UNIVERSITY OF THE PELOPONNESE & NCSR "DEMOCRITOS" MSC PROGRAMME IN DATA SCIENCE

DEEP LEARNING FOR CLUSTERING OF WEATHER AND ATMOSPHERIC DISPERSION FEATURES

Tsiatsios Georgios
DSC17024

A thesis submitted in partial fulfillment
of the requirements for the MSc
in Data Science

Supervisor: Iraklis A. Klampanos

Athens, October 2019

UNIVERSITY OF THE PELOPONNESE & NCSR "DEMOCRITOS" MSC PROGRAMME IN DATA SCIENCE

DEEP LEARNING FOR CLUSTERING OF WEATHER AND ATMOSPHERIC DISPERSION FEATURES

Tsiatsios Georgios
DSC17024

A thesis submitted in partial fulfillment
of the requirements for the MSc
in Data Science

Supervisor: Iraklis A. Klampanos

Approved by the examination committee on October,2019.

(Signature)	(Signature)	(Signature)
.....	.....	.....
I.A. Klampanos	Georgios Paliouras	Konstantinos Limniotis

Athens, October 2019

Declaration of Authorship

(1) I declare that this thesis has been composed solely by myself and that it has not been submitted, in whole or in part, in any previous application for a degree. Except where states otherwise by reference or acknowledgment, the work presented is entirely my own.

(2) I confirm that this thesis presented for the degree of Bachelor of Science in Informatics and Telecommunications, has

(i) been composed entirely by myself

(ii) been solely the result of my own work

(iii) not been submitted for any other degree or professional qualification

(3) I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or processional qualification except as specified.

(Signature)

.....

Georgios Tsiatsios

Athens, October 2019

Περίληψη

Η μείωση διαστάσεων μειώνει σημαντικά το "βάρος" των υπολογισμών σε εργασίες που αφορούν την βελτιστοποίηση και την προγνωστική ικανότητα ενός μοντέλου. Ως εκ τούτου, οι δημοφιλείς προσεγγίσεις μείωσης διαστάσεων δεδομένων στοχεύουν στην προβολή των δεδομένων σε ένα μικρότερο υπο-χώρο που αποτελείται από γραμμικές και μη-γραμμικές σχέσεις που λαμβάνονται από την συμπίεση ενός συνόλου δεδομένων στιγμιότυπων λύσης.

Αυτή η εργασία προσεγγίζει το πρόβλημα λήψεως βαρυσήμαντων αναπαραστάσεων χαμηλών-διαστάσεων από υψηλών-διαστάσεων δεδομένα, που αναφέρεται ως μείωση διαστάσεων, σχεδιάζοντας και εφαρμόζοντας κατάλληλα βαθιά νευρωνικά δίκτυα. Επιπλέον, αυτή η εργασία στοχεύει στην ικανότητα των αλγορίθμων να μάθουν την κρυφή πολλαπλή σχέση των δεδομένων προκειμένου να διευκολύνουν περαιτέρω την επεξεργασία τους. Η καινοτομία της προτεινόμενης τεχνικής είναι ο σχεδιασμός αυτόματων κωδικοποιητών που χρησιμοποιούν τις μεταβλητές του καιρού και την χρονική συνέχεια, προκειμένου να μάθουν τις αναπαραστάσεις των δεδομένων για την ομαδοποίηση δεδομένων καιρικών συνθηκών. Εξετάζουμε μια πληθώρα μοντέλων που βελτιώνουν τη διαδικασία ομαδοποίησης μετεωρολογικών δεδομένων πολλών ετών στην ευρωπαϊκή ήπειρο.

Ένας αριθμός πραγματικών εφαρμογών μπορεί να επωφεληθεί από τέτοια μοντέλα, για παράδειγμα η αναγνώριση ανωμαλίας σε ανανεώσιμες πηγές ενέργειας χρησιμοποιώντας χαρακτηριστικά ροής ανέμου και ηλιακής ενέργειας με σκοπό την υλοποίηση συστημάτων παρακολούθησης. Επιπροσθέτως, είναι δυνατή η υλοποίηση συστημάτων ανάστροφων ατμοσφαιρικών μοντέλων με σκοπό την ανάλυση ατμοσφαιρικών και κλιματικών αλλαγών για την εκτίμηση πηγών ρύπων ή ροής επιβλαβών αερίων. Η γνώση μας στην μετεωρολογία για γεωργικούς σκοπούς μπορεί να ενισχυθεί από τέτοιου είδους δομές συλλέγοντας τα επικρατώντα μετεωρολογικά πρότυπα καιρού.

Από τους ποικίλους τρόπους εκμάθησης αναπαραστάσεων η εργασία αυτή εστιάζει σε βαθιά γνώσης μεθόδων που αποτελούνται από μια σύνθεση μη-γραμμικών μετασχηματισμών, με στόχο την απόδοση πιο αφηρημένων - και τελικά πιο χρήσιμων - αναπαραστάσεων. Παρουσιάζουμε μια εναλλακτική μη-γραμμική μέθοδο αναπαράστασης, όπως τα Convolution και LSTM (Long Short Term Memory) δίκτυα και παραθέτουμε τα πλεονεκτήματα και τα μειονεκτήματα τους εστιάζοντας στην εξαγωγή επαναχρησιμοποιήσιμων και εύρωστων βαθιών χαρακτηριστικών. Αυτοί οι μέθοδοι είναι γνωστό ότι είναι σε θέση να ξεμπερδέψουν τις πολυποίκιλες σχέσεις δεδομένων, δηλαδή να ανιχνεύουν τις λανθάνουσες διαστάσεις καθώς και την χρονική και / ή χωρική πλευρά των δεδομένων. Ωστόσο, η εφαρμογή τους στην ομαδοποίηση δεδομένων για τις καιρικές συνθήκες δεν έχει έως τώρα μελετηθεί επαρκώς.

Τα μοντέλα αξιολογούνται και συγκρίνονται με εναλλακτικές μεθόδους παρουσιάζοντας ανταγωνιστικά αποτελέσματα. Αποδεικνύουν ότι μπορούν να αντιστοιχήσουν τα εισαγόμενα δεδομένα σε μια νέα διάσταση όπου είναι γραμμικά διαχωρίσιμα. Επίσης, οι εξαγόμενες αναπαραστάσεις μπορούν να εφαρμοστούν σε πληθώρα τομέων όπως στην ανακάλυψη πρότυπων καιρικών συνθηκών, υλοποίηση εφαρμογών όπως εφαρμογές έκτακτης ανάγκης στην ατμοσφαιρική φυσική, ραδιολογία, περιβαλλοντική έρευνα, υγεία και άλλα

Λέξεις Κλειδιά: Μείωση διαστάσεων, μη επιβλεπόμενη μάθηση, πολυποίκιλη μάθηση, βαθιά μάθηση, ομαδοποίηση καιρού.

ABSTRACT

Dimensionality reduction alleviates computational burdens faced in various tasks from design optimization to predictive modelling. Hence, popular dimensionality reduction approaches aim at projecting the data onto a subspace spanned by linear and non-linear functions obtained from the compression of a dataset of solution snapshots.

This thesis approaches the problem of obtaining meaningful low-dimensionality representations of high-dimensional data, referred to as dimensionality reduction, by the design and application of suitable deep neural network configurations. Moreover, in the applications targeted by this dissertation it is crucial for the algorithms to learn the underlying data manifold in order to facilitate further processing. The novelty of the proposed technique is to design auto encoders that make use of weather variables and temporality, in order to learn enhanced data representations for weather data clustering. We examine a plethora of end-to-end trainable models improving clustering procedure on multiple years of weather data in the European continent.

A number of real-world applications could benefit from such models, for instance anomaly detection in renewable energy sources utilizing weather features from wind-flow and solar energy in order to develop monitoring systems. Furthermore, scenarios including environment and climate change could be developed for atmospheric inverse analysis systems to estimate emissions or gas flux sources. Our knowledge in meteorology for agricultural purposes could be enhanced from such frameworks by collecting dominant meteorological patterns.

Among the various ways of learning representations, this thesis focuses on deep learning methods: those that are formed by the composition of multiple non-linear transformations, with the goal of yielding more abstract – and ultimately more useful – representations. We present alternative non-linear representation learning methods, such as Convolutional and LSTM (Long Short Term Memory) networks and discuss advantages and disadvantages, focusing on extracting reusable and robust deep features. These methods are known to be able to disentangle the data manifolds, i.e., to uncover the latent dimension(s) along which the temporal and/or spatio-temporal

aspect of the data. However, their application on the clustering of weather data has not been sufficiently studied before.

The models are evaluated and compared to alternative methods demonstrating competitive results. They are shown to map input data into a new space where the data are linear separable. Hence, the extracted representations can be applicable in a plethora of domains, such as in discovering robust weather patterns, developing emergency response applications in atmospheric physics, radiology, environmental research, healthcare and in other application areas.

Keywords: Unsupervised learning, Dimensionality reduction, manifold learning, Deep learning, weather clustering

Contents

List of Tables

1 Introduction

1.1 Problem Formulation

Weather is a complex system. It is described by several elements such as wind-flow, solar, pressure also involving spatial and temporal dependencies. The observations originate from satellites and/or ground sensors. Statistical and especially deep-learning models of weather data could be beneficial in multitude domains such as environmental research, weather and climate forecasting and atmospheric modelling as well.

Often, we would like to extract meaningful weather representations for clustering or other unsupervised-learning applications, from raw weather data. A former study in weather representation learning [1] applies convolutional auto-encoder for clustering exploiting the spatial aspect of the dataset. Aim of this thesis is to improve the former research by developing a framework including the temporality of the data that learn linearly separable representations. Apart from limiting the effectiveness of the deep learning model, the presence of spatial and temporal information makes it possible to consider novel formulations for analysing data. The purpose of this study is to discover relationships and patterns in weather science that advance our understanding of the weather system and help us better prepare for future adverse conditions by informing adaptation and mitigation actions in a timely manner.

