fail to run (due to the problem being too singular) - this is not entirely surprising, as they aim to embed the sphere in $\mathbb{R}^2$ (by respecting the structure of small neighborhoods), which is impossible.

We also trained a simple autoencoder[6]. We can see at the top left of Figure 2.3 that the mapping it learns is much less regular than the one applied by Isomap. Analyzing it

ifold learning techniques on a severed sphere rather a complete one.

[6] Its encoder was composed of two fully connected layers with 10 units each and $tanh$ activation functions and a final linear layer with 2 units; its decoder was also composed of two fully connected layers with 10 units each and $tanh$ activation functions and a linear output layer with 3 units. The loss function was the square distance between the original point and its reconstruction by the autoencoder.

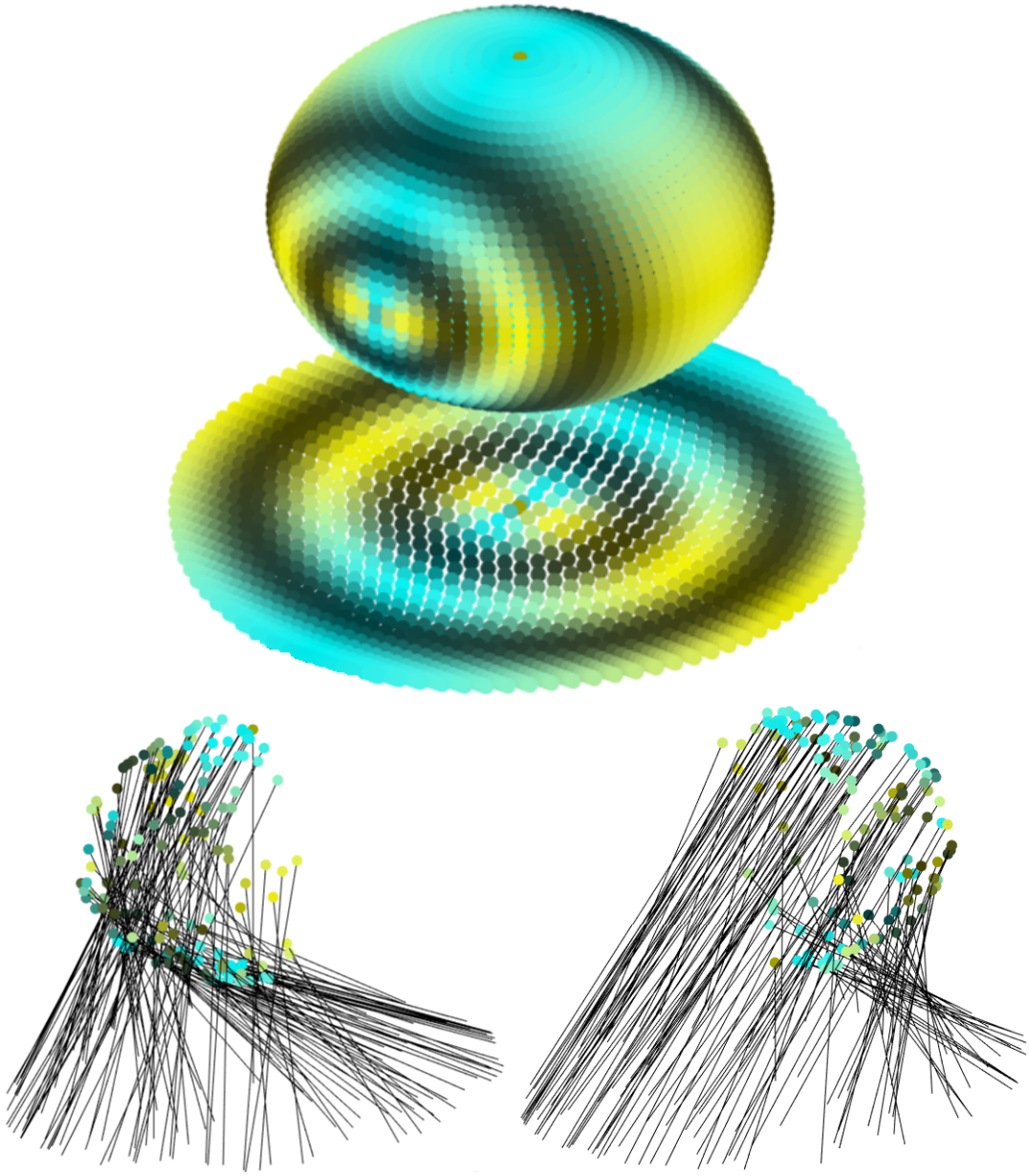


Figure 2.2 – Isomap encoding of the 2-dimensional sphere. At the top, the original points and their encodings in the plane are represented; each point has the same color as its image to help visualise the mapping. Below, the sphere is split into two halves, and each point is linked to its image by a segment. We see that the two halves are mapped to the same image - the mapping is very non-injective.

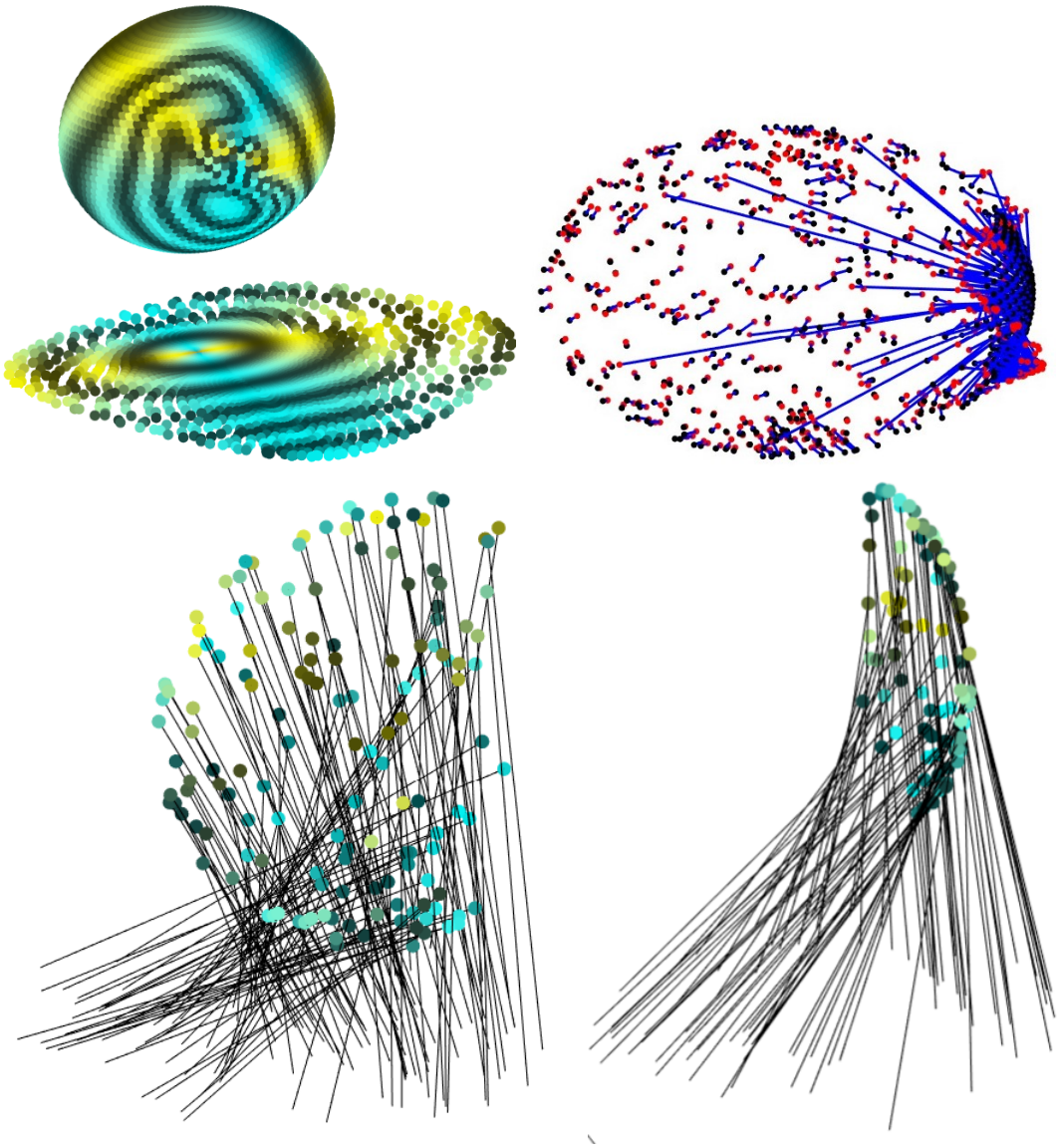


Figure 2.3 – Encoding of the 2-dimensional sphere by an autoencoder. At the top left, the points are represented jointly with their encodings in the plane. At the top right, the initial points (in blue) are connected to their reconstructions (in red) by black segments. We see that most of the sphere is correctly reconstructed, except a small cap on the right. Below (where each point is linked to its image), we see that the image of this small cap by the mapping (on the right) is the same as the image of the rest of the sphere (on the left): the mapping is very non-injective.

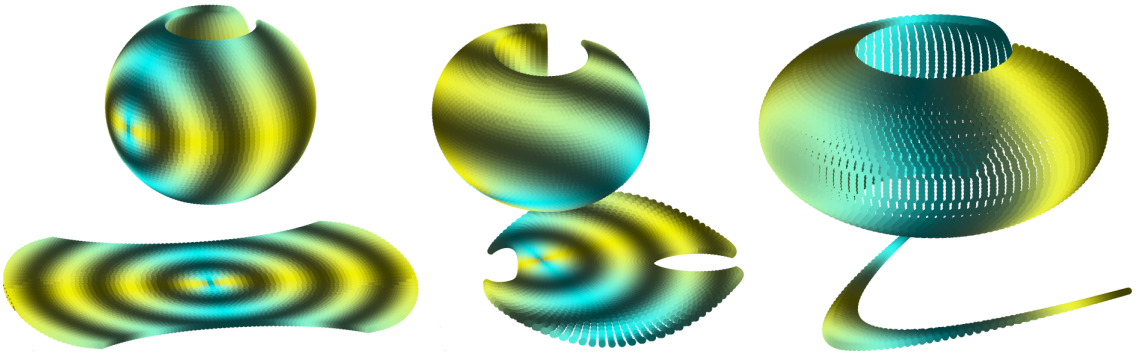


Figure 2.4 – 2-dimensional encoding of a severed sphere by Isomap (left), an autoencoder (middle) and Spectral Embedding (right). Each point has the same color as its image, to help visualise the mapping. We see that Isomap and the autoencoder successfully unfold and embed the severed sphere, while Spectral Embedding flattens it vertically in a non-injective way.

further, we see that the encoder actually learnt the "almost everywhere injective" mapping described in Section 2.1: most of the sphere is flattened on the plane in an injective and somewhat regular fashion, except for a small cap, representing less than 10% of its surface, whose image covers the image of the entire rest of the sphere - this is illustrated on the bottom row of Figure 2.3, with the mapping of the small cap on the right and the mapping of the rest of the sphere on the left.
At the top right, initial points (in black) are connected to their reconstruction (in red) by blue segments. We see that the encoding learnt e□ciently stores the information of most points, which allows them to be correctly reconstructed, at the cost of the few points of the cap being mapped to completely erratic reconstructions. The consequences in terms of unsupervised anomaly detection of this type of behaviour are discussed later in the thesis. Some testing showed that deeper autoencoders result in smaller, more concentrated caps (as they are capable of learning more complicated, less regular functions).

By contrast, we consider the sphere from which both poles and a slice connecting them were removed. As it is now homeomorphic to a disk, there is no theoretical hurdle to embedding it in the plane[7]. We see in Figure 2.4 that the autoencoder (on the right) and Isomap do it well, while the Spectral Embedding technique (with the default hyperparameters of the scikit-learn library) still struggles and flattens the punctured sphere vertically in a non-injective fashion.

[7] This would also be the case for a sphere from which a single pole has been removed, but we observed that most methods (except deep autoencoders) still struggle with it, for reasons that vary depending on the technique used.

Part II

Chapter 3

Unsupervised anomaly detection

As explained in the Introduction, we are interested in examining the assumptions underlying autoencoder-based unsupervised anomaly detection algorithms. In this chapter, we define unsupervised anomaly detection and discuss the possible consequences of the use of autoencoders as an anomaly detection system, particularly in light of our findings of Chapter 2.

3.1 Definition and motivation

Unsupervised Anomaly Detection (UAD) is the task of identifying anomalous data points in a set in which they are mixed with normal points, all of them being unlabelled. Anomaly detection problems can arise in a wide range of situations: to protect a system against cyberattacks, as in the pioneering article [Eskin et al., 2002], prevent damage to industrial machines, as in [Martí et al., 2015], detect health conditions, as in [Schlegl et al., 2017], etc. In many cases (as in data science in general), labelled data is either costly or intrinsically difficult to obtain (as noted in [Eskin et al., 2002]), for example when looking for yet unknown genetic anomalies or new computer viruses.

The variety of contexts in which UAD can be performed begets unique challenges; in particular, there is no universal and rigorous definition (from a data science/information theory point of view) of what constitutes an anomaly. What criterion should algorithms use to identify anomalies? In most of the literature (e.g. in the survey [Goldstein and Uchida, 2016]), anomalies are usually loosely described as points

- whose features differ from normal points, usually by being less structured, and
- which are rare compared to normal points.

A simple case is illustrated in Figure 3.1. Many efficient UAD techniques have nonetheless been developed (see the surveys [Goldstein and Uchida, 2016] or [Thudumu et al., 2020]);

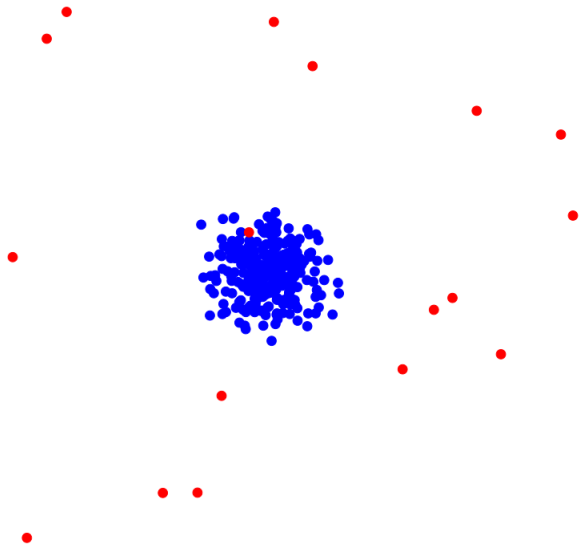


Figure 3.1 – The normal distribution (in blue) is a normal Gaussian, while the anomalies (in red) are more spread out.

we will discuss some of them later in Chapter 4. In the next section, we focus on the autoencoder-based UAD techniques that are our main interest.