Emergency response weather-based applications events are challenging. The identification of weather patterns is important for risk management, informing governmental policy decisions, aid environment protection and advancing our basic understanding of the weather system. Typically, are modelled from large-scale and multi-dimensional data, containing both spatial and temporal features tightly coupled (i.e. the geo-potential height pressures and wind). Any model that aims to use that information should cope with the high dynamics of the data. Our target is to discover structural patterns that elucidate complex spatial and temporal dynamics. Furthermore, we aim to learn the complexity of the data which will lead in discovering meaningful low dimensional spatial-temporal feature representation. This approach will optimize the unsupervised weather clustering objective in order to identify predominant weather events for rapid source estimation during radiological releases.

Nowadays there are many different approaches to learning from data, and the study of these learning algorithms is called machine learning. Real world problems require deriving function estimates from a large number of data sets with many variables. Although having access to abundance of examples with many features is beneficial to an algorithm attempting to generalize from data, handling large number of variables or dimensions often adds to the complexity of the problem making it difficult to perform useful inference.

It is well known that high dimensionality presents obstacle to efficient processing and use of data, which phenomenon is often referred as curse of dimensionality. The heart of the challenge is that as dimensionality D increases the volume of the space increase too fast so that the available data becomes more complex. In many cases high dimensionality is an artefact of the choice of data representation which has nothing to do with the underlying complexity mechanisms that generated the data. Often the variables involved in the representation are correlated through some functional dependence, hence the number of independent variables necessary to efficiently describe the data is small. In such scenario it is possible to represent the data in much fewer dimensions than the dimensions of the original data, and this is referred as dimensionality reduction. Developing such methods of dimensionality reduction and studying their properties is the main goal of the field manifold learning.

Recently several linear and non-linear approaches have been proposed in unsupervised learning applications in order to derive meaningful low-dimensional representations of high-dimensional data to produce good features. Among non-linear methods, such as auto-encoders, have recently attracted great attention by providing noteworthy results on real world data sets. In this thesis combines the dynamics of the machine and deep learning models by creating an end-to-end framework for weather analysis. Several architectures are presented in order to assess their effectiveness for dimensionality reduction and manifold learning in large weather and atmospheric dispersion datasets to determine the key variables endowed with the most explanatory power determining weather patterns, which affect plume behaviour dispersions of a radiation event.

1.2 Outline

This document is organized as follows: Chapter 2 discusses the prior work relative to the thesis domain and its influence on the proposed design. Chapter 3 discusses the proposed architectures, their various implementations, and the evaluation algorithm. The proposed architectures include a LSTM auto-encoder and a Conv-LSTM encoder-decoder model. Chapter 4 details the experiment and our methodology summarizing the potential and effectiveness of our system. Chapter 5 refers to the results of the proposed framework and finally Chapter 6 to the conclusions and future work.

2 Related Work

2.1 Deep Learning

Deep Learning describes a set of practices and algorithms for numerous architectures of deep neural networks, where the term deep refers to architectures consisting of multiple hidden layers. With these deep neural networks, the goal is often to derive hierarchal hidden representations of raw input data in order to solve a narrow task. As an example, in computer vision applications, deep convolutional networks are trained to detect different visual features from given images to categorize objects. This example shows the advantage of neural networks compared to traditional machine learning algorithms like support vector machines (SVM). In general, deep neural networks can learn latent features from raw data, whereas in case of traditional learning algorithms, the input features have to be carefully engineered which often requires extensive domain knowledge.

This practical advantage of deep learning algorithms offers high potential in use cases, where relevant input features cannot be manually defined due to lack of domain knowledge. In some cases, extracting features can also be too complicated to be encoded by an engineer. This can easily be applied to the task of object recognition. In this case, a human being can identify objects intuitively but cannot easily derive a complete set of rules for an algorithm to identify specific objects with invariance to scale, orientation, or the position in an image.

Simple machine learning algorithms have proven to be unsuccessful when it comes to solving tasks like object or speech recognition, which are considered as central problems in artificial intelligence [2]. Especially on data with a high dimensional input space, simple algorithms cannot generalize sufficiently due to the sheer amount of possible different configurations of the input data, which is often much larger than the available training samples.

Recent deep learning publications often reach state-of-the-art performance in many different tasks, which have been subject to active research within the last decades. For

example, deep architectures of convolutional neural networks revolutionized the field of image recognition and ever since have been the first choice for the task of object classification with constantly achieving convenient results.

The recent achievements in deep learning applications result in major attention from media, which influences public expectations towards artificial intelligence. However, many successful applications of deep learning are limited to a very narrow task, whereas transfer of knowledge and incorporation of context remains a subject to be further explored to achieve actual progress towards general artificial intelligence.

2.2 Artificial Neural Networks (ANNs)

Artificial Neural Networks (ANNs) are a family of statistical learning models inspired by biological neural networks and are used to estimate or approximate functions that can depend on a large number of inputs and are generally unknown. It is presented as a network of interconnected neurons whose connections have numeric weights that can be tuned based on experience. It makes the neural nets adaptive to inputs and capable of learning.

Such a network is composed by an input layer, a number of hidden layers and an output layer. The activation of the neurons in the input layer corresponds to the input vector such as Images, word vector for machine learning translation. A weight matrix, followed by a non-linear activation function, then transforms the vector in another representation. The output vector of the first layer is the input vector of the second, and so on until the output layer is reached. The activation of the neurons in the output layer may represent classes, probability distributions, or the estimated value of an unknown function to be learned.

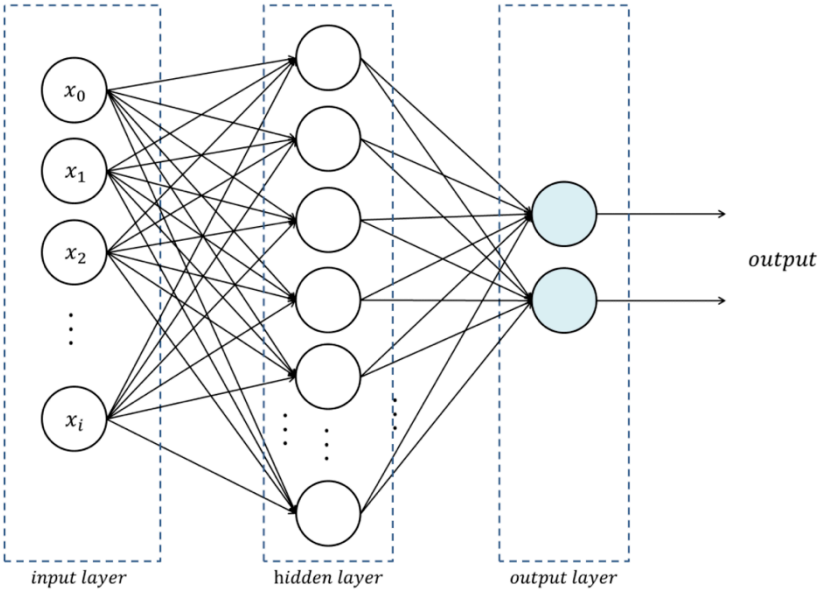

Figure 1 Feed Forward Network

In a feed-forward network, the connections go from one layer to the next, i.e. information only goes in one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. Every neuron weights are updated through a back propagation algorithm that aims to minimize the error between network output and target value. Since the network learns its own weights, it is able to determine on its own what features are important and applicable to the task. While a strong tool, feed forward neural networks utilizing only neurons are not as effective on problems that rely on dynamic spatial or temporal information. This weakness has led to the development of Convolutional and Recurrent Neural Networks [3]. Unlike feed-forward neural networks, an RNN can use its internal memory to process arbitrary sequences of inputs.

2.3 Long Short-Term Memory (LSTM)

A Long Short-Term Memory Network is a recurrent artificial neural network with some extensions. The idea of a recurrent neural network is that not only information only provided through the input layer but the hidden layer of a previous iteration can now pass information to future iterations. How the information is fed back into the network will change the structure of the network but the general idea of recurrent networks is just to feed information back into itself.