3.2 Unsupervised anomaly detection and autoencoders

As discussed in the Introduction, a recent development has been the introduction of deep learning techniques in UAD (see [Chalapathy and Chawla, 2019] for a survey), which have led to successive improvements of the state of the art - see [Sakurada and Yairi, 2014], [Zong et al., 2018], [Bergman and Hoshen, 2020] and [Yoon et al., 2021]. Though each new algorithm introduces its own subtle refinements[1], the basic premise is often the same: an autoencoder composed of neural networks learns a representation of the data set, which contains both normal and anomalous unlabelled data.

As anomalous data points are expected to be both less numerous and more “wild" than the normal ones, they should be less well reconstructed by the autoencoder, which should have learnt the structure of the normal data. Some function of the reconstruction error (either its norm, or something more ingenious, as in [Zong et al., 2018]) can then be used to detect anomalies. Hence, while a classification algorithm might still be able to correctly classify points using an encoding of poor quality, an autoencoder-based UAD algorithm will always identify a normal but poorly reconstructed point as an anomaly. This makes autoencoder-based UAD methods particularly dependent on the quality of the encodings and reconstructions.

[1] We also discuss the fundamental nature of most of those refinements in Chapter 4.

This class of techniques rests on two important (and often implicit) assumptions. The first is that autoencoders are a suitably flexible tool, capable (assuming the autoencoder is powerful enough) of faithfully reconstructing points similar to those in its training set; it is also expected that regions with a high density of training points will be more precisely encoded than those with a low density. The second assumption, which is explicitly made in [Xu et al., 2018] or [Eskin et al., 2002] but always implicitly present, is that normal points are found in high-density areas of the underlying distribution, while anomalies are in low-density ones; in other words, that anomalies are statistical outliers. We find both those assumptions overly optimistic and simplistic - we discuss the first one here below, and the second one in Chapter 4, in the context of a more general reflection on the nature of anomalies.

Though autoencoders have many qualities, autoencoder-based UAD algorithms can suffer from their continuity (and even smoothness) in two ways:

The first one, which is our main focus, is a consequence of what has been exposed in Section 2.1. As autoencoder-based UAD algorithms use the autoencoder's reconstruction error to identify anomalies, they are dependent on the quality of the encoding of normal points. We have seen that continuous autoencoders cannot hope to learn perfect encodings of the entirety of closed manifolds without boundaries, and are likely to struggle with many "manifold-like" structures. This means that if the normal points of a UAD problem live on such a structure, and are distributed densely enough that there can be no "hole" in the manifold that would make it topologically simpler (and thus possible to embed), some of those normal points will necessarily be poorly reconstructed (as was shown in the case of the sphere in Figure 2.3 in Section 2.2), and are then likely to be falsely categorized as anomalies. This phenomenon was illustrated at lengths on various data sets from particle physics by the authors of [Batson et al., 2021] (though they gave no theoretical explanation for it).

Note that increasing the dimension of the latent space so that it is large enough for the manifold to be embedded is not a good solution: though it should allow the system to faithfully reconstruct the normal points, it also means that more information than strictly necessary is being encoded, which increases the risk of points outside the normal distribution being correctly reconstructed - including anomalies, which would then not be detected.

The second way (which was also illustrated in [Batson et al., 2021]) in which the smoothness of autoencoders can do a disservice to UAD algorithms could be described as "overzealous continuous extrapolation": it arises when the autoencoder unexpectedly learns how to faithfully encode anomalous points despite them being very different from the normal points that made up most of the training set, due to them lying in some smooth prolongation of the areas of high density where normal points lie. As an example, consider Figure 3.2, where a toy data set in $\mathbb{R}^2$ comprised of normal points (in blue) and anomalous

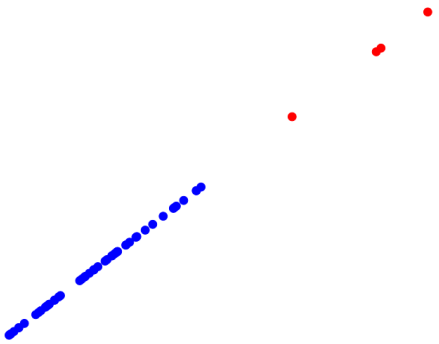


Figure 3.2 – A simple anomaly detection data set. The blue points are normal, while the red points are anomalies. The anomalies and the normal points all live on the same submanifold of the ambient space (a line).

data points (in red) is represented. Though the normal points are easily distinguishable from the anomalies, they all lie on the same line. The easiest 1-dimensional representation of this set is the linear projection of the plane onto that line - an autoencoder that has learnt such a representation would be capable of faithfully reconstructing any point on the line, even if it is very distant from the training points - hence including the anomalies.

Chapter 4

Anomalies and structure

4.1 Introduction

We feel that while many clever UAD algorithms have been developed, the more theoretical aspects of the task have been somewhat neglected - specifically, there seems to be a lack of reflection regarding what anomalies truly are, how their possible definitions are connected to various notions of structure, and how the implicit definition of anomalies underlying various algorithms could make them more or less suited to given data sets. For example, we discussed in Section 3.2 how autoencoder-based UAD techniques expect anomalies to be outliers of the total data distribution.

Analyzing such unstated assumptions (e.g. in the spirit of [Rainforth et al., 2018] or [Curth and van der Schaar, 2021]) can often lead to a better theoretical understanding of research problematics, as well as open the way to concrete improvements to existing algorithms

In this chapter, we aim to partially make up for this lack of reflection. In particular, we discuss the nature of anomalies in Section 4.2 through a series of archetypal examples and identify three levels of structure on which anomalies can di□er from normal data. We also explore the assumptions underlying most methods currently used and their consequences in Section 4.3.

4.2 What is an anomaly?

Ultimately, the definition of what constitutes an anomaly is task-dependant: an anomaly is what we categorize as such in a given context. Accordingly, semi-supervised learning might be the most natural framework in which to work: an end user classifies a small fraction of the data as either anomalies or normal points, which then serves as a guideline for the classification of the rest of the dataset[1].

[1] This, in turns, raises interesting questions in terms of interactive machine learning: given a dataset, which points should the algorithm ask the user to label in order to maximise the quality of the subsequent

However, human intervention is not always possible, either because it is too costly or because humans experts themselves cannot identify anomalies due to the nature and complexity of the data (genomics, cybersecurity, etc.). Hence the demand for UAD algorithms, which in turn must necessarily rely on some aspects of the data to classify them as either normal or anomalous.

Ideally, we would like to come up with a working definition of “anomaly" that satisfyingly captures those aspects. As tasks are designed by humans and require to identify what human users think of as anomalies, a definition should preferably also align with the human intuition of what constitutes an anomaly. However, even an informal definition proves to be hard to formulate. Most articles either do not define what an anomaly is at all and assume it to be common knowledge (including good papers that introduce very e□cient techniques, such as [Bergman and Hoshen, 2020]), or rely on overly vague statements such as those mentioned in Chapter 3 (see e.g. [Goldstein and Uchida, 2016] or [Chalapathy and Chawla, 2019]): anomalies

- di□er from normal points in their features, and
- are few in numbers compared to normal points.

Defining what is meant by “di□er" is a non-trivial question that has important practical consequences (as discussed in Section 4.3). For example, it is often assumed that normal points have structure, while anomalous points are more wild, i.e. noisy, spread out, etc. There are many cases in which such assumptions fail. Consider the case of medical data: while some anomalies are likely not to follow any fixed pattern, e.g. injuries resulting from violent accidents, other anomalies, e.g. cancers, will be structured and follow patterns, even if these patterns will be di□erent from those in normal data.

Though we cannot hope for a perfect definition of anomality, a higher level of precision than “they di□er from the norm" should be achievable. In this spirit, we identify three levels on which data in the Euclidean space[2] can behave anomalously compared to the rest of a distribution. Let p_a be the distribution of the anomalies, p_n the distribution of the normal points, and p_t the distribution of the total data set.

- Probabilistic: p_a can di□er from p_n in ways that are purely probabilistic - in other words, they can be formulated purely in terms of probability spaces, without taking into consideration the topology, the metricity or the di□erential structure of the ambient space $\mathbb{R}^n$. The simplest example is for the anomalies to be outliers for the distribution p_n (and hence also for p_t, as there are few anomalies) - this is in fact the criterion used to define anomalies in some articles (e.g. [Zong et al., 2018]).

classification?

[2] Parts of these reflections also apply to categorical data.

- Metric: anomalies can differ from normal points by their position in space, for example by being very far for the Euclidean distance from most of the normal points, or by having a very different variance.

- Continuous/differential: anomalies can differ from normal points in subtler ways, related to continuous or even differential structures. Example: all normal points lie on a smooth manifold while anomalies do not, even though they are sufficiently close to the manifold that metric methods would have a hard time identifying them as anomalous.
  This type of anomality is the hardest to capture using formal criteria, though not necessarily the hardest to handle in practice (see Section 4.3).

The percentage of anomalies in the total data set also plays an important role, as showcased below, which further complicates the picture.

4.2.1 Menagerie of examples

Though this taxonomy of anomalies is not perfect, we prove its usefulness by applying it to the analysis of a few schematic data sets[3] that allow us to demonstrate the difficulty of formulating a single, unifying definition of “anomaly".

- The simplest, least ambiguous example might be that in which the normal points follow a simple, relatively concentrated distribution (for example a low-variance Gaussian), while the anomalies follow an extremely spread out one (for example a very high-variance Gaussian). This is illustrated in Figure 4.1.
  Note that we do not mean that each anomalous point is necessarily easy to identify as such (as some might fall in the middle of normal points), but rather that anomalies, as a group, behave very distinctly from normal points. Not only are most anomalies outliers for the normal distribution, but they also tend to be quite distant from most normal points: they differ both on a probabilistic and a metric level.

[3] We do not pretend that all of these situations are equally likely to appear in real data sets - they nonetheless constitute interesting thought experiments.

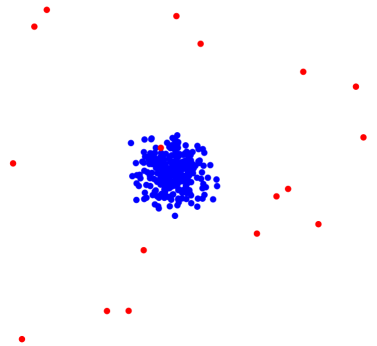


Figure 4.1 – The normal distribution (in blue) is a Gaussian, while the anomalies (in red) are more spread out.

- Consider now the case where the normal points follow a normal Gaussian distribution while the anomalous distribution p_a is another, much more concentrated Gaussian distribution, as in Figure 4.2 (this could for example represent measurements from a sensor that sometimes dysfunctions and returns 0 readings). It illustrates two difficulties. The first is the impact of the proportion of anomalies in the total distribution on the definition of what constitutes an anomaly, particularly when some points form *micro clusters* away from the others (this terminology is used e.g. by [Goldstein and Uchida, 2016]), as the red points do here. If there are only a few of them, they can be safely categorized as anomalous, but if they represent 30% of the data set, they cannot (without additional information) be unambiguously declared to be anomalies rather than another regular subset of the normal distribution.
  We can also see here the limits of purely probabilistic definitions of anomality. If the percentage of anomalies is low (which allows us to unambiguously define them as anomalies), but not too low, and if p_a is sufficiently concentrated, the area where the anomalies are found will have a higher density for the total distribution p_t; in other words, anomalies will tend to be LESS “outlierish" than normal points. On the other hand, metric criteria can easily identify anomalies as such here.

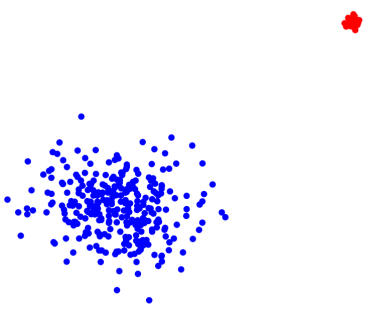


Figure 4.2 – The normal distribution (in blue) is a Gaussian and the anomalous distribution (in red) is another, more concentrated Gaussian.

- Even more tricky is the case[4] of a normal Gaussian, and another, extremely concentrated Gaussian with the same mean (almost a Dirac), as in Figure 4.3. Not only are the anomalous points less “outlierish" than normal points rather than more, as before (note that this could also be used as a criterion), but metric methods will also struggle to distinguish them from normal points.
  It provides another illustration of the importance of the percentage of anomalies. Should it be very low and the data set relatively small, the fact that some points do not follow the normal distribution would be almost impossible to detect, while it would become very apparent once the percentage becomes high enough (which does not mean that a given anomaly would become easy to distinguish from a normal point). Interestingly, the more concentrated the anomalous distribution and the easier the problem (as the over-concentration of points near the mean would stand out more).

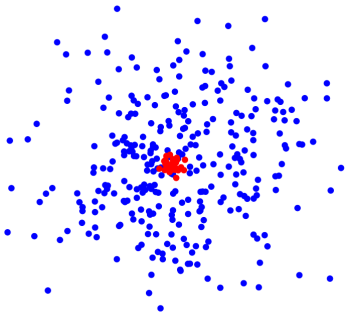


Figure 4.3 – The normal distribution (in blue) is a Gaussian and the anomalous distribution (in red) is another Gaussian with the same mean but much lower variance.

- Figure 4.4 represents perhaps the most problematic and subtle of the configurations considered here. The normal distribution is a mixture of Gaussians. The anomalous points (which constitute about 10% of points here) are distributed according to a Gaussian, distinct from any Gaussian that makes up the mixture, but similar to them. In a purely unsupervised setting, this problem is impossible, as there is no a priori way of telling which of the Gaussians is the anomalous one - yet the anomalous distribution is very di□erent from the normal one both in probabilistic and metric terms (since the anomalies are all rather clustered, and fairly distant from normal points), which shows that such di□erences in behaviour are not enough to make a problem easy.
  The key element here is that while the anomalous distribution p_a is very di□erent from the normal one p_n, the total distribution p_t is very similar in nature to p_n, essentially because “p_a locally looks like p_n". We have found this intuition di□cult to formalize.

[4] One could for example imagine that some sensors sometimes dysfunction and return 0 readings.

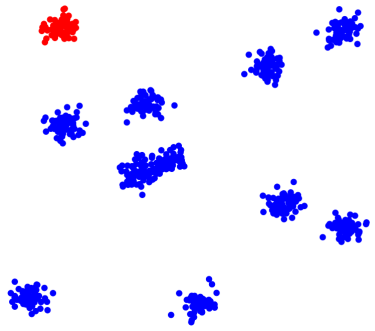


Figure 4.4 – The normal distribution (in blue) is mixture of Gaussians and the anomalous distribution (in red) is another Gaussian, distinct from but similar to those in the mixture.

- In Figure 4.5, we see the importance of more continuous or even smooth structures, such as manifolds: what clearly (for a human observer) distinguishes the red points from the rest is that while they are close to some normal points, they are not on the same line, which can be hard to capture for purely distance-based algorithms.

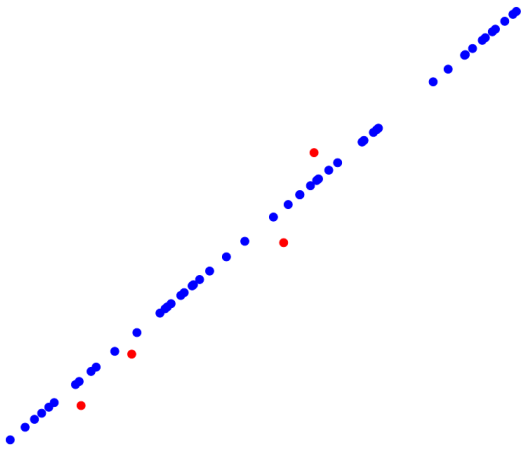


Figure 4.5 – The normal points in blue all lie on a line, while the anomalies in red do not.