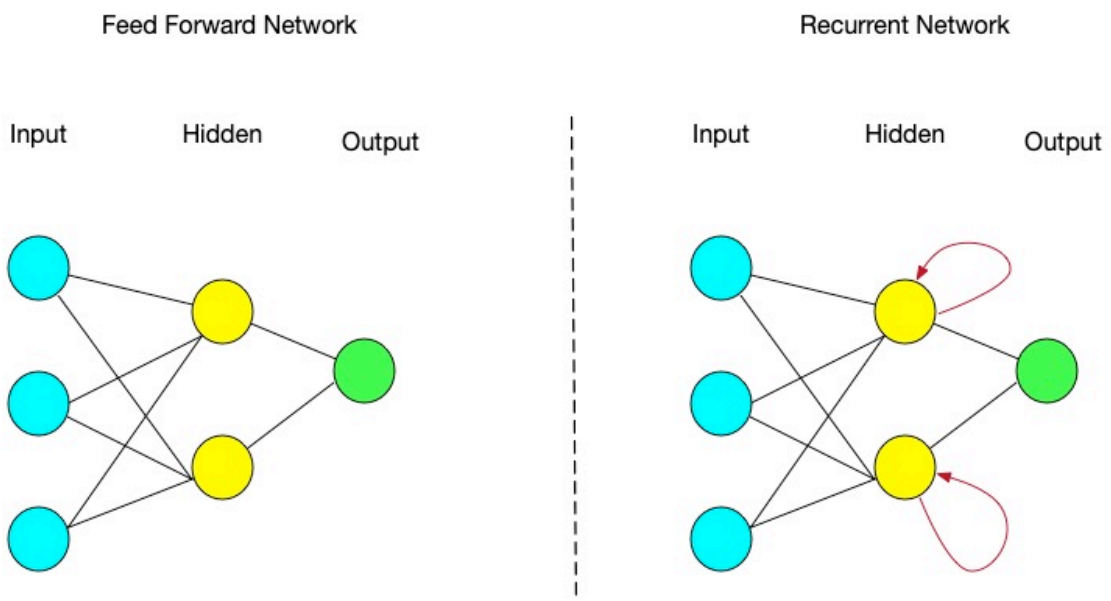

Figure 2 Feed Forward vs Recurrent Network

The intuition behind such a network is that sequences and series are now utilized. An element of time has been introduced, which was not present in the feed-forward network presented above. The basic idea is to add paths through time, that cannot have exploding or vanishing gradients [2]. Hence, a very special version of a recurrent neural network is the Long Short-term Memory network or LSTM which has been introduced by Hochreiter and Schmidhuber in 1997 [4] and further enhanced by Gers et al. (2000) [5]. LSTM networks incorporate gating mechanisms to enable the model to decide whether to accumulate or forget certain information regarding the transferred cell state. This allows the network to operate at different timescales and therefore effectively model short as well as long-term dependencies. For instance, the model can store information on a pattern that is based on several different characteristics occurring in a relatively small time frame. Once the pattern is complete, the model can discard detailed information about the previously recorded characteristics and is therefore able to detect the next, similar type of pattern. This capability allows to efficiently solve several kinds of tasks that require sequence-modelling with temporal dependencies.

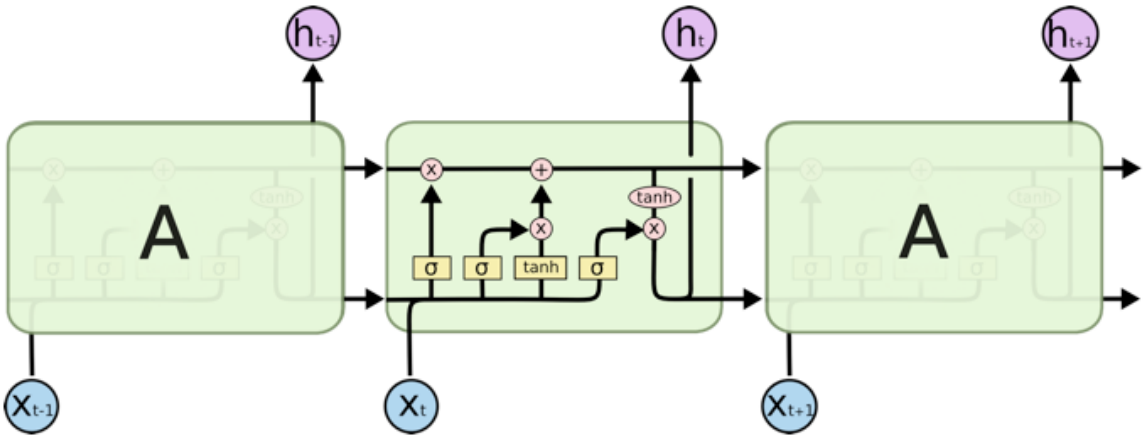

Figure 3 LSTM architecture

The figure above shows the LSTM *cells* per time step[1].The input gating mechanisms are implemented through weighted connections with sigmoid activations, as the sigmoid function converges at 0 or 1 and can therefore be seen as a differentiable binary decision between true (1) and false (0). This allows the forget gate f to update the cell's state. As an example, the gate can force certain information in the state to be forgotten by setting it to zero.

Hence, adding a forget gate allows the model to discard irrelevant information from the previous cell state by evaluating the information given by the input at the current time step t. The output of the forget gate is calculated in the following manner, where W is weight matrix whose parameters are to be learned during the training of the model, h refers to the previous hidden state and x_t in the input:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

Figure 4 LSTM Output of forget gate

The model can learn to accumulate certain information C_t from the current time step by taking into account the previous output. In a similar manner to the forget gate, the update gate I_t subsequently decides which information from the current time step will be added to the cell state.

[1] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$C' = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

Figure 5 LSTM Input gate layer

Finally, the updated state c can be calculated out of the previous state C_{t-1}, the output of the forget gate f and the output of the update gate it as stated in equation below:

$$C_t = f_t * C_{t-1} + i_t * C'_t$$

Figure 6 LSTM Cell State

Here, the * operation denotes the element-wise vector product. The output of the cell h_t at the current time step is subsequently calculated with the updated cell states c_t:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o$$

$$h_t = \sigma * \tanh(C_t)$$

Figure 7 LSTM Hidden / Output gate

LSTMs proved to perform well in many recent publications and are rather easy to train. Therefore, LSTMs have become the baseline architecture for tasks, where sequential data with temporal information has to be processed.

2.4 Convolutional Neural Networks

A Convolutional neural network (CNN) is a neural network architecture that is especially suited for 2- and 3- dimensional data structures, e.g. images. CNN architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture.

Unlike the Feed Forward Networks portrayed before, Convolutional Neural Networks take advantage of the fact that the input consists of images and they constrain the architecture in a more sensible way. In particular, unlike a regular Neural Network, the layers of a ConvNet have neurons arranged in 3 dimensions: width, height, depth

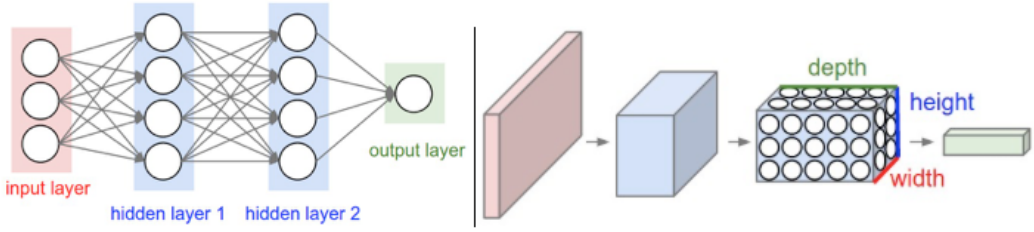

Figure 8 Left: Feed Forward Network, Right: Convolution with 3 dimensions[2]

These neurons then make the forward function more efficient to implement and vastly reduce the amount of parameters in the network. This allows to recognize certain patterns in the input data.

To achieve this, a weighted kernel K[3] is moved over every possible position in the input image. This is computationally equivalent to a 2-dimensional convolution of the input image and the kernel. For an image I the convolution can be thus written as follows, where the asterisk $*$ denotes the convolution operation:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, j-n) K(m,n)$$

Figure 9 Convolution kernel function

The resulting image S is referred to as feature map. The dimension of the feature map depends on the step size (stride) for shifting the kernel over the image. In order to retain the size of the original image, zero padding of the input image and a stride of 1

[2] http://cs231n.github.io/convolutional-networks/#architectures
[3] Usually, width and height of a kernel are equal. Also, the smallest reasonable kernel would be of size 3 × 3, as this contains a center pixel with one pixel on every surrounding side.

can be used. The weights of a kernel are estimated during the training of the model. Typically, a single convolutional layer contains k different kernels that are applied to the same input image and therefore result in k different feature maps. This allows extracting feature maps, which are sensitive to certain visual features and invariant to their position in the image.

By stacking multiple convolutional layers, the model learns to extract hierarchical visual features. The first feature maps can e.g. detect edges in the original image, whereas the following feature maps detect patterns, composed of the previously extracted a more abstract features.

Each convolutional layer is usually followed by a non-linear transfer function. In most applications, the architecture of CNNs includes pooling layers for down sampling of the intermediate feature maps. Alternatively, this can be achieved solely by stride convolutions as shown in the all-convolutional architecture proposed by Springenberg et al. (2015) [6].

2.5 Auto-Encoders

An auto-encoder is an artificial neural network composed of n input and output units and m hidden units. It is used for learning efficient representations [7] [8][9]. The aim of an auto-encoder is to learn a distributed representation (encoding) for a set of data. An auto-encoder is trained to encode the input x into some representation z so that the input can be reconstructed from that representation. It is thus a generative model. Hence the target output of the auto-encoder is the auto-encoder input itself. If there is one linear hidden layer and a loss criterion is used, such as mean squared error, to train the network, then the k hidden units learn to project the input in the span of the first k principal components of the data [10]. If the hidden layer is non-linear, the auto-encoder behaves differently from Principal Component Analysis (PCA), with the ability to capture multi-modal aspects of the input distribution [11]. The hope is that the code z is a distributed representation that captures the main factors of variation in the data: because Z is viewed as a "*lossy"* representation of X, it cannot be a good representation (with small loss) for all X. So learning drives it to be one, that is a

good representation in particular for training examples, and hopefully for others as well (and that is the sense in which an auto-encoder generalizes), but not for arbitrary inputs.