- Conversely, anomality could be characterized by an excess of geometrical structure (for example in a situation where anomalies are meaningful events among noisy background activity), as in Figure 4.6, where most points follow a non-degenerate two dimensional Gaussian distribution, while a small subset of them lie on a line.

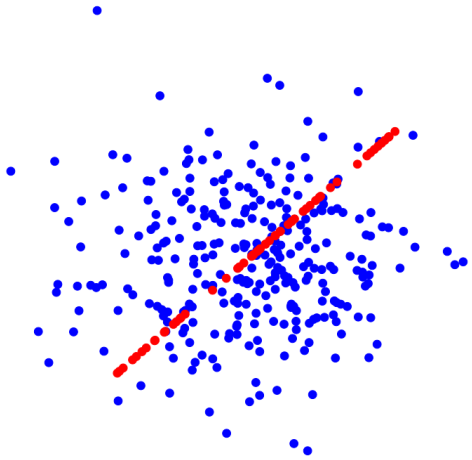


Figure 4.6 – The normal points in blue are sampled from a non-degenerate Gaussian distribution, while the anomalies in red all lie on a line.

- Finally, one can see in Figure 4.7 a case in which the anomalies and the normal points follow the same geometric constraints (they all lie on a line), but different distributions within those constraints (similar to the data sets described in [Batson et al., 2021]); here, distance-based methods might be more efficient than those that capitalize on smoother structures (more on this in the next section).

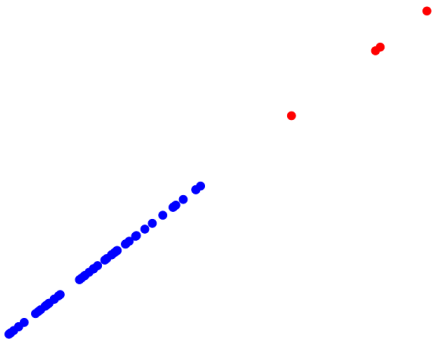


Figure 4.7 – The normal points in blue and the anomalies in red are concentrated in different regions of the line.

Observe that while we struggle to find a unifying definition of “anomaly", an untrained human would easily and unambiguously identify the anomalous points in most of these cases, which suggests that such a definition should be possible.

Though the artificial examples examined above represent reasonable situations that are likely to exist in the real world, it would be valuable to measure the prevalence of each type of anomalies in typical data sets of various fields of application.

4.3 How do we identify anomalies?

In this section, we discuss the criteria explicitly or implicitly used by a few existing UAD algorithms to identify anomalies, with a particular emphasis put on encoder-based techniques, in connection to the rest of this thesis.

Following the taxonomy which we developed in Section 4.2, many classical UAD algorithms (see [Goldstein and Uchida, 2016] for a survey) were based on metric criteria to identify anomalies: for example, the k-nearest neighbour global anomaly score (see [Ramaswamy et al., 2000]) uses the distance between a point and its k-th nearest neighbour as a measure of anomality. Others took a more probabilistic approach: for example, Cluster-Based Local Outlier Factor (CBLOF, see [He et al., 2003]) clusters the data (using the Euclidean distance); the clustering is then used to make distribution density estimations. Conversely, one-class Support Vector Machine (SVM) methods adapted to UAD (see [Amer et al., 2013], in which a one-class SVM is used to separate the image of the entire data set from the origin in a kernel space (using a kernel trick) relies almost entirely on continuous/smooth criteria dictated by the choice of kernel function, as the implicit feature map needs not to respect Euclidean distances and no estimation of density is explicitly carried out. In Table 4.8, we give a rough classification of some well-known methods according to which criteria (probabilistic, metric or continous/smooth) they use to distinguish normal points from anomalies.

A few UAD algorithms				
Name	Reference	Probabilistic	Metric	Continuous/ Smooth
Autoencoder-based techniques	[Chalapathy and Chawla, 2019]	✓	7	✓
k-nearest neighbours global anomaly score	[Ramaswamy et al., 2000]	7	✓	7
LOF	[Breunig et al., 2000]	✓	✓	7
CBLOF	[He et al., 2003]	✓	✓	7
HBOS	[Goldstein and Dengel, 2012]	✓	7	7
One-class SVM for UAD	[Amer et al., 2013]	7	7	✓

Figure 4.8 – A few UAD algorithms or family of algorithms, with either the original reference or a more accessible and up-to-date one. We indicate which criteria of the taxonomy introduced in Section 4.2 - probabilistic, metric, continous/smooth - the method (primarily) uses to identify anomalies.

We choose to focus on the more recently developed deep learning-based techniques that have enjoyed considerable success (see for example [Chalapathy and Chawla, 2019]). As explained in Section 3.2, most of these algorithms train an autoencoder to reconstruct the training data, and use some function of the reconstruction error as an anomaly score

- the assumption being that normal points, being prevalent in the training data, will be better reconstructed than anomalies, as the encoding and decoding will be of better quality in regions where many training points can be found. Of course, various refinements can be brought to that basic idea, such as the use of more sophisticated neural network architectures (Convolutional Neural Networks, Long Short-Term Memory, etc.).

Though it is usually not formulated in those terms, learning the anomaly scored obtained as a function of the reconstruction error can be thought of as somewhat equivalent to learning the (unnormalized) density function of a distribution that models the training set, except that the reconstruction error is a decreasing function of that density: the reconstruction error will be low in regions of high density, and high in regions of low density. More specifically, we are (indirectly) learning some distribution that fits the data *under some regularity constraints* that are induced by the architecture chosen and the details of the algorithm. If we go back to the taxonomy introduced in Section 4.2, we see that the assumption underlying those techniques is that anomalies can be identified thanks to a mix of probabilistic criteria (anomalies are outliers, found in regions of low density) and smooth/geometric criteria (induced by the regularity constraints).

As discussed previously in our analysis of Figures 4.2 and 4.3, learning a distribution (or something equivalent to a distribution, such as the reconstruction error) and treating outliers as anomalies is not a sure-fire strategy: a su□ciently powerful autoencoder will be able learn good encodings of small, concentrated clusters of anomalies, and will fail to correctly identify them as anomalies. This should also apply to other methods relying on density estimations, such as CBLOF.

There are also consequences to the ways regularity constraints are practically implemented in those algorithms. They stem from basic architecture choices, such as the width, depth and activation functions of the neural networks used, but also from more subtle manipulations. A good example is that of the Self-Trained One-class Classification algorithm presented in [Yoon et al., 2021], which achieved excellent performances. The core idea is to iteratively refine the training set: an unsupervised anomaly detector (based on some version of the “autoencoder/reconstruction error as anomaly score" idea) is trained, then used to remove from the training set points that are suspected of being anomalous. A new detector is trained on the refined data set, which is then used to further refine it, etc. The basic ingredient is always the reconstruction error: all that this clever iterative procedure does is impose *some kind* of regularity constraint on the final classification.

These regularity constraints are crucial: without them, a model powerful enough would be able to perfectly encode and decode a given data set, normal points and anomalies alike, resulting in abysmally poor performances. The issue is that the way those constraints are brought about is almost entirely implicit: they depend on the network architecture and on clever yet unprincipled design choices and “tricks", such as those in [Yoon et al., 2021] or [Zong et al., 2018]; there is no real justification, besides somewhat convincing heuristic

arguments, as to how they improve performance. This is beyond the scope of this work, but creating an e□cient unsupervised anomaly detection algorithm that would learn a distribution with an explicit regularity parameter[5] could be an interesting research project.

The particular architecture of autoencoder-based methods also has specific consequences, which we already discussed in Section 3.2: though it is in an unprincipled way, they are naturally suited to learning smooth structures, such as manifolds, as their activation functions are themselves smooth or piecewise smooth. While this can be beneficial, they can be TOO good at it, as they will faithfully reconstruct anomalies that are well-structured, particularly if they have the same structure as the normal points (but can be distinguished in other ways) - this was observed on high energy physics data in [Batson et al., 2021]. The second consequence of the continuity of the encoding learnt by classical autoencoders is that they will struggle to encode closed compact manifolds, as was argued at length in Chapter 2.

[5] Parameters are bad, but having no explicit control over the regularity of the distribution learnt is probably worse.

Part III

Chapter 5

Deep Atlas

In this chapter, we present a novel piecewise-continuous autoencoder called Deep Atlas which we hope might avoid the limitations of classical autoencoders discussed in Chapter 2. We start with an exposition of the mathematical intuition behind it.

5.1 A geometric intuition

In Section 2.1, we recalled the definition of an n-dimensional topological manifold M as a space that is locally similar to subsets of $\mathbb{R}^n$ (see [Munkres, 2000] for more on manifolds). An indexed collection $\{U_\alpha, \phi_\alpha\}$ of sets $U_\alpha \subset M$ and maps $\phi_\alpha : U_\alpha \to \mathbb{R}^n$ that certifies this property, i.e. such that $\bigcup_\alpha U_\alpha = M$ and $\phi_\alpha|^{\phi_\alpha(U_\alpha)} : U_\alpha \to \phi_\alpha(U_\alpha)$ is an homeomorphism for all index α, is called an *atlas*[1]. The couples (U_α, ϕ_α) are called *charts*. It is easy to show that any n-dimensional manifold M admits an atlas such that $\phi_\alpha(U_\alpha)$ is the open Euclidian ball in $\mathbb{R}^n$; in other words, it can be covered by simple subspaces homeomorphic to balls.

We think that there is an interesting intuition here, that applies both to proper manifolds and to messier “manifold-like" geometric structures: though they can be overall very complicated, their local structure is much simpler. This is illustrated in Figure 5.1, where the circle is decomposed into subsets homeomorphic to segments.

Hence the idea to build an autoencoder using a collection of smaller autoencoders, each playing the role of a chart: each autoencoder specializes in a small region of the space to be encoded, and a gating network is tasked with sending the input to the appropriate autoencoder. This is described in details in the next section.

There are two things we hope to gain from this architecture. We have seen in Chapter 2 that as neural networks encode continuous maps (at least with the vast majority of activation functions used), a single big neural network simply cannot learn a good encoding of many simple manifolds. By introducing an element of discontinuity through the gating

[1] A similar, though more constraining definition is used in the case of di□erential manifolds.

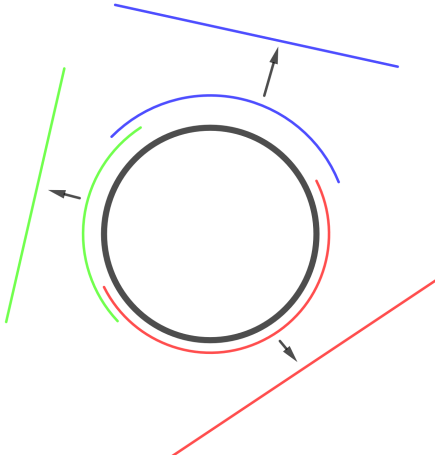


Figure 5.1 – An atlas on the circle.

network, we avoid this limitation.

From a more computational point of view, we suspect that having one big neural network learn an encoding might also be less e□cient than having several smaller networks learn local maps (partially for the same reasons that support the use of ensemble learning, see [Sagi and Rokach, 2018]). Consider a basic example: it is easy to approximately map a segment to a part of the circle with a simple quadratic polynomial, but mapping it to the entire circle is much harder for a typical neural network. It can be done by making the network deeper and wider, but it is computationally expensive and increases the risk of overfitting.

5.2 The main algorithm

We call the autoencoding system inspired by the considerations exposed above a *deep atlas*. As mentioned before, its core components are a gating network G and a collection of autoencoders $\{C_i\}_{i=1}^k$, which we call charts. The gating network is a neural network that takes as input a data point $X \in \mathbb{R}^m$ (for some fixed $m \in \mathbb{N}$) and outputs weights $G(X) = (G(X)_1, \ldots, G(X)_k)$. We typically use a softmax following a linear output layer to obtain $G(X)$. Each chart C_i is an autoencoder, i.e. the combination of a neural network E_i, called the encoder, and another neural network D_i, called the decoder, such that the output dimension n of the encoder is the input dimension of the decoder; it is normally meant to be smaller than m. Various architectures can be chosen for the decoder and the encoder. In what follows, we use simple feedforward neural networks. Likewise, any reasonable activation functions (tanh, ReLu, etc.) can a priori be used, though the output layer of the decoder should be linear (unless there is some known special constraint on the data).

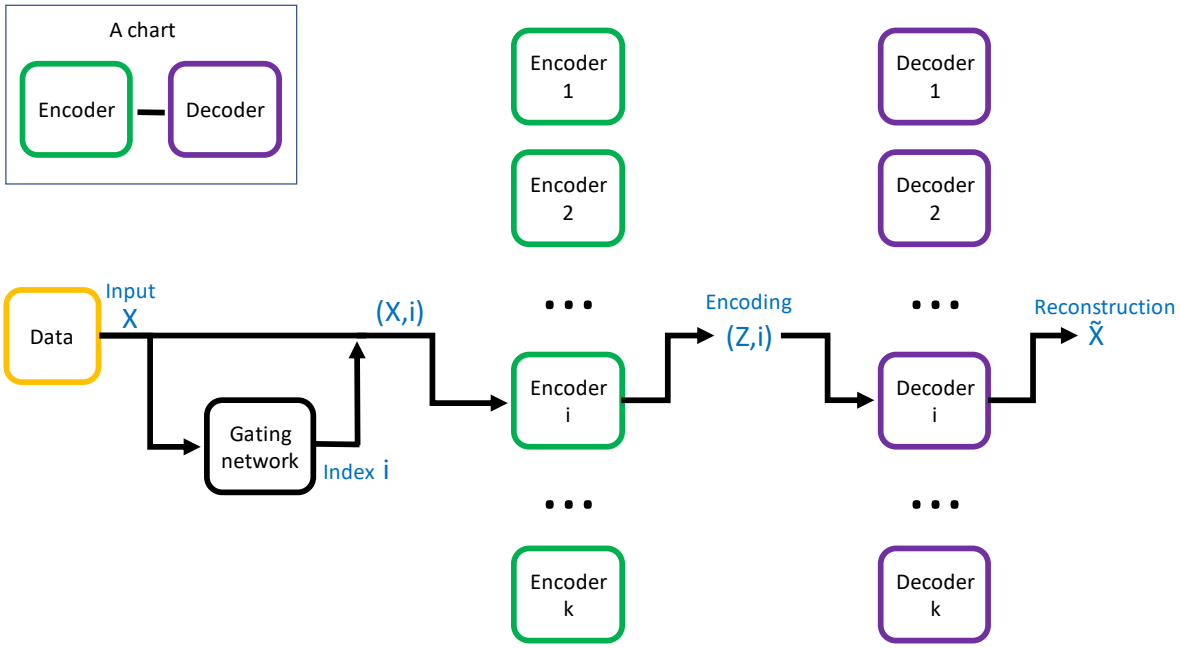


Figure 5.2 – Diagram of the proposed system.

At inference, the gating network takes the input X and outputs the weights $G(X)$. They are used to select an index $j := \text{argmax}\{G(X)_j | i = 1, \dots, k\}$. The encoding of X is then defined as the pair (Z, j), where $Z := E_j(X) \in \mathbb{R}^n$, and its reconstruction as $\tilde{X} := D_j(Z) = D_j(E_j(X)) \in \mathbb{R}^m$. This is illustrated in Figure 5.2.

5.3 Training

There are several difficulties to setting up and training a deep atlas, among which:

- Train an intrinsically discontinuous system using continuous methods.
- Help charts specialize in different regions of the input space.
- Balance the load between charts, so that no chart ends up useless.
- Choose an appropriate number of charts, and an appropriate level of complexity for the encoders and decoders.

We address the first three points using appropriate loss functions in Subsection 5.3.1, and the last one with a chart splitting mechanism in Subsection 5.3.2. All of our code is available on `https://github.com/CharlesArnal/Deep_Atlas`.

5.3.1 Losses

We define the total loss function L to minimize as a weighted sum of a collection of specialized loss functions:

- The main reconstruction loss function, which forces the reconstructed output of the encoder to be close to its input:

$$L_R(\mathcal{D}) := \frac{1}{|\mathcal{D}|} \sum_{X \in \mathcal{D}} \sum_{i=1}^{k} \alpha_i ||D_i(E_i(X)) - X||_2^2, \tag{5.3.1}$$

  where $(\alpha_1, \ldots, \alpha_k) = \text{softmax}(G(X)_1, \ldots, G(X)_k)$ and $\mathcal{D}$ is the training set. The softmax allows the loss function to be di□erentiable, while still being a reasonably good approximation of

$$||D_j(E_j(X)) - X||_2^2 \qquad j := \text{argmax}\,\{G(X)_i | i = 1, \ldots, k\}\,.$$

  Hopefully L_R should encourage both the gating network to maximise the weight $G(X)_l$ corresponding to the chart C_l best suited for X (i.e. that for which $||D_l(E_l(X)) - X||_2^2$ is minimal), as well as help C_l improve its reconstruction of X.
  We consider three possible variations to L_R: the first one would be to replace it by

$$L_R^{inside}(\mathcal{D}) := \frac{1}{|\mathcal{D}|} \sum_{X \in \mathcal{D}} \left|\left| \sum_{i=1}^{k} \alpha_i D_i(E_i(X)) - X \right|\right|_2^2,$$

  where α_i is as before. We expect L_R^{inside} to yield poorer results, as it encourages a linear combination of the reconstructions to be good, instead of the best among them.
  The second variation, which was introduced in [Jacobs et al., 1991b] for mixtures of experts in general, would be to replace L_R by

$$L_R^{exp}(\mathcal{D}) := \frac{-1}{|\mathcal{D}|} \sum_{X \in \mathcal{D}} \log \left(\sum_{i=1}^{k} \alpha_i \exp\left[-\frac{1}{2} ||D_i(E_i(X)) - X||^2 \right] \right).$$

  We would like each chart to specialize and become particularly adept at reconstructing the points that are assigned to it. However, as chart C_i gets better at reconstructing a given point X, the gradient

$$\frac{\partial L_R(X)}{\partial W_i} := 2 \sum_{i=1}^{k} \alpha_i (D_i(E_i(X)) - X)$$

  of L_R with respect to the weights W_i of C_i becomes smaller in norm, while it is not the case for the gradient with respect to the charts that poorly reconstruct X -

it means that X might paradoxically have a larger e□ect on charts that are poorly suited to encoding it (though this e□ect is partially compensated by the fact that the coe□cients α_i should be large for charts that reconstruct X well).
This risk is averted with L_R^{exp}, as we see that

$$\frac{\partial L_R^{exp}(X)}{\partial W_i} := \frac{\alpha_i \exp\left[-\frac{1}{2}||D_i(E_i(X)) - X||^2\right]}{\sum_{i=1}^k \alpha_i \exp\left[-\frac{1}{2}||D_i(E_i(X)) - X||^2\right]}(D_i(E_i(X)) - X),$$

hence $D_i(E_i(X)) - X$ appears in the gradient of L_R^{exp} with respect to W_i multiplied by a renormalizing factor that is larger when $D_i(E_i(X)) - X$ is small in norm, i.e. when the chart correctly reconstructs X.
The third variation (which can be combined with either L^R, L_R^{inside} or L_R^{exp}) would be to compute the coe□cients α_i in Formula (5.3.1) with a modified, more “intense" softmax:

$$\alpha_i := \frac{\exp(\lambda G(X)_l)}{\sum_l \exp(\lambda G(X)_l)}$$

for some $\lambda > 1$. This would have the e□ect of giving even more weight to the dominant chart $j := \text{argmax}\{G(X)_i | i = 1, \ldots, k\}$, and thus make L_R a better approximation of $||D_j(E_j(X)) - X||_2^2$.

- A load loss, intended to help distribute somewhat evenly the data points among the various charts. Similar losses are often used in Mixture of Experts systems to keep issues such as the “zero-coe□cient problem" - where the gating network never attributes any data point to some of the experts (here some of the charts), see e.g. [Hansen, 1999] - from arising.
We would like to define a load vector $Load(\mathcal{D}) \in \mathbb{R}^k$ as

$$Load_i(\mathcal{D}) := \sum_{X \in \mathcal{D}} 1\{i = \text{argmax}\{G(X)_i | i = 1, \ldots, k\}\},$$

i.e. the i-th coordinate is the number of data points $X \in \mathcal{D}$ attributed to the i-th chart by the gating network. As this would not be continuous, we replace it with

$$Load_i(\mathcal{D}) := \sum_{X \in \mathcal{D}} F\left(\alpha_i(X), \{\alpha_l(X)\}_{l \neq i}\right), \tag{5.3.2}$$

where $F(x, \{y_1, \ldots, y_{n-1}\})$ is a smooth approximation of $1\{x > \max(y_1, \ldots, y_{n-1})\}$. In our implementation, we test two choices for F. The first is

$$Load_i(\mathcal{D}) := \sum_{X \in \mathcal{D}} \Phi\left((\alpha_i(X) - \max\{\alpha_l(X)\}_l) \cdot Ck\right), \tag{5.3.3}$$

where Φ is the cumulative function of the normal distribution[2], $C > 0$ is a constant and k is the number of charts. The term Ck serves as a normalizing factor, as the differences between the coefficients α_i tend to get smaller when k gets larger. Though Function (5.3.3) is continuous but not smooth, it is smooth almost everywhere - we have not observed any problem during training. The other choice of F we have considered is

$$Load_i(\mathcal{D}) := \sum_{X \in \mathcal{D}} \frac{\alpha_i(X)^\mu}{\sum_l \alpha_l(X)^\mu}, \tag{5.3.4}$$

for some constant $\mu > 1$.

We then define the load loss function on the training set $\mathcal{D}$ as

$$L_{load}(\mathcal{D}) := CV(Load(\mathcal{D}))^2,$$

where CV is the coefficient of variation of $Load(\mathcal{D})$, i.e. its standard deviation divided by its mean. When all charts receive an equal number of points, $L_{load}(\mathcal{D})$ will be 0, while it will be large when the load is unevenly distributed. Our load loss was inspired by the one introduced in [Shazeer et al., 2017], though we discarded the probabilistic formulation used by the authors and generalized by focusing on what we identified as the underlying principle at work (which is summarised in Formula (5.3.2)).

- The importance loss is also intended to balance out the number of samples attributed to each chart. We define the importance vector $Importance(\mathcal{D}) \in \mathbb{R}^k$ as

$$Importance_i(\mathcal{D}) := \sum_{X \in \mathcal{D}} \alpha_i(X),$$

where the $\alpha_i(X)$ are as before the coefficients obtained by applying a softmax to $G(X)$. The importance loss function is

$$L_{importance}(\mathcal{D}) := CV(Importance(\mathcal{D}))^2,$$

where CV being as before the coefficient of variation.

- The inverse reconstruction loss

$$L_I(\mathcal{D}) := \sum_{X \in \mathcal{D}} \left|\left|E_{i(X)}(D_{i(X)}(Z(X))) - Z(X)\right|\right|_2^2,$$

where $(Z(X), i(X))$ is the encoding of X, is meant to put additional constraint on the encoders and decoders to be the inverse of each other (when restricted to the training data). When computing the gradient of the loss (see below), the encodings

[2] Other (possibly computationally cheaper) smooth approximations of $1_{\{x>0\}}$ could also work.

$(Z(X), i(X))$ are treated as constants.

- The classification loss

$$L_C(\mathcal{D}) := \sum_{X \in \mathcal{D}} -\log\left(\alpha_{i(X)}(D_{i(X)}(Z))\right),$$

where as above $(Z(X), i(X))$ is the encoding of X, encourages the decoder and the gating network to place the reconstruction $\tilde{X} = D_{i(X)}(Z)$ in a region of the input space associated to the $i(X)$-th chart by the gating network (as X comes from such a region).

We define the total loss as

$$L(\mathcal{D}) := L_R(\mathcal{D}) + \gamma_{load} L_{load}(\mathcal{D}) + \gamma_{importance} L_{importance}(\mathcal{D}) + \gamma_I L_I(\mathcal{D}) + \gamma_C L_C(\mathcal{D}),$$

for weights $\gamma_{load}, \gamma_{importance}, \gamma_I, \gamma_C > 0$. The system is then trained using standard gradient descent methods. Note that for L_{load} and $L_{importance}$ to be accurately estimated during training, the proportion of points attributed to each chart in each batch must be close to the average proportion over the entire training set. Consequently, the dataset must have been randomly shu□ed, and the size of each batch divided by the number of charts cannot be too small.

We do not necessarily expect all sub-losses to be useful. In particular, the importance loss seems like a simpler, less relevant[3] version of L_{load}; though we want to test its e□ects, we expect it to be superfluous. The inverse reconstruction loss and the classification loss were meant to allow us to exploit the training data more thoroughly, but they might also turn out to be redundant when paired with the main reconstruction loss. In Chapter 7, we test various choices of weights to compare the impacts of the sub-losses.

5.3.2 Chart splitting algorithm

Having to set a priori the “good" number of charts (see Section 5.4 for more on that) would be an inconvenient limitation. As an answer to that, we have devised a chart-splitting mechanism that automatically creates new charts during training to adapt to the data. The core idea is for the system to identify every T epochs of gradient descent training the worst-performing chart and to split it into two almost identical copies, that are then free to specialize as training resumes. The rationale is that if a chart significantly underperforms compared to the others, it must be because it is trying to encode a region of space that is too complicated to be e□ciently modeled by a single chart - the burden must be shared. In more details:

[3] As what matters is the number of points attributed to each chart; if the coe□cient α_i associated to chart C_i is the second largest for each data point, the importance of C_i might be large even though C_i is never selected by the gating network.

- The performance of a chart C_i over the training set is measured using the average square reconstruction error $ER(C_i)$ over the data points attributed to it.

- Every T epochs (for a fixed $T \in \mathbb{N}$), the worst performing chart C_j is selected. If $ER(C_j)$ is larger than either a predefined threshold or than

$$\text{Mean}(ER(C_1), \dots, ER(C_k)) + 1.5 \cdot IQR(ER(C_1), \dots, ER(C_k)),$$

  i.e. the average performance of all charts plus 1.5 times the interquartile range of the performances[4], then C_j undergoes the splitting procedure.

- We start by creating a copy $\tilde{C_j}$ of the chart C_j, i.e. a new Encoder-Decoder couple with the exact same architecture and weights as C_j. This is illustrated in the upper left part of Figure 5.3 (where the chart being split is the last chart C_k).

- We then need to create a new output unit associated to $\tilde{C_j}$ in the gating network. When splitting the chart C_j, we want the set S of points of the training set attributed to it by the gating network in inference mode to be distributed among C_j and $\tilde{C_j}$ in a geometrically sensible way. For example, should the chart being split be tasked with encoding two disconnected regions of the data manifold, we wish for each of these regions to be inherited by a different copy so as to make the region encoded by each chart as topologically simple as possible.
  To ensure that, we apply a self-supervised clustering algorithm to either:

  - The set S itself in the input space. It can be problematic if the input space is very high-dimensional.
  - The image of the set S by the encoder E_j.
  - The image of the set S by the gating network minus the last linear layer.

  The clustering algorithm must be set so that it identifies two clusters. In our implementation, we use either k-Means or Agglomerative clustering (from the scikit-learn library, see [Pedregosa et al., 2011]) for the clustering algorithm. This yields a partition of S into two sets, S_1 and S_2. We then use a Support Vector Machine (again with scikit-learn, see also [Bishop, 2006, Chapter 7]) to separate the images of these two sets by the first part of the gating network, i.e. all of it minus the last (linear) output layer. The SVM gives us a separating hyperplane $\{V \cdot X + b = 0\}$. This is illustrated in the upper right part of Figure 5.3.

- We create a copy of the output unit U_j associated to the chart C_j in the last linear layer of the gating network. It has the same weights and bias W_j, b_j. We then add

[4] This is a somewhat arbitrary criterion to determine what an "outstandingly" bad performance is - it was inspired by a similar criterion commonly used to define outliers.