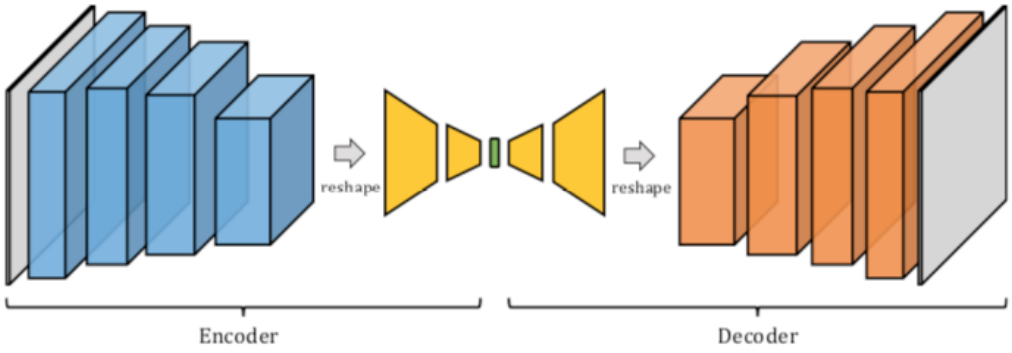

Figure 10 Visual Example of Auto-Encoder

The auto-encoder can typically be used for dimensionality reduction by learning a compressed representation of the data. Another application is feature extraction before classification or prediction, for which we want a higher dimensionality for easier separability. One serious issue with this approach is that if there is no other constraint, then an auto-encoder with n - dimensional input and an encoding of dimension $n \geq m$ could potentially just learn the identity function. There are different ways that an auto-encoder with more hidden units than inputs could be prevented from learning the identity, and still capture something useful about the input in its hidden representation Z. The strategies are the following:

Sparse auto-encoders. One strategy, based on the concept of sparse coding, is to add a sparsity constraint on the code. While an ordinary auto-encoder has an encoder part which computes P(z|x) and a decoder part which computes P(x|z), sparse coding systems only parameterize the decoder: the encoder is implicitly defined as the solution of an optimization. A middle ground between ordinary auto-encoders and sparse coding was proposed in [12] applied to image data recognition and machine vision tasks. They propose to let the codes z be free (as in sparse coding algorithms), but include a parametric encoder (as in an ordinary auto-encoder) and a penalty for the difference between the free non- parametric codes z and the outputs of the parametric encoder. In this way, the optimized codes z tries to satisfy two objectives:

reconstruct well the input (like in sparse coding), while not being too far from the output of the encoder (which is stable by construction, because of the simple parameterization of the encoder)

Denoising auto-encoders. Another strategy is to add noise in the encoding. The denoising auto-encoder thus minimizes the error in reconstructing the input from a stochastically corrupted transformation of the input. Intuitively, a denoising auto-encoder does two things: try to encode the input (preserve the information about the input), and try to undo the effect of a corruption process stochastically applied to the input of the auto-encoder. [13] [14] [15]

2.6 Data Clustering

Data clustering is, nowadays, a well-known process of partitioning or grouping a given set of patterns into classes of similar objects. Clustering or cluster analysis, as it is called, is an unsupervised method. The main task of exploratory data mining and a technique for statistical data analysis. Furthermore, cluster analysis has many applications and it is used in many fields, including machine learning, pattern recognition, bioinformatics and data compression. The main goal of clustering analysis is to reveal the natural groupings of a set of patterns, points or objects. The objects within a group will be similar (or related) to one another and different (or unrelated) to objects in other groups. The greater the similarity within a class and the difference among classes, the better or more distinct the clustering result. A general definition of clustering can be stated as follows: Given a database of n objects, find K groups based on a measure similarity, such that the similarities between objects in the same group are high, while the similarities between objects in different groups are low.

K-means is one of the most widely used algorithms in machine learning for clustering. The K-means algorithm is implicitly based on pairwise Euclidean distances[4]. This method, describes the best possible partitioning of a data set containing k number of clusters. The method is defined by its objective function which aims to minimize the

[4] https://en.wikipedia.org/wiki/Euclidean_distance

sum of all squared distances within a cluster, for all clusters. The data are represented as linear combinations of a small number of cluster centroid vectors where linear combination weights.

The need to analyse weather data led the scientists to firstly try to apply variation of cluster algorithms taking account the spatial and temporal aspect of the data. The developed methods aim to reveal complex weather patterns and relationships in order to aid meteorological applications [16] [17]. Mining this data can produce new insights into weather and climatological trends that can aid in applications such as operations planning. Data mining techniques may also be used to reveal anomalous climatic regions that may be used to modify large scale weather models to include locally relevant phenomena.

2.7 Auto-encoders for deep clustering

Given a large collection of unlabelled data represented by raw pixels the question arises how to divide them into K groups in terms of inherent latent semantics. The traditional way is first extracting feature vectors according to domain-specific knowledge and then employing clustering algorithm on the extracted features [18]. By applying deep learning approaches, the researchers aim to combine feature learning and clustering into a unified framework which can directly cluster raw data with even higher performance. We refer to this new category of clustering algorithms as Deep Clustering [19].

One branch of popular methods for clustering is k-means [20] and Gaussian Mixture Models (GMM) [21]. These methods are fast and applicable to a wide range of problems. However, their distance metrics are limited to the original data space and they tend to be ineffective when input dimensionality is high [22].

Several variants of k-means have been proposed to address issues with higher-dimensional input spaces [23] [24] [25]. De la Torre & Kanade (2006) [26]; Ye et al. (2008) [27] to perform joint dimensionality reduction and clustering by first clustering the data with k-means and then projecting the data into a lower dimension where the inter-cluster variance is maximized. Thus, these frameworks are limited to linear embedding. For this reason, methods employing deep neural networks have been

studied and developed to perform non-linear embedding to transform the data into more clustering-friendly representation [28].

Generally clustering is central to many data-driven applications domains and has been studied extensively in terms of distance functions and grouping algorithms. Existing deep clustering algorithms broadly fall into two categories:

- Applying clustering after having learned a representation.
- Approaches that jointly optimize the feature learning and clustering.

Moreover, in recent years Generative Adversarial Networks models became popular. They can not only perform the clustering task but also generate new samples from the resulting clusters such as Information Maximizing Generative Adversarial Network InfoGAN [29] and clusterGAN [30]

In the first category of solution use application dependent dimensionality reduction algorithms for feature selection. For example, Deep Embedding Clustering DEC [31]. DEC acted as reference of many researchers [32] [33] [34]. The auto-encoder is trained by using the reconstruction loss and then drops the decoder part. The features extracted by the encoder network serve as the input of clustering module. After that, the network is fine-tuned using the cluster assignment loss. Meanwhile, the clusters are iteratively refined by minimizing the KL-divergence[5] between the distribution of soft labels and the auxiliary target distribution. As a result, the algorithm obtains a good result and become a reference to compare the performances of new deep clustering algorithms. However, they ignore the preservation of data properties, which may lead to the corruption of feature space [35]. In addition, Discriminatively Boosted Clustering**,** DBC [36] has almost the same architecture with DEC and the only improvement is that it utilizes fully convolutional auto-encoder. It is worth mentioning that the aforementioned unsupervised methods use only the clustering loss to train the network.

[5] https://en.wikipedia.org/wiki/Kullback–Leibler_divergence

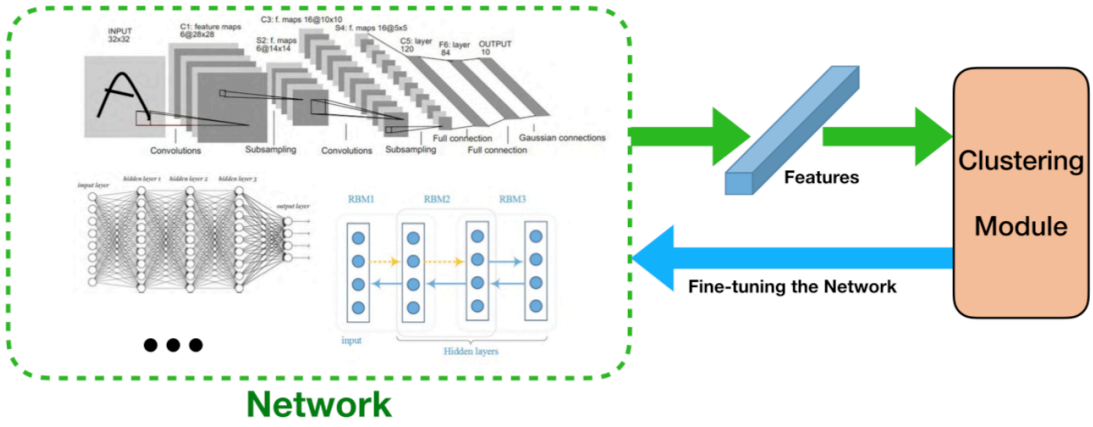

Figure 11 General Architecture for applying clustering after having learned a representation [37]

The latter category of algorithms explicitly introduces a joint learning framework to minimize the unified clustering and reconstruction loss functions together and train all network layers simultaneously. Deep Clustering Network DCN [38] is one of the most remarkable methods in this field, which combines auto-encoder with the k-means algorithm. Unlike DEC, it jointly optimizes the reconstruction loss and k-means loss. Since k-means uses discrete cluster assignments, the method requires an alternative optimization algorithm. The objective of DCN is simple compared with other methods and the computational complexity is relatively low. On the other hand, Deep Embedding Network DEN [39] proposes a deep embedding network to extract effective representations for clustering. It first utilizes a deep auto-encoder to learn reduced representation from the raw data. Secondly, in order to make the learned representations suitable for clustering, a locality- preserving constraint is applied. Furthermore, it also incorporates a group sparsity constraint to diagonalize the affinity of representations. Together with the reconstruction loss, the aforementioned losses are jointly optimized to fine-tune the network for a clustering-oriented representation. The locality-preserving and group sparsity constraints serve as the auxiliary clustering loss thus, as a last step, k-means is required to cluster the learned representations.

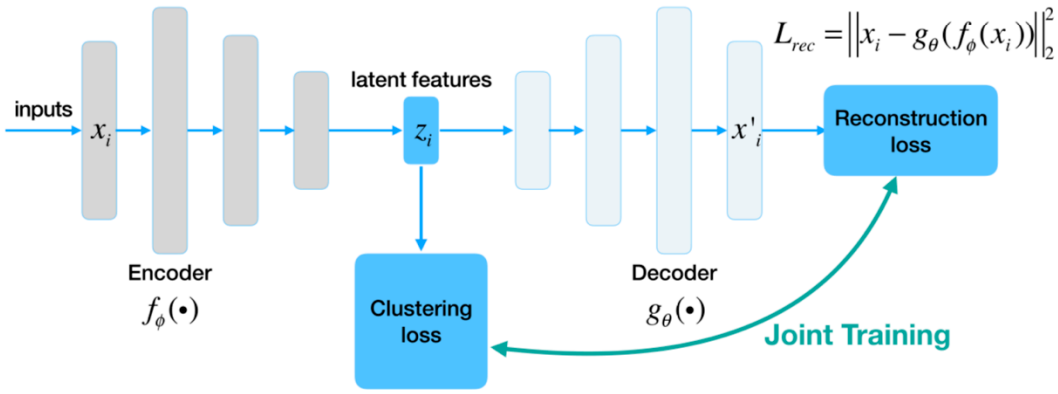

Figure 12 Joint Reconstruction and Clustering [37]

In contradiction with the previous approaches the authors in [40] are motivated and develop a simple yet effective model based on auto-encoders to learn a clustering friendly embedded space that separates the natural clusters in the data well. They propose the Representation Network. This network simultaneously learns both the mapping from latent representation space of the auto-encoder to an embedding space by minimizing cross-entropy between two probability distributions that denote pairwise similarity of points in auto-encoder latent space and the embedding space respectively.

Most of the previous work focusing only on spatial feature learning and clustering. A key reason for their success is that they can be applied on any specific domain or dataset, like satellite, medical images, or on images captured with a new modality, like depth, where annotations are not always available in quantity.

On the other hand, there are also a lot of existing research in retrieving important patterns from temporal data such as time series. This method is also known temporal clustering which is a challenging problem in machine learning. Notable methods are Brockwell and Davis, 2013[41]; Keogh et al., 2001[42] but they lack a general methodology for the selection of an effective latent space that captures the properties of time series data. In [43] the authors propose the Deep Temporal Clustering DTC and achieve effectively latent representation learning by proposing a jointly training algorithm architecture for clustering and auto-encoding aiming at optimizing the

clustering task. Moreover, clustering temporal data can be combined with segmentation providing better results compared to cluster algorithms [44] for learning human gesture segmentation. This can be achieved by simultaneously discover suitable deep representations and temporal boundaries as well. The cluster process is providing supervisory cues for updating temporal boundaries.

Due to the time-series nature, the encodings could also be used for prediction [45] [46] in network traffic and in extreme time series events [47].

While Unsupervised and Supervised Representation Learning is a well-studied topic in the broad field of machine learning, as previously described, not much work has been done towards the analysis of with cross-domain features such as spatial and temporal.

Spatio-temporal problems have been studied in broad domains [48] [49] [50], such as object detection, emotion recognition and in transportation systems, power grid networks and weather forecasting, where data are collected in a geographical area over time. Most of those methods take advantage of the temporality in order to extract important patterns and utilize them for prediction [51].

A notable work using spatio-temporal data and specially in weather data is [1]. The authors take advantage of the spatial part of the data, in order to extract dominant weather features to improve weather clustering and develop emergency response applications for nuclear or radiological events.

To sum up while clustering techniques have been successfully applied in a plethora of topics, their extension to real-data weather data using time series data with spatial features remains an open problem. This has left a gap in technology for accurate unsupervised learning of time series data which encompasses many areas of science and engineering such as medicine and wind analysis. The problem of unsupervised spatio-temporal clustering is particularly challenging and exhibits considerable variations in important properties and features, temporal scales, and dimensionality.

3 Architecture

This chapter describes the approach to develop a representation learning framework. The framework extracts useful information from the spatio-temporal real data and perform deep feature clustering. Since Temporal data are sequences, the proposed approach should consider the temporal dependencies intrinsic to time series data. Moreover, learning good representations of data allows to understand the data in meaningful ways and makes it possible to execute further machine learning tasks. By leveraging from the temporal aspect of the data in contradiction with the [1] we expect the clusters to be contiguous in space and also have similar temporal characteristics. The main motivation behind the clustering objective is that points are grouped together not only based on their spatial proximity but also their temporal similarity. Dealing with raw spatial-temporal datasets is too large for any algorithm to process; the goal our strategy is to reduce the size of that data by producing a smaller representation of the dataset, as opposed to compressing the data and then uncompressing it later for reuse. Furthermore, we aim to reduce and transform the data so that it can be managed and mined interactively.

The objective of the first stage is to reduce the size of the initial data without losing any relevant information. On the other hand, the purpose of the second stage is to apply mining techniques such as clustering, association rules on the tightly grouped data objects to produce new knowledge and ready for evaluation and interpretation. Clustering is a high-performance tool for detecting patterns in spatio-temporal data analysis [52]. Some of the benefits of using clustering techniques to analyse such datasets include a) the visualisation of clusters can help with understanding the structure of spatio-temporal datasets, b) the use of simplistic similarity measures to overcome the complexity of the datasets including the number of attributes, and c) the use of cluster representatives to help filter (reduce) datasets without losing important/interesting information. In our case the ground truth is missing therefore the contribution of a partition is actually unknown in general. Fortunately, existing clustering validation criteria measure the clustering quality of a partition from different perspectives, e.g., the validation of intra and interclass variation of clusters

such as Calinski-Harabasz Index [53] and Silhouette Coefficient [54]. To a great extent, we can employ clustering validation criteria to estimate contributions of partitions in terms of clustering quality. Our work primarily focuses on obtaining meaningful representations of weather data in order to describe weather patterns, in more detail cluster descriptors representatives in our case. This will help us to eventually develop more accurate emergency-response applications aiming at estimating pollutant releases, as well as other types of applications.

The dimensionality of the latent space z, d_z has a major impact on the learning process. Therefore, it is important to discuss the effect of this hyper-parameter. On one hand, choosing a very small d_z would lead to under-fitting to training data and, in that case, the model might not be able to reconstruct well enough the normal pattern of the sequences. On the other hand, choosing a too large d_z could cause over-fitting to the training data and lead to poor generalisation. Moreover, with a code of larger size, the model could start learning to reconstruct even anomalous sequences and, thus, the performance of the objective would be reduced.

Therefore, the choice of the dimensionality of the latent space, d_z, is just another instance of the bias-variance trade-off. In a fully unsupervised scenario, it is difficult to choose this parameter. According to the literature this choice is performed empirically.

In the following subchapters presents the proposed approach, which consists of two fundamental stages: representation learning and clustering. The proposed architectures for representation learning are separated in 2 parts:

- The temporal Encoder-Decoder Model
- The Spatio-temporal Encoder-Decoder Model

3.1 The temporal Encoder-Decoder Model

LSTM units have been proposed and used as auto-encoder in as described in the literature and specially in [43] [55] [47]. They take advantage only of the temporal information of the data. These units have been utilized to create a network

architecture, the LSTM auto-encoder. In this subsection we will break the proposed LSTM auto-encoder network to understand it layer by layer.

Consider n high dimension unlabelled instances $x_1 x_2 \ldots x_n$ of temporal sequence X The goal of the auto-encoder is to perform temporal unsupervised dimension reduction of these n unlabelled sequences.

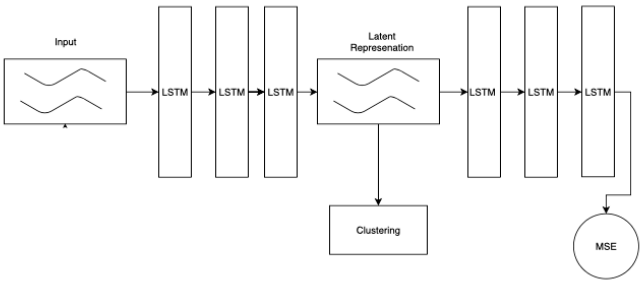

Figure 13 FC-LSTM Auto-encoder

Effective latent representation is a key aspect in clustering. The dimensionality reduction is crucial for further processing to avoid very long sequences which can lead to poor performance. The architecture consists from 2 parts. The encoder and the Decoder. The encoder accepts a sequence of reshaped frames in chronological order as input. The input consists from fixed vectors with shape <Samples, Timesteps, Data Features>. Each vector is a subsequence of the input data, where samples is the number of data in each subsequence, timesteps is equivalent to the length data sequence and data features represents the number of data dimensionality. On the other hand, the decoder tries to reconstruct the input sequences. Finally, the LSTM latent representation is assigned to the clustering layer, which will be explained in later section. This module groups the sequences $x_i \ldots x_n \;\; n = 1 \ldots n$ clusters.

The cost function is provided by the mean square error (MSE). The activation for all layers in rectifying linear units (RELU) except the last layer of the encoder, the input and the output, where Linear has be chosen. This should enable the network to

achieve a wide range of neuron values in the output and bottleneck layer. The middle layer or bottleneck is what forces the network to learn high level representations of the input data. Hence, the larger the middle layer, the more information can be captured during the process, so it is a feature highly depends on the data. Although this method can also be used to solve our spatiotemporal sequence forecasting problem, the fully connected LSTM (FC-LSTM) layer adopted by this model does not take spatial correlation into consideration.