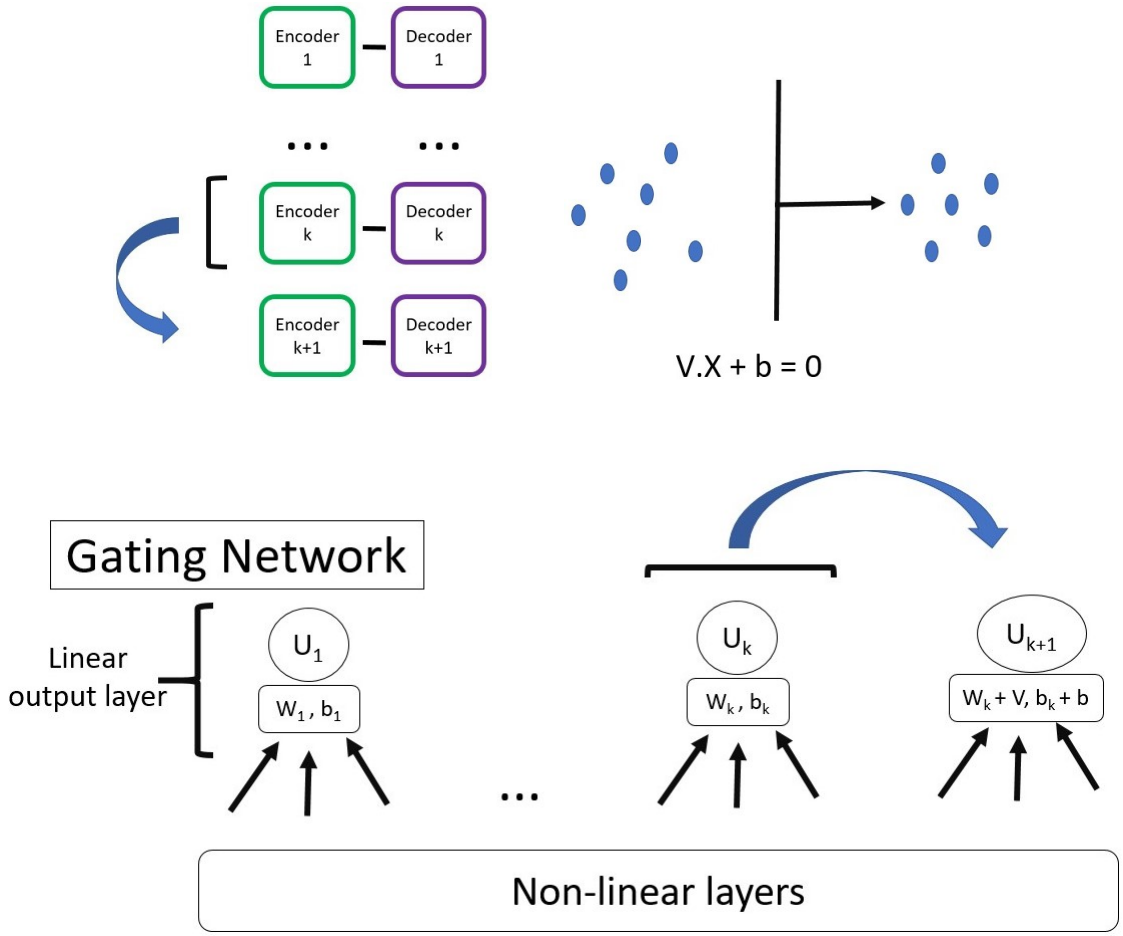


Figure 5.3 – Splitting a chart. The chart itself is duplicated (upper left), the points attributed to it are clustered and separated (upper right), and the associated output unit is copied and modified using the separating hyperplane (bottom).

to them the vector V and intercept b obtained from the SVM. We make sure to first renormalize V and b so that they are very small compared to the rest of the weights of the output layer. This new unit is associated to the new chart $\tilde{C}_j$ being constructed. This has the effect that almost all points are attributed to the same charts as before, except the points S previously attributed to C_j: the points that belong to S_1 are still attributed to C_j, while those that belong to S_2 are now attributed to $\tilde{C}_j$. This is illustrated at the bottom of Figure 5.3.

5.4 On the minimal number of charts required

Instead of adaptively adding new charts, one might want to estimate a priori the number of charts required to encode the variety. Many methods have been developed to estimate the dimension of a manifold from data points that live on it (see for example

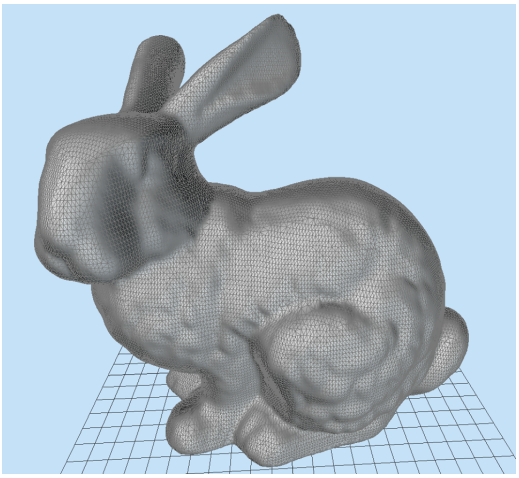


Figure 5.4 – The Stanford Bunny.

[Camastra and Staiano, 2016]). Once the dimension, or at least a good estimate of it, is known, there are some theoretical guarantees: given an n dimensional connected compact manifold M, let $b(M)$ be the minimum number of (topological) balls needed to cover M. Then $b(M) \leqslant n + 1$ if M has no boundary, and $b(M) \leqslant n$ if M has a non-empty boundary (see [Cavicchioli and Grasselli, 1985]). The second bound becomes tight if we ask that the balls' intersections be particularly nice, but not necessarily otherwise (e.g., the n dimensional sphere S^n can always be covered by two balls).

However, this approach is not as relevant as it might seem, though it can give us some useful early estimate. Indeed, the number of charts required to faithfully encode the manifold in practice depends on the power of the charts used, i.e. the depth and width of their neural networks, and might be much larger than the theoretical minimum: as the approximation power of each of our chart is limited by the power of its encoder and decoder, they cannot model correctly something that is topologically a ball but geometrically too "wiggly". As an example, consider in Figure 5.4 the surface of the well-known Stanford Bunny from [Turk and Levoy, 1994]: it is topologically a sphere, and can thus be covered by two disks, but a small autoencoder would not be able to encode all the fine details of half of its surface. Conversely, a powerful enough autoencoder can potentially faithfully encode more than a single subset of the manifold homeomorphic to a ball; hence, any theoretical lower bound is not to be trusted either.

Chapter 6

Comparison to other algorithms

In this section, we compare Deep Atlas to some existing systems with which it shares commonships.

6.1 Autoencoders

Deep Atlas is a refinement of the original autoencoder idea (see [Kramer, 1991]). The key difference is the introduction of the gating network, which induces an element of discontinuity (whose importance was emphasized in Chapter 2) and allows subsystems (the charts) to specialize in different regions of the input space.

Note that the charts which we consider in this thesis are themselves very basic autoencoders, as our focus was on the gating network and the chart splitting mechanism. Of course, some of the various, more sophisticated types of autoencoders that have been developed - sparse autoencoder, denoising autoencoder, variational autoencoder, etc. (see [Bank et al., 2020] for a survey) - could be integrated to our architecture to improve the performance of each individual chart.

6.2 Mixture of experts

At a first glance, Deep Atlas could be seen as a special case of the classical Mixture of Experts idea (introduced in [Jacobs et al., 1991b]). In their survey [Masoudnia and Ebrahimpour, 2014], S. Masoudnia and R. Ebrahimpour distinguish between Mixtures of Implicitly Localised Experts (MILE), i.e. Mixtures of Experts in which the attribution of the data points to different experts is trained conjointly with the experts, and Mixtures of Explicitly Localised Experts (MELE), where the dataset is partitioned first into subsets (using some clustering algorithm), then a different expert is trained on each of these subsets. Following that taxinomy, Deep Atlas as we have implemented it would be a MILE. It could be turned into a MELE by applying any kind of clustering algorithm to

the training dataset and training each chart on a specific subset, as well as replacing the gating network by some classification algorithm that would learn that initial partition, but we expect it not to be very e□cient. Indeed, when performing manifold learning, obtaining a reasonable partition of the data is in a sense the core of the problem, as opposed to some secondary pre-processing task - it involves capturing part of the topology of the manifold, and tends to be very hard (many clustering algorithms tend to struggle when applied to manifolds). This is less the case for other tasks, such as classification, for which MELE techniques have been successfully applied (e.g. in [Tang et al., 2002]).

There are however important di□erences between Deep Atlas and most existing Mixture of experts-like algorithms (beyond the fact that many do not use deep learning to its full potential, due to having been developed before it became fashionable).

The first would be that the use of several experts is usually motivated by increased robustness, as their errors are hoped to be partially uncorrelated, which would allow them to compensate for each other's failures, as well as by increased performance due to specialization (see [Nowlan and Hinton, 1991]). Though Deep Atlas also benefits from those e□ects, our core reason to advocate for the use of several charts is much more data-driven: we have argued in Chapters 2 and 5 that using several charts coupled with a discontinuous gating mechanism is not only better suited to the task of manifold learning, but a theoretical necessity in many cases.

A second di□erence is that most Mixture of Experts systems output some linear combination of the predictions of their experts. This often encourages several experts to produce a somewhat good prediction for a given data point, though some might dominate. Conversely, a single chart is selected for each point by Deep Atlas; the predictions of the others can be absolutely aberrant, and are in fact expected NOT to be relevant for most points that are not attributed to them (this is illustrated in Section 7.2).
This sparsity and discontinuity of the output makes Deep Atlas somewhat comparable to [Shazeer et al., 2017], where the predictions of only the top k-experts are combined. However, the fact that $k = 1$ in our case creates important di□erences. In particular, the training loss must be adapted and a distinction training/inference must be made, as computing the loss using predictions made in inference mode (as is done in [Shazeer et al., 2017]), i.e. with a single chart per data point, would result in the gradient being 0 for the weights of all charts except one (for a given data point), as well as for the weights of the gating network. The system would get stuck in very suboptimal local optima. Another distinction in spirit is that as with other Mixture of Experts, the experts of [Shazeer et al., 2017] are meant to compute the *same thing*, though they are specialized in di□erent subsets of the data, which is not the case for Deep Atlas: each chart represents a di□erent encoding, and the output of the encoder of a chart cannot be meaningfully compared to that of another encoder.

The last core di□erence to classical Mixtures of Encoders is the adaptive creation of new experts, in the form of the chart-splitting mechanism. There have been similarly minded

e□orts to adapt the system as a function of the results, but only in the form of changes to the gating network (as far as the author is aware), as in [Jacobs et al., 1991a].
More generally, adaptive neural networks have been designed, but the changes possible seem to usually be limited to adding new fully or partially connected layers, as in [Cortes et al., 2017], as opposed to an entire sub-neural network coupled with a non-continuous gating network, as is the case with Deep Atlas. The specific mechanism by which we initialize the weights of the new expert and create an associated output unit for the gating network seems to be original as well.

Splitting the charts also allows us to partially alleviate a problem that often plagues MILEs (as was observed in [Tang et al., 2002]), which is for the learnt partitions of the dataset to be nonsensical, e.g. with unrelated and disconnected regions of space attributed to the same expert. Should the chart being split cover several disconnected regions of the manifold, the technique used (see details in Section 5.3) aims to distribute these among the old chart and the newly created one, reducing the number of disconnected regions per chart.

6.3 Other piecewise continuous manifold learning techniques

The idea of learning a manifold as an atlas was first discussed in [Pitelis et al., 2013] - their geometric intuition was similar to ours, though they do not discuss the theoretical limitations of continuous encoding algorithms in as much detail. However, their implementation of that initial intuition is very di□erent. Firstly, the authors limit themselves to linear charts, rather than using deep autoencoders as we do. Though manifolds can be well-modelled by linear charts (as long as there are enough of them), one can hope for deeper models to achieve better performances. Another di□erence is that they do not use a gating network to attribute points to charts - instead, the points are hard-assigned to charts under some constraints regarding the assignments of their k-nearest neighbours. As a result, the training procedure (which is based on an Expectation-Maximisation process) is entirely di□erent from ours, and their system does not explicitly learn a mapping from the input space to the latent space and vice-versa (though some nearest neighbours-based trick could be used to encode out-of-sample points). Finally, there is no analog to our chart-splitting mechanism that would allow the number of charts to increase adaptively.

Chapter 7

Experiments

7.1 Introduction

In this chapter, we test our implementation of Deep Atlas on a variety of tasks and data sets. More specifically, we present the data sets used, we run simple experiments to illustrate the main mechanisms of Deep Atlas in Section 7.2, we test various hyperparameters combinations in Section 7.3 to measure their effect on performance, and we compare Deep Atlas (with the best hyperparameters combination found) to a classical autoencoder on classification and unsupervised anomaly detection tasks in Sections 7.4 and 7.5.
Our main goal here is to study the core principles underlying Deep Atlas, and to compare it to classical autoencoders; consequently, we choose for each data set a reasonable configuration but do not spend significant time trying to optimize it to maximise performance. In particular, we do not use convolutional layers for image data sets.

We use four toy data sets where the points lie on nice, smooth manifolds: a sphere in $\mathbb{R}^3$, the union of a sphere and a torus in $\mathbb{R}^3$, the famous Swiss roll, and a 40-dimensional sphere embedded in $\mathbb{R}^{80}$. We split those data sets into classes (5, 4 and 3 classes respectively) in such a way that the classes roughly correspond to distinct regions of space. The dimension

Figure 7.1 – The union of the sphere and the torus, a digit from the MNIST data set and the Swiss roll.

of the latent space was 2 for the 2-sphere, union of sphere and torus and Swiss roll and 40 for the 40-dimensional sphere. We also add 5% of noisy anomalies to the data sets when performing unsupervised anomaly detection. Additional details on the generation of those data sets can be found in Appendix A.

We also use a classical benchmark data, the MNIST collection of 28×28 pictures of handwritten digits ($60'000$ in the training set, $10'000$ in the test set), which we flatten into vectors of $\mathbb{R}^{784}$ (see [LeCun et al., 2010]). When performing classification, we use as classes the numbers represented. The latent space dimension chosen was 50 (similar to what is done in [Burda et al., 2016]). The union of the sphere and the torus, a digit from the MNIST data set and the Swiss roll are represented in Figure 7.1.

All data sets are renormalised to be of variance 1 before running experiments so as to make reconstruction losses comparable.

The artificial data sets were chosen to help us capture in a simple, easy to analyze context the essence of the properties in which we are interested, i.e. manifold-like structure (and more specifically closed manifold-like structure in the case of the spheres and the torus). The 40-dimensional sphere is meant to help us study Deep Atlas's behavior in higher dimensions, while remaining mathematically simple. Conversely, the MNIST data set should help us test the limits of Deep Atlas, and see whether it can be of use in cases where no manifold-like structure is expected (unlike what might be the case with data from physics or computer vision).

7.2 Illustration of the main mechanisms

In this section, we illustrate with simple qualitative experiments the working of Deep Atlas. In Figure 7.2, a Deep Atlas with 4 charts is trained to encode in 2 dimensions and reconstruct points from the 2-dimensional sphere in $\mathbb{R}^3$. The original data set is represented at the top of the left part of the figure; the color of the points indicate which charts they have been assigned to at inference time. Each chart encodes the points associated to it in a di□erent copy of $\mathbb{R}^2$; those encodings are represented conjointly below the sphere (at di□erent vertical coordinates) for ease of visualisation. We see that the gating network has successfully learnt to separate the sphere into subsets that are easy to encode. On the right part of the figure, the points are shown in black, their reconstruction in red, and a blue segment connects each point to its reconstruction. We see that no point is dramatically poorly reconstructed, unlike what was observed when using a single continuous autoencoder in Section 2.2.

In Figure 7.3, we see that Deep Atlas learns relevant decompositions into charts: each branch of the simple V-shaped data set is attributed to a di□erent chart (as indicated by the color scheme), which allows them to be correctly encoded and decoded by simple linear maps.

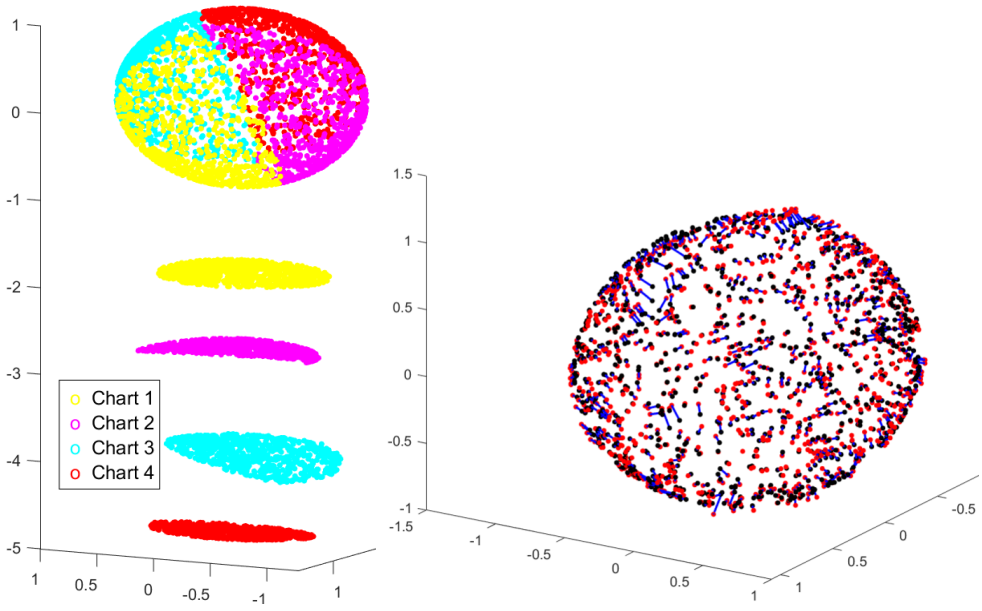


Figure 7.2 – On the left, Deep Atlas separates the 2-dimensional sphere into 4 distinct charts; the encodings are represented below the sphere. On the right, the data points (in black) are represented jointly with their reconstruction (in red).

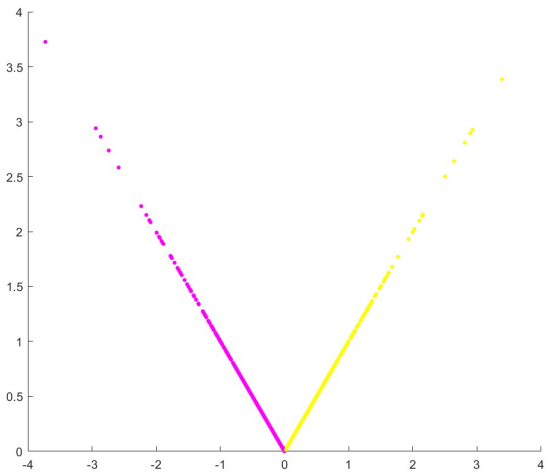


Figure 7.3 – Deep Atlas learns to split the V-shaped data set into charts in a relevant fashion.

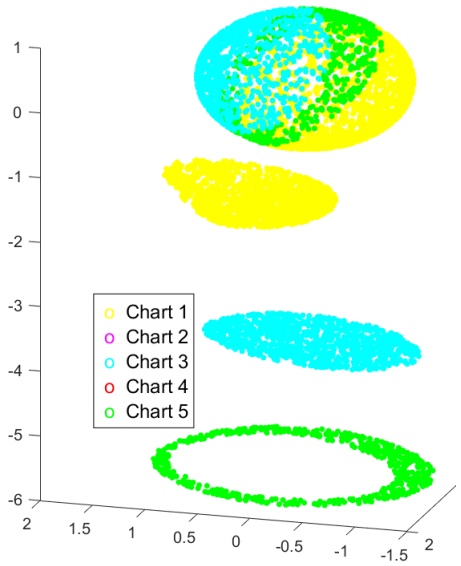


Figure 7.4 – The charts are organized in a non-trivial way: no point is attributed to two of them, and the points associated to charts number 5 form a topologically non-trivial submanifold of the sphere.

In Figure 7.4, another instance of training a Deep Atlas on the 2-sphere. We see that the repartition of the points between the di□erent charts can sometimes be more complicated than in Figure 7.2: after training, the gating network does not attribute any point to charts number 2 and 4, and the points attributed to chart number 5 form a topologically non-trivial subset of the sphere (though it can be faithfully embedded in $\mathbb{R}^2$).

Charts are only meant to be locally relevant, in the sense that they should faithfully encode and decode points in the region of the data set that is attributed to them by the gating network. This is illustrated in Figure 7.5, where we depict how the charts of a Deep Atlas trained to encode the 2-sphere map the encoding space $\mathbb{R}^2$ to $\mathbb{R}^3$; in other words, instead of only depicting the images by the decoder D_i of the encodings by E_i of the points attributed to chart C_i by the gating network, we show the extended graph of the decoder $D_i : \mathbb{R}^2 \to \mathbb{R}^3$.

We see that as expected, the graph of the decoders are tightly wrapped around the area of the sphere attributed to the relevant chart, but diverge from the surface of the sphere and become “nonsensical" away from those areas - the charts are not meant to faithfully encode and decode the entire manifold, only regions of it.

The complexity of the mappings learnt by the charts depends on the architecture of their encoder and decoder. Figure 7.6 represents the graphs of decoders of a Deep Atlas that were limited to a single linear layer. As a result, they learnt very crude local linear approximations of the surface of the Swiss roll that they are trying to encode.

In Figure 7.7, we illustrate the chart splitting mechanism. On the left, we see that the

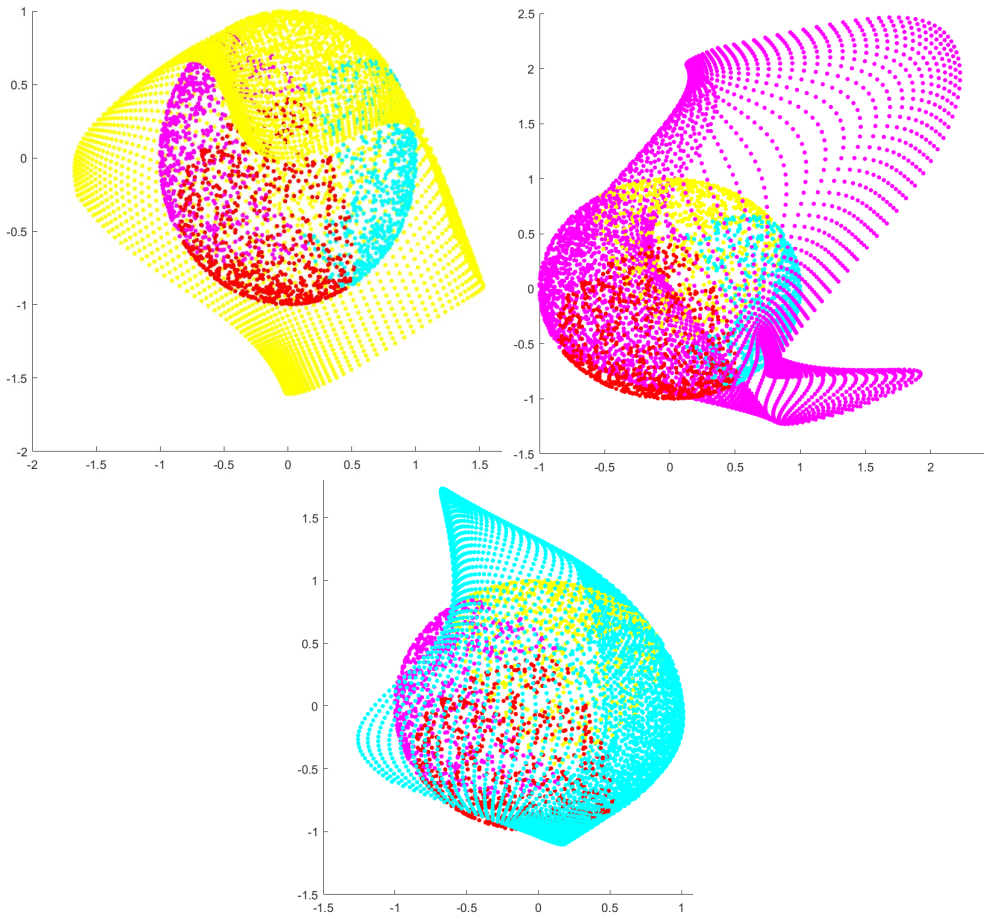


Figure 7.5 – The graphs of the decoder of various charts of a Deep Atlas trained on the 2-sphere.

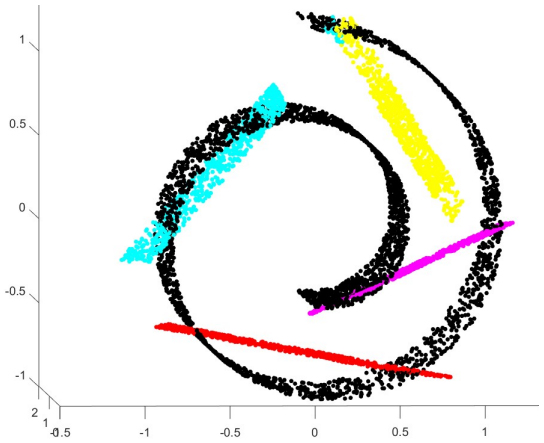


Figure 7.6 – The graphs of the decoders of the charts of a Deep Atlas; those decoders were constrained to be linear.

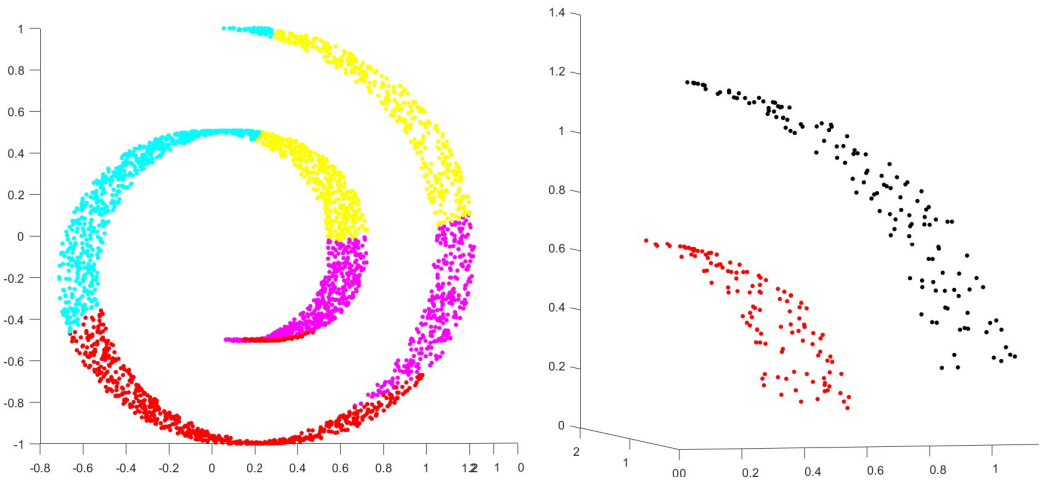


Figure 7.7 – On the left, the regions attributed to various charts of a Deep Atlas - we see that some of these regions are not connected. On the left, the region associated to the worst performing chart is split in two by a clustering algorithm.

points attributed to some of the charts form disconnected subsets of the Swiss roll. As we would like each chart to encode a region of the data set as simple as possible, we wish to avoid such situations.

When splitting charts, the worst performing chart is selected - in this case, the chart represented in yellow. A clustering algorithm is applied either to the points associated to the chart, or to some encoding of them (see Subsection 5.3.2 for details). On the right, we see that it successfully separates the two connected component of the region attributed to the chart. Two charts are then created to replace the yellow chart, each inheriting one of the two connected components.

7.3 Hyperparameter testing

In Chapter 5, our architecture was introduced with many possible variations and hyperparameters to set. In this Section, we explore the effects of those hyperparameters.

In the experiments below, we let one parameter vary while the rest are set according to our baseline configuration, which is as follows[1]:

- The main reconstruction loss is L_R^{exp}, with a softmax coefficient λ of 1.
- The weights $\gamma_{load}, \gamma_{importance}, \gamma_I$ and γ_C associated to the load loss, importance loss, inverse reconstruction loss and classification loss are set to 0.
- The number of charts is fixed and set to 6 for the union of the torus and the sphere, 2 for the 40-dimensional sphere and 10 for the MNIST data set.

[1] See Section 5.3 for the definition of the various terms used.

For each experiment, we then let one parameter vary. The exact architectures and training procedures used are described in details in Appendix A. Performance is mainly measured using the average reconstruction error

$$L_E(\mathcal{D}) := \frac{1}{|\mathcal{D}|} \sum_{X \in \mathcal{D}} ||D_{i(X)}(E_{i(X)}(X)) - X||_2^2$$

computed in inference mode on the test set (where $i(X) = \text{argmax}\{G(X)_j | j = 1, \ldots, k\}$).

In Table 7.8, we test the three main reconstruction losses L_R, L_R^{inside} and L_R^{exp}. As L_R^{exp} seems to perform slightly better (most likely for reasons discussed in 5.3, and in line with what was observed in [Jacobs et al., 1991b]).

Various reconstruction losses			
	Reconstruction loss		
Data set	L_R	L_R^{inside}	L_R^{exp}
Torus union sphere	0.0019	1.750	0.0016
40-dim sphere	0.173	2.702	0.170
MNIST	0.091	14.578	0.092

Figure 7.8 – Average reconstruction error using various main reconstruction losses during training.

In Table 7.9, we test 4 different values for the softmax exponent λ used in the computation of the charts weights α_i. It does not seem to have much of an impact, most likely because the gating network naturally learns to give almost all the weight to the best chart for a given point.

Choice of softmax exponent				
	Softmax exponent λ			
Data set	1	2	4	8
Torus union sphere	0.0015	0.0015	0.0015	0.0075
40-dim sphere	0.173	0.181	0.174	0.173
MNIST	0.102	0.110	0.104	0.104

Figure 7.9 – Average reconstruction error using various softmax exponents λ in the computation of the weights α_i during training.

In Table 7.10, we test various combinations of weights γ_{load} and $\gamma_{importance}$ for the load loss L_{load} and the importance loss $L_{importance}$ and measure the average reconstruction error and load loss (the load loss is computed using Formula (5.3.3)). We see that while they do impact the load loss, a decrease in load loss does not translate to a significant improvement in performance, unlike what was observed in [Shazeer et al., 2017]. In the case of the union

of the sphere and the torus, this is because the intrinsic geometry of the problems makes it natural for some charts to have more points attributed to them than others: if the sphere is split into 2 charts and the torus into 4, they will naturally have di□erent number of points, and forcing them to inherit points from the other connected component will hurt performance. Moreover, denser regions do not necessarily need more charts to be properly encoded, as long as their geometry is simple enough, leading the associated charts to naturally receive more points than charts associated to sparser regions.

We also see that without any load or importance loss, the load loss on the MNIST tends to be close to 9, which indicates that a single chart is tasked with encoding all the points; the charts failed to specialize in di□erent regions of the data set. However, performance does not increase when the load loss forces the load to be more evenly distributed; we discuss the reasons for this further below.