3.2 The Spatio-temporal Encoder-Decoder Model

In order to model well the spatiotemporal relationships, we extend the idea of FC-LSTM to Conv-LSTM which has convolutional structures in both the input-to-state and state-to-state transitions. Convolutional LSTM (Conv-LSTM) units have recently been proposed and used successfully by [51]. It takes advantage of the spatial information retained by training convolutional weights to better propagate spatial features temporally in the LSTM. These units have been utilized to create two distinct network architectures, a Composite Conv-LSTM Encoder Decoder. To be more precise, Conv-LSTM architecture is used to retain the spatial structure of the multi-dimensional input data

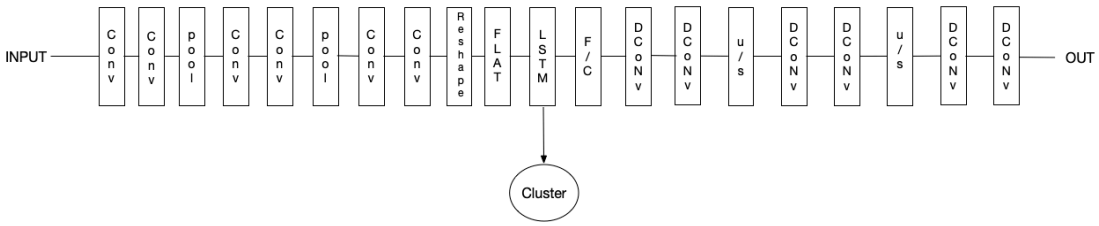

Figure 14 CNN - LSTM

The Encoder of the model is consisting of a stack of convolutional neural network. A sequence of weather snapshot is used as input to the first convolution layer.

Hence, cast the data sequence into a more compact representation while retaining most of the relevant spatial information. There are also max-pooling layers between the convolution layers to shrink the width and height of the feature map. Max-pooling helps the encoder to provide spatial invariance, that is, if we want to learn any

changes in weather between the sequences. The output of stacked convolution layer is feed then in LSTM layer to obtain the latent representation.

The LSTM consists of Input gate, Output gate, Forget gate and New Memory cell. The Input gate, Output gate, forget gate and New Memory cell/Current State is generated from the raw input, precious hidden state, thought vector from the convolution layers. The information flows unchanged into the network to preserve sequential correlations and to learn temporal dynamics. The encoder architecture allows to collapse the input sequences in all dimensions, and to cast the input into a much smaller latent space. The output of the LSTM layer is passed through the decoder to draw the final output.

In contradiction, the decoder consists from a Fully Connected layer which aggregated the encoded information and de-convolution layers. The new encoding at each time-step is decoded through de-convolution layers. Each convolution is followed by up-sampling layers to restore the image to its former size.

Learning in both CNN and LSTM is driven by interleaved minimization of a cost function. The cost function is provided by the mean square error (MSE) of the input sequence reconstruction from the LSTM latent representation; this ensures that the sequence is still well represented after the dimensionality reduction from the convolution layers We chose to use a depth of 3 convolution and pooling layers, since it leads to the best (average) validation accuracy. The depth is likely to be related to the sequence length though, and longer sequences may benefit from being processed by a deeper network, if one wanted to change the input length.

The activation for all layers in rectifying linear units (RELU) except the bottleneck, where Linear has be chosen. The loss function is described as:

$$MSE = \frac{1}{m}\sum_{i}(\hat{y}^{(i)} - y^{(i)})^2$$

The equation above shows the calculation of the MSE for a reconstruction step $y^{\wedge(i)}$ and the actual value in at that time step

3.2 Clustering Module

The encodings from the proposed architectures are used as input. The temporal clustering layer consists of k centroids $w_{i,j} \in 1...k$. The cluster centroids are initialized using the latent signals z_i obtained by feeding input sequence x_i through the Auto Encoder. The extracted z_i are then used to perform clustering in the feature space Z through a similarity metric. Our objective here is to mine the most dominant weather patterns, which will help us later calculate the plume dispersions with finer geospatial resolution. Since clusters are derived from weather data, each cluster consist from weather snapshots and describes a specific weather pattern. The clustering method used in this task is K-means. By default, K-means uses centroid as descriptors. Even though this is sensible for fitting additional weather snapshots to the model, it is not useful for producing synthetic atmospheric dispersions as it does not contain any comprehensive temporal and spatial information.

To find the most representative weather snapshot among a sequence in a cluster we examined 2 criterion approaches:

1. Using pairwise distance as metric to find the Top 1 nearest point to cluster centroid.
2. The second approach is based on capturing a more concrete temporal representation of each cluster. Therefore, instead of take into account only the Top 1 closest point to the centroid we choose as descriptor to be the result of the top N points nearest to centroid in our case we used 10.

This approach should be able to capture all weather trends within each cluster effectively. After the weather patterns have extracted we calculate the atmospheric dispersions as will be described in later sections.

4 Methodology and Experiments

This chapter describes the selection and implementation of a suitable deep learning pipeline for deep spatio-temporal clustering using weather data. In the first subsection includes extensive information about the dataset, the challenges to pre-processing with the scope of utilizing the data for the training of the proposed deep learning architectures. the proposed a new architecture leverages from auto-encoders for feature extraction, achieving superior performance compared to the baseline.

4.1 Dataset Description

The dataset used to evaluate this framework consists of weather gridded data that originates from optimal combinations of Numerical Weather Prediction (NWP) data and observations. The dataset consists of 40 years observations covering spatial (less than 1-) and temporal (up to 6hr) features over the region of interest, i.e. the European continent[6].

For this study we used ERA-Interim data covering 40 years (1986-1993 and 1996-2018) with a 6hr temporal resolution provisioned by NCAR in the GRIB format. The collected data contains 37 standard pressure levels in ERA-Interim in mb (or hPa) are, from highest isobaric altitude to lowest isobaric altitude in the atmosphere. In this study we focus on the geopotential height (GHT) variable, a "gravity-adjusted" height, and more specifically to either GHT700hPa or a combination of GHT@500, 700 and 900hPa. According to the literature, GHT has been shown to be predictive of weather circulation patterns [56], [57], [58], [59]. The weather dataset firstly, is pre-processed via the Weather Research and Forecasting (WRF) [60] pre-processor WPS to define meteorological data on a Cartesian domain that covers Europe, with a spatial resolution of 64×64 cells of 75×75 in the west-east and south-north directions. In the vertical direction, we retained the original pressure levels. Generally, WRF is an atmospheric dispersion model which can produce simulations based on actual atmospheric conditions (i.e., from observations and analyses) or idealized conditions. WRF offers operational forecasting a flexible and computationally-efficient platform,

[6] http://www.ecmwf.int/en/research/climate-reanalysis/era-interim.

while reflecting recent advances in physics, numeric, and data assimilation (chemical reactions, radioactive decay, etc.).

The resulting data used to derive weather patterns are either 2D grids of 64×64 cells when using 1 GHT pressure, or 3D grids of $64 \times 64 \times 3$ cells, combining all pressures. An example of GHT at the 700hPA pressure level is shown below. The weather patterns discovered as a result of feature extraction and analysis are "downscaled" temporally using WRF, resulting to hourly data of the same spatial dimensionality as before.

The dataset can be found online the European Centre for Medium range Weather Forecast (ECMWF)[7] .

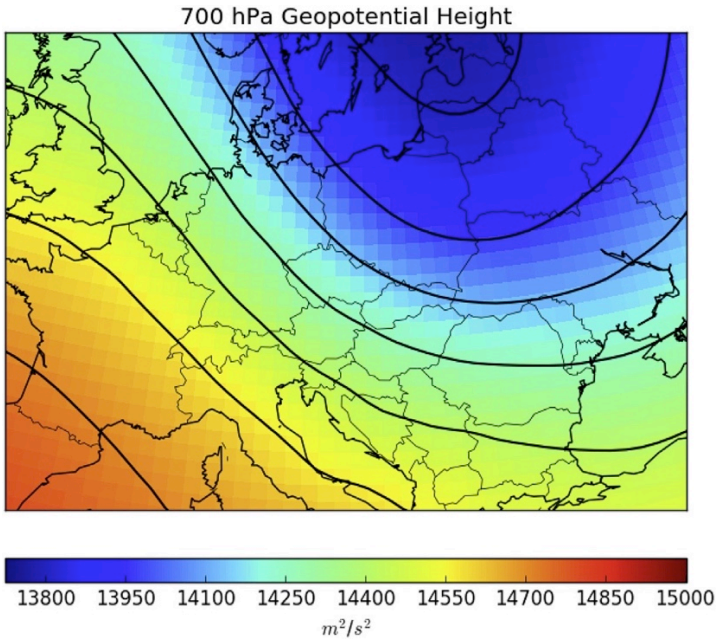

Figure 15 GHT@700

In our case the weather reading are taken from sensors across Europe every 6 hours. We propose a framework for data compression. From the machine learning perspective this problem can be regarded as a spatiotemporal sequence dimension reduction problem. Suppose we observe a dynamical system over a spatial region

[7] https://www.ecmwf.int/en/forecasts/datasets/reanalysis-datasets/era-interim

represented by an $M \times N$ grid which consists of M rows and N columns. Inside each cell in the grid, there are P measurements which vary over time. Thus, the observation at any time can be represented by a tensor $X \in \Re \quad P \times M \times N$ where $\Re$ denotes the domain of the observed features. If we record the observations periodically, we will get a sequence of tensors $x_1 ... x_n$. The goal of the spatio-temporal sequence dimension reduction is to achieve an efficient feature representation in a lower dimension space.

4.2 Experimental Setup

The auto encoders were trained using 38 years of 6-hourly weather snapshots Weather data used for training and evaluation were represented by a grid 64×64. Each sample is represented as a 2-day snapshot; longer windows affect the temporal resolution in each sample. In order to obtain more features to train we used a sliding window method generates $x_t \in R_{s \times w \times f}$ where w is time window and $t \in [0, \bar{t}]$, $\bar{t}$ represents the number of time steps w, s number of samples and f the features. Each sliding window has a size of 2-day (4 x 6 hours measure every day, resulting to 8 samples for 2-days) overlapping 1 day. The utilization of overlapping windows will improve the performance of the proposed Neural Networks to learn the temporal dependencies between the samples. The implementation of time windows is presented graphically below.