Impact of the load-balancing losses						
	$(\gamma_{load}, \gamma_{importance})$					
Data set	$(0,0)$		$(0.01,0)$		$(0.1,0)$	
	L_E	L_{load}	L_E	L_{load}	L_E	L_{load}
Torus union sphere	0.0014	0.429	0.0025	0.002	0.0040	0.002
40-dim sphere	0.173	0.078	0.172	0.001	0.173	0.003
MNIST	0.104	9.00	0.090	0.673	0.095	0.005
	$(0,0.01)$		$(0.01,0.01)$		$(0.1,0.01)$	
	L_E	L_{load}	L_E	L_{load}	L_E	L_{load}
Torus union sphere	0.0010	0.003	0.0020	0.001	0.0020	0.002
40-dim sphere	0.171	0.0002	0.172	0.0002	0.171	0.00008
MNIST	0.104	9.00	0.090	1.008	0.095	0.002
	$(0,0.1)$		$(0.01,0.1)$		$(0.1,0.1)$	
	L_E	L_{load}	L_E	L_{load}	L_E	L_{load}
Torus union sphere	0.0016	0.002	0.0023	0.004	0.0027	0.001
40-dim sphere	0.173	0.0002	0.172	0.004	0.171	0.00001
MNIST	0.095	0.004	0.096	0.005	0.095	0.004

Figure 7.10 – Average reconstruction error L_E and load loss L_{load} using various combinations of weights γ_{load} and $\gamma_{importance}$ for the load loss L_{load} and the importance loss $L_{importance}$.

In Table 7.11, we test the alternate load loss defined with a softmax-like formula (see Equation (5.3.4)) with various coe□cients μ and a set value $\lambda_{load} = 0.01$. A small improvement in performance seems to be observed for $\mu = 4$ on the union of the torus and the sphere, but further testing on a wider variety of data sets would be necessary to reach definitive conclusions.

Alternative load loss L_{load} from Equation (5.3.4)								
	Softmax exponent for the load loss							
Data set	1		2		4		8	
Torus union sphere	0.0015	0.0004	0.0013	0.200	0.0011	0.002	0.0020	0.009
40-dim sphere	0.172	0.00006	0.171	0.006	0.171	0.0006	0.172	0.0001
MNIST	0.084	4.000	0.086	4.000	0.084	2.338	0.085	2.350

Figure 7.11 – Average reconstruction error L_E and load loss L_{load} for various softmax exponents μ in the alternative load loss formula from Equation (5.3.4), with a load loss weight $\gamma_{load} = 0.01$.

In Table 7.12, we test various weights for the inverse reconstruction loss L_I and the classification loss L_C. No clear effect can be observed, most likely because they are made redundant by the main reconstruction loss which encourages the same behaviour from the system (good reconstruction and appropriate attribution of charts).

Inverse reconstruction loss and classification loss				
	γ_C			
Data set	0	0.01	0.1	0.25
Torus union sphere	0.0011	0.0019	0.0033	0.0025
40-sphere	0.182	0.174	0.174	0.174
MNIST	0.108	0.105	0.103	0.104
	γ_I			
Data set	0	0.01	0.1	0.25
Torus union sphere	0.0014	0.0020	0.0021	0.0013
40-sphere	0.172	0.173	0.179	0.180
MNIST	0.102	0.104	0.104	0.111

Figure 7.12 – Average reconstruction error L_E various inverse reconstruction loss weight γ_I and classification loss weight γ_C.

In Table 7.13, we compare the reconstruction error obtained with various (fixed) numbers of charts. While it seems to have an important impact on performance for the union of the torus and the sphere, with less than 6 charts leading to a lack in flexibility and 9 charts to some overfitting, it has very little effect for the other two data sets.

Various fixed number of charts					
Torus union sphere	Num charts	1	3	6	12
	R. E.	0.0331	0.0029	0.0012	0.0019
40-dim sphere	Num charts	1	2	4	8
	R. E.	0.173	0.181	0.198	0.220
MNIST	Num charts	1	5	10	20
	R. E.	0.105	0.103	0.102	0.102

Figure 7.13 – Average reconstruction error L_E for various softmax exponents μ in the alternative load loss formula from Equation (5.3.4), with a load loss weight $\gamma_{load} = 0.01$.

In Table 7.14, we test the chart splitting mechanism by starting with a single chart, then repeatedly splitting it (after a round of training) until reaching 6 charts for the union of the torus and the sphere, 2 charts for the 40-dimensional sphere and 10 charts for the MNIST data set. “Clustering space" indicates whether the clustering of the points attributed to the split chart was done in the input space (“Input"), the latent representation space (“Encoding"), or in the image of the hidden layers of the gating network (“Internal"). The clustering algorithm was either k-means or agglomerative clustering, and the performance of the charts was measured using either their total or average reconstruction error - see Section 5.3.2 for further explanations.

We observe that applying the chart splitting mechanism leads to reduced performance for all data sets compared to starting with a fixed and reasonable number of charts, probably because starting with a low number of charts gets the system stuck in a very suboptimal local optimum. None of the variations in procedure seem to have a significant impact.

For comparison, we also measured average reconstruction error on the MNIST data set when training for each class of digits a specialized autoencoder and using it to encode the points of this class (on the row “Manually organized charts"); this should correspond to an almost optimal chart distribution, where each chart is tasked with encoding a distinct class. We see that the performance is no better than that obtained in previous experiments (perhaps even slightly worse). We comment on this in Section 7.6.

Chart splitting mechanism					
Clust. space	Clust. alg.	Chart error	Torus & sphere	40-sphere	MNIST
Input	k-means	Total	0.0030	0.187	0.144
		Average	0.0024	0.191	0.167
	Agg. clust.	Total	0.0030	0.189	0.167
		Average	0.0031	0.189	0.553
Encoding	k-means	Total	0.0025	0.190	0.193
		Average	0.0030	0.188	6.288
	Agg. clust.	Total	0.0023	0.190	0.175
		Average	0.0040	0.187	0.195
Internal	k-means	Total	0.0023	0.188	0.147
		Average	0.0024	0.191	0.187
	Agg. clust.	Total	0.0024	0.191	0.180
		Average	0.0034	0.189	0.211
Manually organised charts					0.112

Figure 7.14 – Average reconstruction error L_E when starting with a single chart and repeatedly applying the chart splitting mechanism with various combinations of options. As a comparison, we include the average reconstruction error obtained on the MNIST data set with 10 charts, each trained and tested only on the pictures representing one of the 9 digits.

Finally, we test classical autoencoders of various sizes on the same data sets: the small ones have roughly the same number of parameters as a single one of the charts we used while the large ones have about as many parameters as a complete Deep Atlas [2]. We compare the best classical autoencoder configuration to Deep Atlas on the classification and UAD tasks of the next sections.

Performance of classical autoencoders of various sizes			
	Simple autoencoder size		
Data set	Small	Medium	Large
Torus union sphere	0.0337	0.0256	0.0192
40-sphere	0.171	0.204	0.251
Swiss roll	0.062	0.045	0.055
MNIST	0.109	0.066	1.004

Figure 7.15 – Average reconstruction error for various sizes of classical autoencoders.

[2] See Annex A for exact descriptions.

7.4 Classification

We encode the data sets using either Deep Atlas or a classical autoencoder, then apply a simple k-neighbour classification algorithm (using the scikit-learn implementation, see [Pedregosa et al., 2011]) to the latent representations. For Deep Atlas, we use a distinct classifier for each chart.

We see in Table 7.16 that Deep Atlas is about 8 and 6 times more precise than a similarly sized classical autoencoder on the union of the sphere and the torus and the Swiss roll, which leads to improved precision. On the other hand, Deep Atlas does not improve performance on the 40-dimensional sphere or the MNIST data set (and reconstruction error is even slightly worse for MNIST than with the best classical autoencoder).

Classification accuracy of classical autoencoders and Deep Atlas				
	Classical Autoencoder		Deep Atlas	
Data set	Prec.	R. error	Prec.	R. error
Torus union sphere	80.2	0.0206	88.3	0.0026
40-sphere	45.5	0.174	47.6	0.172
Swiss roll	69.9	0.0432	84.5	0.0071
MNIST	95.7	0.0637	95.1	0.093

Figure 7.16 – Classification precision (in $\%$ of correct answers) and average reconstruction error of simple autoencoders and Deep Atlas.

7.5 Unsupervised anomaly detection

As detailed in Annex A, we added 5% of noisy anomalies to the artificial data sets. For the MNIST data set, we used the classes 0, 1 and 2 as normal points, and added 10% of points from the other classes as anomalies.

We encode and reconstruct each training set, then use the norm of the reconstruction error as a measure of anomaly. More specifically, we let $\rho > 0$ be such that the percentage of data points whose reconstruction error is of norm greater than ρ is equal to the percentage of anomalies in the data set. We then classify a point as an anomaly if its reconstruction error is of norm greater than ρ.

We observe in Table 7.17 that Deep Atlas underperforms compared to the simpler classical autoencoders on all datasets.

UAD performance of classical autoencoders and Deep Atlas				
	Simple Autoencoder		Deep Atlas	
Data set	TP	TN	TP	TN
Torus union sphere	58.0	97.8	42.5	97.0
40-sphere	4.5	95.0	3.5	95.0
Swiss roll	45.0	97.1	27.0	96.2
MNIST	1.7	94.8	1.4	94.9

Figure 7.17 – True positive and true negative rates (in %) of simple autoencoders and Deep Atlas.

7.6 Discussion

Somewhat disappointingly, most hyperparameter choices seem to have little impact, though more systematic testing on wider variety of data sets might show some small effect.

As hoped for, Deep Atlas is much more precise than similarly sized classical autoencoders on simple manifolds, such as the Swiss roll or the union of the sphere and the torus. Regarding the 40-dimensional sphere, we suspect that we might have underestimated the complexity of the data set due to its mathematical simplicity: though we were not aiming for state of the art performance, a single hidden layer in the encoder and the decoder was probably largely insufficient to accurately encode it, leading to results too mediocre to allow meaningful comparisons.

We chose MNIST on purpose as an example of a data set with little manifold-like structure: indeed, manually distributing the charts in an a priori almost optimal way (one autoencoder per class) leads to almost no improvement upon using either Deep Atlas or a single big autoencoder. This seems to suggest that there is no easy way to split the data set into simple, easy to encode regions[3]; rather than our method failing, our starting assumptions regarding the structure of the data simply do not apply to MNIST.

The chart splitting mechanism seems to lead to worsened performance compared to simply starting with a reasonable number of charts. As this is due to the system getting stuck in bad local optima, a refinement in the training routine might help alleviate this problem. As suggested by the case of the union of the torus and the sphere, using the chart splitting mechanism might also beat starting with a much too low number of charts when training on a data set for which a reasonable number of charts is hard to guess.

Regarding the classification task, increased encoding precision seems to yield increased performance, as expected. This is not the case for unsupervised anomaly detection. Though it might seem counter-intuitive, it is a consequence of the considerations exposed in Chapter 3. As the number of charts increases, the volume of the regions where Deep

[3] Though it might be once again that deeper autoencoders would have been necessary to capture subtler structure.

Atlas precisely reconstructs points increases as well; consequently, though normal points are better reconstructed, so do some anomalies, and the latter effect seems to dominate[4].

The terrible true positive rates of both the classical autoencoder and Deep Atlas on MNIST illustrate the same problem: both models learn too much structure and become TOO proficient at extrapolating, and thus manage to faithfully reconstruct digits that are very rare in the data set.

[4] As we use a quantile-based anomaly score threshold, this impacts both the true positive AND the true negative rate.

Conclusion

We have shown the existence of a theoretical limitation on the precision of continuous autoencoders when encoding closed manifolds, illustrated how it can manifest, and discussed the effects it can have in the particular context of Unsupervised Anomaly Detection (UAD).

We have also analyzed the assumptions regarding the nature of anomalies underlying most UAD algorithms, and shown through a series of simple examples how they can be limiting. We think that keeping such considerations in mind might lead to more principled progresses in the field.

An important issue which we did not have the time or space to study is that of the nature of most real life data: though we have shown that closed manifolds can be a weakness of continuous autoencoders and that anomalies can a priori come in many forms and need not necessarily be outliers of the total distribution, such phenomena have only been seriously studied in some data sets from particle physics (see [Batson et al., 2021]). Measuring how common closed manifolds or "non-outlierish" anomalies are in typical data sets from various fields would be very valuable (though difficult), and help us decide whether those are minor epiphenomena, or whether they should be treated as a major issue.

We have also tried to suggest a solution to the problem of continuous autoencoders struggling to accurately model manifolds. Our model, Deep Atlas, significantly outperformed classical autoencoders in encoding precision on simple data sets meant to illustrate the type of structure for which it was designed. Conversely, in the complete absence of such structure (as with the MNIST data set), the gain in performance was essentially zero (though this might have been partially due to training difficulties and suboptimal architecture choices). We also observed that as predicted in Chapters 3 and 4, a gain in encoding precision does not necessarily translate to an improvement in UAD performance.

The next steps would be to test Deep Atlas on a wider variety of appropriate real world data sets using more finely tuned architectures in order to reach more definitive conclusions, as well as to further refine the algorithm itself.

Possible improvements could be:

- Replacing the gating network by another system tasked with attributing points to charts, e.g. using a k-means-like clustering subroutine. This might lead to a geo-

metrically more sensible distribution of points among the charts. Some Expectation-Maximisation-like training could also help the model avoid local minima.

- Further improving the adaptiveness of the system by adding a chart destroying or chart merging mechanism. Once a good system of dynamic chart management has been identified, we could also try integrating it to various mixture of experts models (adaptively adding or removing experts) used in tasks other than dimensionality reduction.
- Adding a probabilistic component to the autoencoders, in the same spirit as Variational Autoencoders (see [Kingma and Welling, 2014]).
- Finding a way to link the charts together: currently, each chart is entirely independent from the others, and there is no easy way to determine whether the regions encoded by two charts are adjacent or distant. Such information might be useful, for example when performing classification tasks.

Bibliography

[Abadi et al., 2015] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[Amer et al., 2013] Amer, M., Goldstein, M., and Abdennadher, S. (2013). Enhancing one-class support vector machines for unsupervised anomaly detection. pages 8–15.

[Balasubramanian and Schwartz, 2002] Balasubramanian, M. and Schwartz, E. (2002). Schwartz, e.l.: The isomap algorithm and topological stability. science 295, 5552. *Science (New York, N.Y.)*, 295:7.

[Bank et al., 2020] Bank, D., Koenigstein, N., and Giryes, R. (2020). Autoencoders.

[Batson et al., 2021] Batson, J., Haaf, C., Kahn, Y., and Roberts, D. (2021). Topological obstructions to autoencoding.

[Bergman and Hoshen, 2020] Bergman, L. and Hoshen, Y. (2020). Classification-based anomaly detection for general data. In *International Conference on Learning Representations*.

[Bishop, 2006] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

[Breunig et al., 2000] Breunig, M., Kriegel, H.-P., Ng, R., and Sander, J. (2000). Lof: Identifying density-based local outliers. volume 29, pages 93–104.

[Burda et al., 2016] Burda, Y., Grosse, R., and Salakhutdinov, R. (2016). Importance weighted autoencoders.

[Camastra and Staiano, 2016] Camastra, F. and Staiano, A. (2016). Intrinsic dimension estimation: Advances and open problems. *Information Sciences*, 328:26–41.

[Cavicchioli and Grasselli, 1985] Cavicchioli, A. and Grasselli, L. (1985). Minimal atlases of manifolds. *Cahiers de topologie et géométrie di érentielle catégoriques*, 26.

[Chalapathy and Chawla, 2019] Chalapathy, R. and Chawla, S. (2019). Deep learning for anomaly detection: A survey.

[Cortes et al., 2017] Cortes, C., Gonzalvo, X., Kuznetsov, V., Mohri, M., and Yang, S. (2017). AdaNet: Adaptive structural learning of artificial neural networks. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 874–883. PMLR.

[Curth and van der Schaar, 2021] Curth, A. and van der Schaar, M. (2021). Doing great at estimating cate? on the neglected assumptions in benchmark comparisons of treatment e ect estimators.

[Engel et al., 2012] Engel, D., Hüttenberger, L., and Hamann, B. (2012). A survey of dimension reduction methods for high-dimensional data analysis and visualization. *OpenAccess Series in Informatics*, 27:135–149.

[Eskin et al., 2002] Eskin, E., Arnold, A. O., and Prerau, M. (2002). A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data.

[Geng et al., 2015] Geng, J., Fan, J., Wang, H., Ma, X., Li, B., and Chen, F. (2015). High-resolution sar image classification via deep convolutional autoencoders. *IEEE Geoscience and Remote Sensing Letters*, 12(11):2351–2355.

[Goldstein and Dengel, 2012] Goldstein, M. and Dengel, A. (2012). Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm.

[Goldstein and Uchida, 2016] Goldstein, M. and Uchida, S. (2016). A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one*, 11:e0152173.

[Gorban et al., 2008] Gorban, A., Kégl, B., Wunsch, D., and Zinovyev, A. (2008). *Principal Manifolds for Data Visualisation and Dimension Reduction, LNCSE 58*.

[Hansen, 1999] Hansen, J. V. (1999). Combining predictors: comparison of five meta machine learning methods. *Information Sciences*, 119(1):91–105.

[Hatcher, 2005] Hatcher, A. (2005). *Algebraic Topology*. Cambridge University Press.

[He et al., 2003] He, Z., Xu, X., and Deng, S. (2003). Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9):1641–1650.

[Hinton and Salakhutdinov, 2006] Hinton, G. and Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science (New York, N.Y.)*, 313:504–7.

[Jacobs et al., 1991a] Jacobs, R. A., Jordan, M. I., and Barto, A. G. (1991a). Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks. *Cognitive Science*, 15(2):219–250.

[Jacobs et al., 1991b] Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991b). Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87.

[Johnstone and Titterington, 2009] Johnstone, I. and Titterington, D. (2009). Statistical challenges of high-dimensional data. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 367:4237–53.

[Jolli□e, 2002] Jolli□e, I. (2002). *Principal Component Analysis*. Springer Series in Statistics. Springer.

[Kingma and Ba, 2014] Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

[Kingma and Welling, 2014] Kingma, D. and Welling, M. (2014). Auto-encoding variational bayes.

[Kramer, 1991] Kramer, M. A. (1991). Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243.

[Kruskal, 1964] Kruskal, J. (1964). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27.

[LeCun et al., 2010] LeCun, Y., Cortes, C., and Burges, C. (2010). Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2.

[Lee et al., 2004] Lee, A. B., Pedersen, K., and Mumford, D. (2004). The nonlinear statistics of high-contrast patches in natural images. *International Journal of Computer Vision*, 54:83–103.

[Martí et al., 2015] Martí, L., Sánchez-Pi, N., Molina, J., and Garcia, A. C. (2015). Anomaly detection based on sensor data in petroleum industry applications. *Sensors*, 15:2774–2797.

[Masoudnia and Ebrahimpour, 2014] Masoudnia, S. and Ebrahimpour, R. (2014). Mixture of experts: A literature survey. *Artificial Intelligence Review*, 42.

[Melas-Kyriazi, 2020] Melas-Kyriazi, L. (2020). The mathematical foundations of manifold learning. *CoRR*, abs/2011.01307.

[Michor et al., 2008] Michor, P., Scharlemann, M., Cox, D., Krantz, S., and Mazzeo, R. (2008). *Topics in Di erential Geometry*. Graduate studies in mathematics. American Mathematical Society.

[Munkres, 2000] Munkres, J. (2000). *Topology*. Topology. Prentice-Hall.

[Nowlan and Hinton, 1991] Nowlan, S. and Hinton, G. E. (1991). Evaluation of adaptive mixtures of competing experts. In Lippmann, R. P., Moody, J., and Touretzky, D., editors, *Advances in Neural Information Processing Systems*, volume 3. Morgan-Kaufmann.

[Outerelo and Ruiz, 2009] Outerelo, E. and Ruiz, J. (2009). *Mapping Degree Theory*. Graduate studies in mathematics. American Mathematical Soc.

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

[Pitelis et al., 2013] Pitelis, N., Russell, C., and Agapito, L. (2013). Learning a manifold as an atlas*. *IEEE Conference in Computer Vision and Pattern Recognition*.

[Rainforth et al., 2018] Rainforth, T., Kosiorek, A. R., Le, T. A., Maddison, C. J., Igl, M., Wood, F., and Teh, Y. (2018). Tighter variational bounds are not necessarily better. In *ICML*.

[Ramaswamy et al., 2000] Ramaswamy, S., Rastogi, R., and Shim, K. (2000). E□cient algorithms for mining outliers from large data sets. volume 29, pages 427–438.

[Roweis and Saul, 2000] Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326.

[Sagi and Rokach, 2018] Sagi, O. and Rokach, L. (2018). Ensemble learning: A survey. *WIREs Data Mining and Knowledge Discovery*, 8(4):e1249.

[Sakurada and Yairi, 2014] Sakurada, M. and Yairi, T. (2014). Anomaly detection using autoencoders with nonlinear dimensionality reduction. pages 4–11.

[Schlegl et al., 2017] Schlegl, T., Seeböck, P., Waldstein, S., Schmidt-Erfurth, U., and Langs, G. (2017). Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. pages 146–157.

[Schölkopf et al., 1998] Schölkopf, B., Smola, A., and Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319.

[Shazeer et al., 2017] Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. https://arxiv.org/abs/1701.06538.

[Sorzano et al., 2014] Sorzano, C., Vargas, J., and Montano, A. (2014). A survey of dimensionality reduction techniques.

[Steinbach et al., 2003] Steinbach, M., Ertöz, L., and Kumar, V. (2003). The challenges of clustering high dimensional data. *Univ. Minnesota Supercomp. Inst. Res. Rep.*, 213.

[Tang et al., 2002] Tang, B., Heywood, M., and Author, M. (2002). Input partitioning to mixture of experts. *Proceedings of the International Joint Conference on Neural Networks*, 1:227 – 232.

[Tenenbaum et al., 2000] Tenenbaum, J. B., Silva, V. d., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323.

[Thudumu et al., 2020] Thudumu, S., Branch, P., Jin, J., and Singh, J. (2020). A comprehensive survey of anomaly detection techniques for high dimensional big data. *Journal of Big Data*, 7.

[Turk and Levoy, 1994] Turk, G. and Levoy, M. (1994). Zippered polygon meshes from range images. *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*.

[van der Maaten and Hinton, 2008] van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605.

[Vincent et al., 2008] Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 1096–1103, New York, NY, USA. Association for Computing Machinery.

[Xu et al., 2018] Xu, H., Chen, W., Zhao, N., Li, Z., Bu, J., Li, Z., Liu, Y., Zhao, Y., Pei, D., Feng, Y., Chen, J. J., Wang, Z., and Qiao, H. (2018). Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. *Proceedings of the 2018 World Wide Web Conference*.

[Yoon et al., 2021] Yoon, J., Sohn, K., Li, C.-L., Arik, S., Lee, C.-Y., and Pfister, T. (2021). Self-trained one-class classification for unsupervised anomaly detection.

[Yoon et al., 2020] Yoon, J., Zhang, Y., Jordon, J., and van der Schaar, M. (2020). Vime: Extending the success of self- and semi-supervised learning to tabular domain. *Part of Advances in Neural Information Processing Systems*, 33.

[Zong et al., 2018] Zong, B., Song, Q., Min, M. R., Cheng, W., Lumezanu, C., Cho, D., and Chen, H. (2018). Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International Conference on Learning Representations*.

Appendix A

Additional details on the experimental protocols

A.1 Datasets

We provide additional details on the generation of the artificial experimental data sets.

A.1.1 Torus union sphere

The first artificial data set is the union in $\mathbb{R}^3$ of a torus and a sphere. The points x on the torus are obtained by uniformly sampling $\phi \in [0, 2\pi], \theta \in [0, 2\pi]$ and setting

$$x = ((\cos(\theta) + 2)\cos(\phi), (\cos(\theta) + 2)\sin(\phi), \sin(\theta)) .$$

The points on the sphere are obtained by uniformly sampling from a three-dimensional normal distribution, and dividing the samples by their norm. A Gaussian noise of covariance $0.005 * I_3$ (where I_3 is the 3×3 identity matrix) is added to each point. We training and the test set both contain 2500 points on the torus and 1000 on the sphere. We also artificially attribute classes to the points using a mixture of Gaussians model, where each Gaussian corresponds to a class[1]. As a result, the classes are organized in a "geometrically coherent" fashion using , i.e. in such a way that each class roughly corresponds to a region of space, as illustrated at the top of Figure A.1. In the training set, the five classes contain $781, 674, 300, 1367$ and 378 points respectively - proportions are similar in the test set. Anomalies for the UAD experiment were sampled from a Gaussian of mean $(3, 0, 0)$ and covariance $20 \cdot I_3$.

[1] Details can be found in the code.

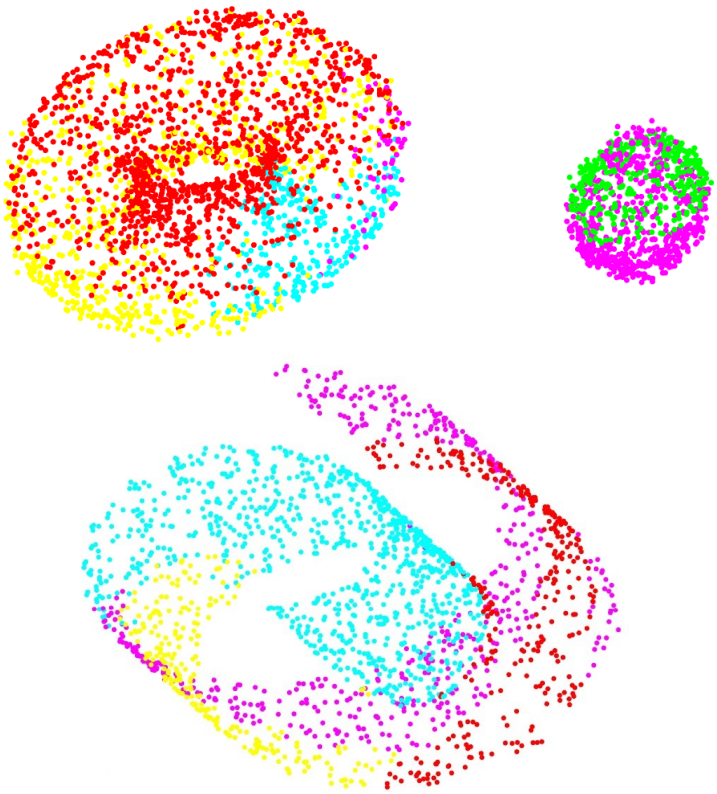


Figure A.1 – At the top, the union of a sphere and a torus. At the bottom, the Swiss roll data set. Colors indicate classes used for the classification experiments.

A.1.2 2-dimensional sphere

The two dimensional sphere used to illustrate the main mechanisms of Deep Atlas was obtained in the same way as the sphere component of the union of the sphere and the torus described above. It contained 1000 samples.

A.1.3 Swiss roll

We use the scikit-learn library [Pedregosa et al., 2011] to sample points on the Swiss roll manifold (plus a Gaussian noise of covariance $0.05 * I_3$), with 2000 points in the training set and the test set each. Points are divided into four classes using a mixture of Gaussians[2], as is illustrated at the bottom of Figure A.1. In the training set, the classes contain $242, 510, 928$ and 320 points respectively. Anomalies for the UAD experiment were sampled from a Gaussian of mean 0 and covariance $100 \cdot I_3$.

A.1.4 40-dimensional sphere

We obtain points on the sphere of dimension 40 embedded in $\mathbb{R}^{80}$ by sampling from a Gaussian of covariance

$$C = \begin{bmatrix} I_{41} & 0_{41\times 39} \\ 0_{39\times 41} & 0_{39\times 39} \end{bmatrix}$$

and dividing the samples by their norm, where I_n is the identity $n \times n$ matrix and $0_{n\times m}$ is the $n \times m$ zero matrix. A Gaussian noise of covariance $0.05 * I_{80}$ is then added. There are 1000 points in the training set and another 1000 in the test set. As above, the samples are split into classes using a mixture of Gaussians. In the training set, the classes contain $159, 564$ and 277 points respectively, and the proportions are similar in the test set. Anomalies for the UAD experiment were obtained by sampling points on the sphere and adding a Gaussian noise of mean 0 and covariance $0.005 \cdot I_{80}$.

A.2 Model configuration and training

We detail the architecture of the charts and gating network chosen for each data set in our quantitative experiments, as well as the training procedure.

Our neural networks used Scaled Exponential Linear Units (SELU) activation functions for their hidden layers, and linear activation for their output layers. With each data set, we used the Tensorflow implementation of the Adam optimizer (see [Abadi et al., 2015] and [Kingma and Ba, 2014]), with an initial learning rate of 10^{-3} which we divided by 10 after each epoch of training. Batch size was set to 1000.

[2] Details can be found in the code.

A.2.1 Torus union sphere

Both our encoders and decoders had 3 hidden layers of 7 units. The gating network had two hidden layers of 10 units each. When testing various simple autoencoders sizes, the encoder and decoder of the small one had 3 hidden layers of 7 units each. The encoder and decoder of the medium one had 3 hidden layers of 21 units each, and the the encoder and decoder of the large one had 3 hidden layers of 42 units each. We ran 3 training epochs of $2000, 4000$ and 8000 passes respectively (both for Deep Atlas and for the simple autoencoders).

A.2.2 Swiss roll

The architectures chosen for the Swiss roll are exactly the same as for the union of the sphere and the torus.

A.2.3 40-dimensional sphere

Both our encoders and decoders had 1 hidden layers of 40 units. The gating network had 0 hidden layer. When testing various simple autoencoders sizes, the encoder and decoder had 1 hidden layers of 40 (respectively 60 and 80) units each for the small autoencoder (respectively the medium and large ones). We ran 3 training epochs of $2000, 4000$ and 8000 passes respectively.

A.2.4 MNIST

The encoder had 2 hidden layers of 250 and 100 hidden units respectively; the decoder had 2 hidden layers of 100 and 250 hidden units respectively (we followed [Burda et al., 2016] regarding what constitutes a “reasonable" configuration). The gating network had 2 hidden layers of 250 and 50 units each. When testing various simple autoencoders sizes, the encoder (respectively decoder) of the small one had 2 hidden layers of 250 and 150 units (respectively 2 hidden layers of 100 and 200 units). The encoder (respectively decoder) of the medium one had 2 hidden layers of 750 and 300 units (respectively 2 hidden layers of 300 and 600 units). The encoder (respectively decoder) of the large one had 2 hidden layers of 2500 and 1500 units (respectively 2 hidden layers of 1000 and 2000 units). We ran 3 training epochs of $500, 2000$ and 4000 passes respectively.