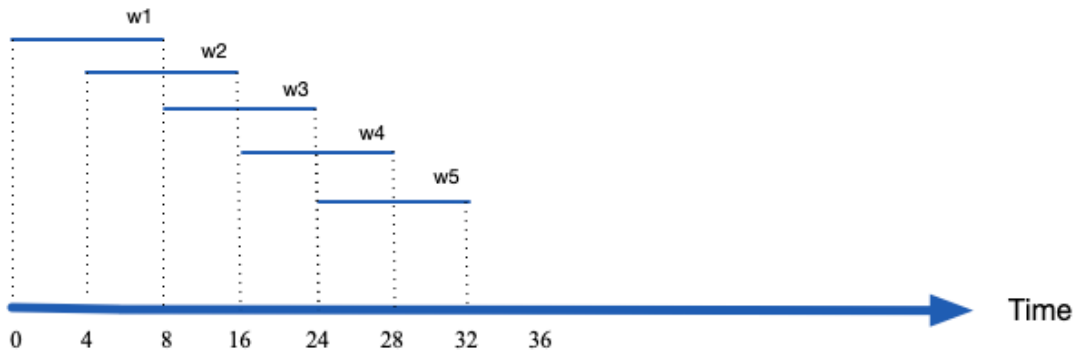

Figure 16 Sliding Window over Time

The clustering module is evaluated was evaluated using 2-day dispersions spanning 2 years of European weather (1994-1995), for a choice of 20 nuclear facilities. Besides

that, based on cluster scores and domain expert's opinions regarding observed weather patterns in Europe we set as number of clusters $k = 15$.

Furthermore, as a preprocessing step before applying our method the range of the independent variables is rescaled in [0,1] to eliminate any bias toward features who have a broad range of values.

4.3 Inverse modelling estimation application

The methodology for modeling the inverse source estimation problem comprises in the following steps:

1 Estimation of the weather patterns using the proposed framework from section 3
2 Computation of the atmospheric dispersions
3 Release source origin estimation

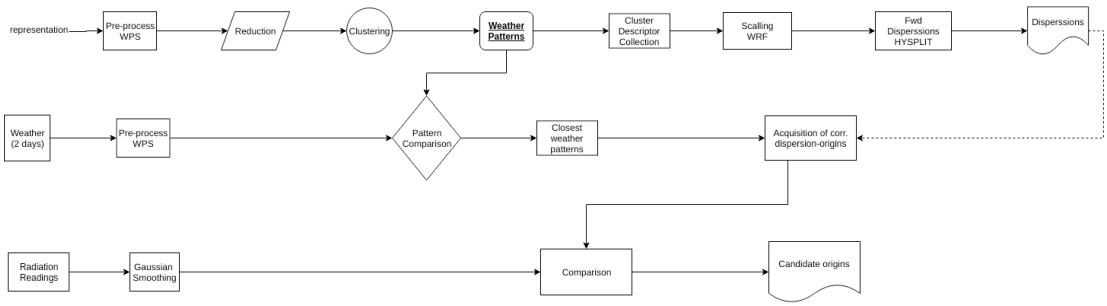

Figure 17 Methodology of inverse modelling estimation model

Dispersion model calculations are based on meteorological data, such as these processed by WRF, described in previous section.

Dispersion models can operate proactively when the emissions locations and the pollutant fluxes are available. In addition, they can be used to estimate unknown emissions sources and rates. In our dataset where long range spatial range and high meteorological variability is available, we use the so-called "Langarian" models can produce a more realistic depiction of the link between the concentrations at the receptor and the sources influencing it. To cope with this, we exploit the NOAA HYSPLIT atmospheric dispersion model [61], which works well with the WRF. HYSPLIT, supports back-trajectory analysis to determine the origin of air masses and establish source–receptor relationships. Air concentration of pollutants and deposition on the ground are computed on a grid defined independently of the meteorological data grid.

Following this, to estimate the release origin and complete the inverse estimation we propose the following procedure below. When an event has occurred via the produced clusters we obtain the most representative weather pattern that best represents the current weather. We treat this matching as an additional k-means assignment step, we choose the cluster which minimises the Euclidean distance between the observed weather and the centroid of the the extracted cluster from our proposed deep clustering model in question.

After choosing the most representative weather pattern we then consider the dispersions previously calculated for our set of fixed candidate release origins. To detect the probability, the release source location origin can we compare the release distributions of the cluster-based dispersions against hypothetical detection readings.

A hypothetical event is detected by a sequence of readings at certain locations. We model these point readings as a discrete probability distribution across the geographical area of study $r_{x,y} \in R$ (after dividing each reading by their total sum). This results in a reading at all locations where radiation has been detected and to 0 at all other locations. Then, the readings distribution is compared against a number of dispersion distributions given a weather pattern and a point of origin source.

Since $r_{(x,y)}$ is typically sparse we pass it through an isotropic 2D Gaussian smoothing filter obtaining $R = r_{g(x,y)}$ Based on the assumption that cells neighbouring readings

are likely to also be contaminated, albeit with a decreasing probability, applying such a filter increases the number of cells with non-zero radiation values and therefore the likelihood to obtain meaningful positive matches between readings and dispersions.

Finally, we collect the inverse order of the dispersions corresponding to the weather pattern according to their distance from the detection distribution R. We compare the readings and dispersion vectors using the cosine distance metric due to their sparsity.

4.4 Parameter Setup

In the following table the settings of the Neural Networks are being presented. The models were trained using stochastic gradient descent (SGD) on batch size 256 of the training data for 300 epochs. The learning rate for the FC-LSTM auto-encoder was 0.001 and for the Conv-LSTM 0.01. The input for both network is <Samples,8, Features>, where 8 is the dimension of a 2-day snapshot, since each day has 4 samples. The number of features depends on the input. If we choose to use GHT700hPa (results 4096 features) or a combination of GHT@500, 700 and 900hPa (results 12288 features).

The bottleneck layer was set to 256 for both architectures, this results to an image of shape 16 x 16. The following table portraits the detailed setting of the NNs.

Fully Connected - LSTM	*Input: <samples, 8, Features >* *LSTM 1: 1024* *LSTM 2 :512* *Latent LSTM Layer: 256* *LSTM 3: 512* *LSTM3:1024*

Figure 18 Fully Connected LSTM

Convolution - LSTM	*Input: <samples, 8, Features >* *Conv 1: 64 (3 x 3)* *Conv 2: 64 (3 x 3)* *Max-Pool (2,2)* *Conv 3: 32 (3 x 3)* *Conv 4: 32 (3 x 3)* *Max-Pool (2,2)* *Conv 5: 16 (3 x 3)* *Conv 6: 16 (3 x 3)* *Reshape ()* *Flatten ()* *Latent LSTM Layer: 256* *Dense - Fully Connected: 256* *Deconv: 16 (3 x 3)* *Deconv: 16 (3 x 3)* *UpSampling(2,2)* *Deconv: 32 (3 x 3)* *Deconv: 32 (3 x 3)* *UpSampling(2,2)* *Deconv: 64 (3 x 3)* *Deconv: 64 (3 x 3)*

Figure 19 Convolution - LSTM

Architectures with more layers and hidden units per layer seem to reduce the training loss but did not necessarily result in an improvement during release source estimation. Instead, the training time for the same amount of epochs was increased due to more model parameters. These results may show that the amount of training data is not sufficient for generalization.

5 Results and Discussion

The networks were implemented and tested using Python, TensorFlow 1.3 [8]and Keras 2.0[9] software on Nvidia GTX 1080Ti graphics processor.

5.1 Evaluation in an inverse modelling estimation application

Based on the inverse model descripted extensively in section 4.3 we evaluate the performance of the algorithm Deep Weather Clustering (DWC) is evaluated using the dataset as discussed in section 4.1. The results from the proposed models are matched again k-means clustering in Raw data. The following tables portray the obtained auto encoders unsupervised performance among different cluster descriptors for 10 and 30 readings using GHT@700 and the combination of GHT@500, 700, 900. All models are using GHT@700 except CNN-LSTM3D, which accepts as input all pressures as channels. The auto-encoder seems to be able to reconstruct the input effectively. We present the best results:

[8] https://www.tensorflow.org/
[9] https://keras.io

Results	*1@10pts*		*10@10pts*		*1@30pts*		*10@30pts*	
	Top1	***Top3***	***Top1***	***Top3***	***Top1***	***Top3***	***Top1***	***Top3***
Raw K-means	26.606	53.445	32.974	59.063	36.332	67.235	46.630	75.943
FC – LSTM	27.989	55.334	33.202	60.011	37.010	68.168	48.102	76.568
CNN – LSTM	29.082	56.213	34.928	61.634	38.222	71.064	50.428	79.464
CNN – LSTM 3D	30.090	57.110	35.377	62.090	38.990	71.800	51.002	79.901

Figure 20 Results from evaluating the our proposed models for 10 and 30 simulated locations reporting accuracy as a function of the dimensionality reduction configuration and the choice of the proposed cluster descriptor.

We observe that the Top10 10@10pts and 10@30pts descriptor performs better in all cases in comparison with top 1, 1@10pts and 1@30pts. This can be attributed that the Top 10 descriptor effectively represents all weather snapshot of the cluster. Beside that the conv-lstm auto-encoder work better than the other methods leading to better clustering outcome. This observation is also supported by the fact that applying inclusion of convolutional layers allow a better overall result in terms of both error of reconstruction and separability of the feature space. Beside that, including convolutional layers helps filtering spatial relationships between the pixels of the image.

The results obtained from the Neural Networks are fairly close to each other. This fact supports the idea that the key challenge in unsupervised anomaly detection is to learn good (expressive) representations of the data. This is the reason why this Thesis is strongly focused on representation learning.

5.2 Cluster Analysis and Latent Representation Space

At the heart of an auto-encoder lies its latent space: a low-dimensional representation of the data that encodes its underlying factors of variation. Therefore, it is interesting to visualize these representations. For visualization purposes, the dimensionality of

the latent space was reduced from the bottleneck dimension 16 x 16 D (d_z = 128) to 2D using t-Distributed Stochastic Neighbour Embedding (t-SNE).

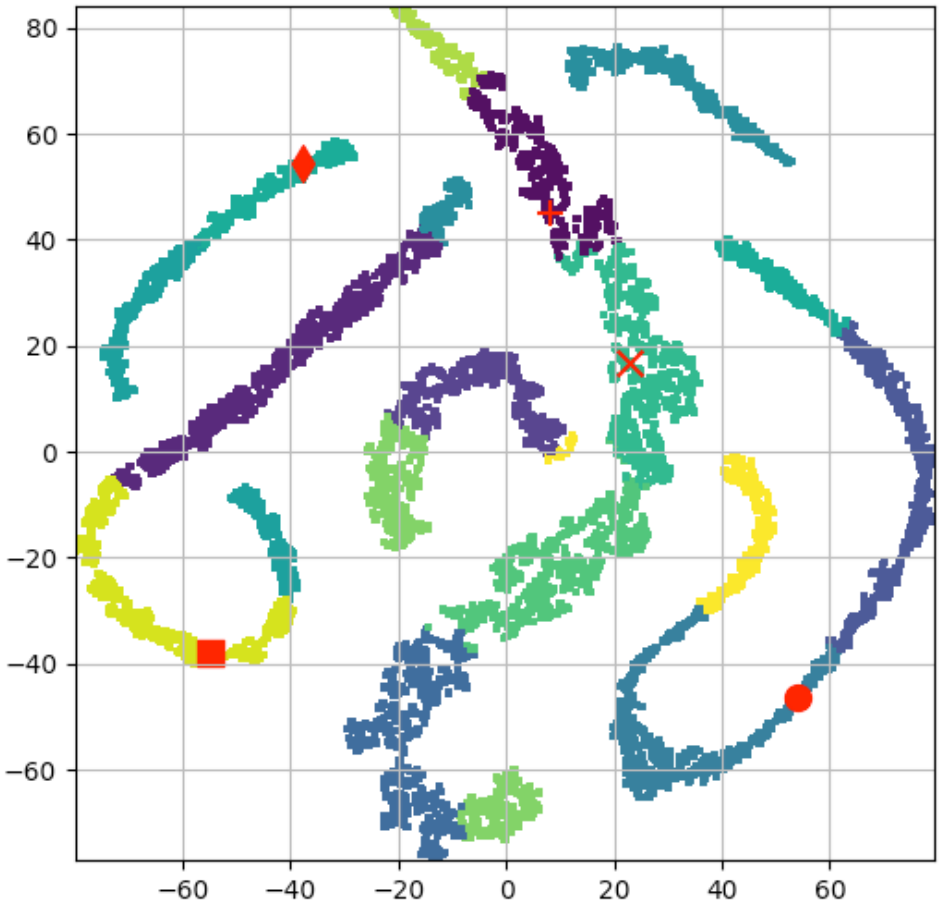

Figure 21 T-SNE Latent Space Visualization from the FC-LSTM, distances between a point of interest marked as (+) in Latent Space in different clusters. The t-SNE approach provides clustering visualization of the data points from the embeddings. Colors indicate the clusters from 1 to 15. The algorithms have learned the topogical structure of the data in the low-dimensional space. This can be proved by comparing the distances of between (x) point and square point. The distance of square from (+) point is far greater than the (x) point.

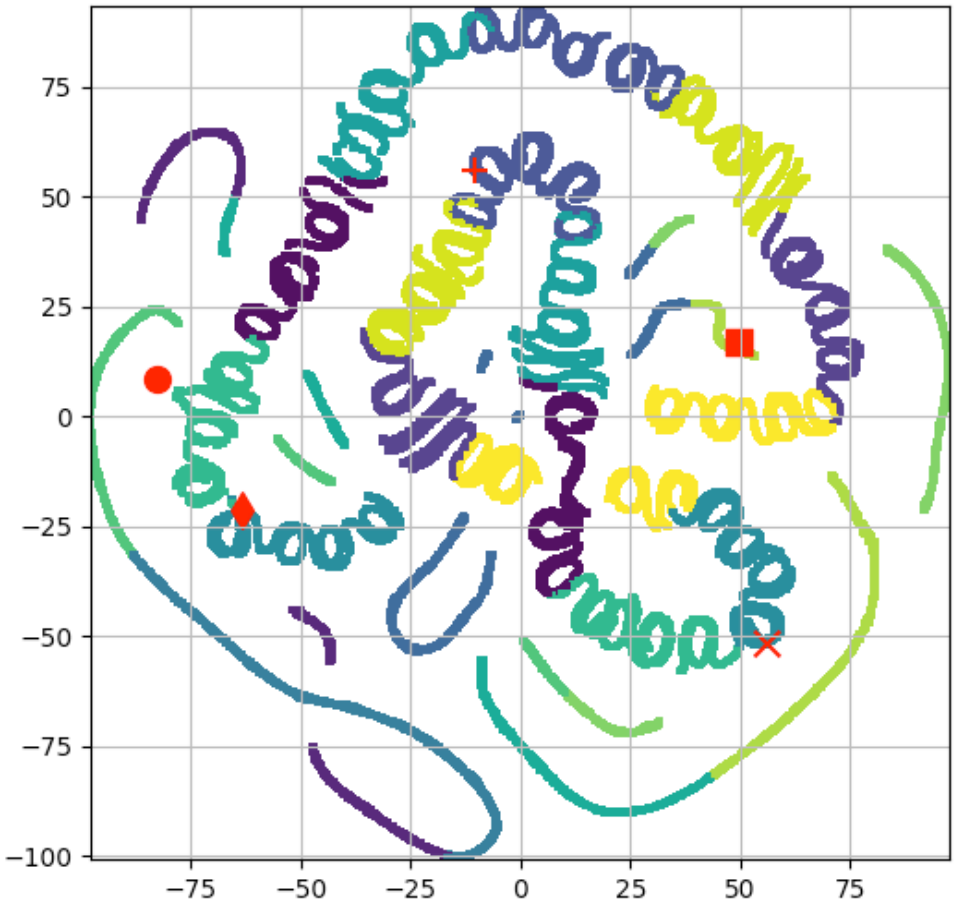

Figure 22 T-SNE Latent Space Visualization from the ConvLSTM, distances between a point of interest marked as (+) in Latent Space in diffent clusters. The t-SNE approach provides clustering visualization of the data points from the embeddings. Colours indicate the ground truth labels corresponding to the clusters from 1 to 15. The algorithms have learned the topogical structure of the data in the low-dimensional space. This can be proved by comparing the distances of between (x) point and square point. The distance of square from (+) point is far greater than the (x) point.

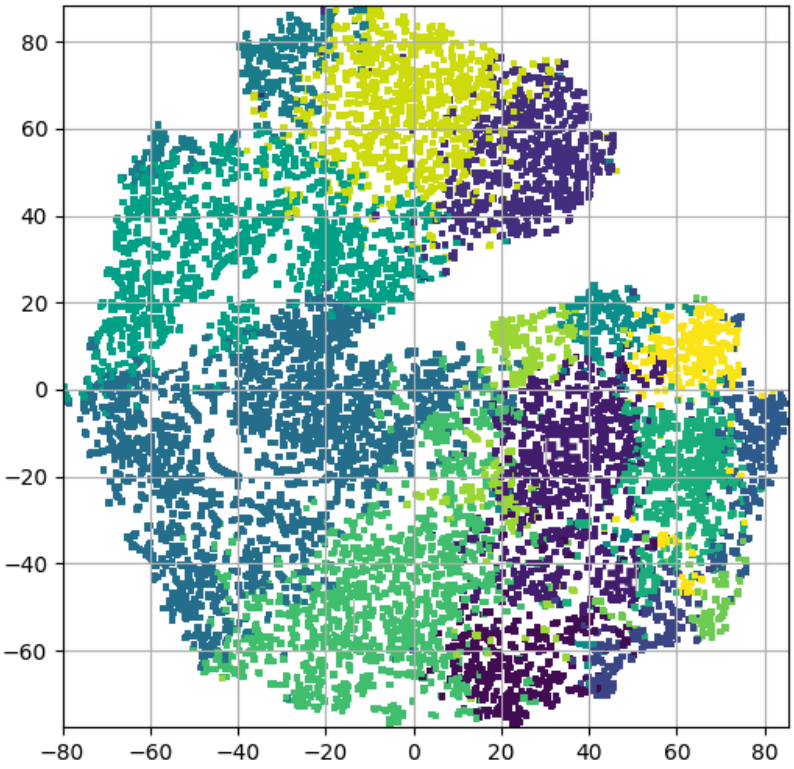

Figure 23 K-means latent representation. Each colour represents a cluster. The clusters are not well separable.

To demonstrate the efficiency of the feature reduction methods we attempt to uncover the manifold structure in the data. In the context of cluster quality and manifold learning we illustrate the distances between random points from the initial dimension space in the latent dimension space. We prove that the distances between the data points is similar in the latent space (Figure 20,21) and the clusters are well separated compared to k-means (Figure 22). This approach proves that the auto-encoder learned the internal model of the data. In contradiction working with more data we expect the clusters to be denser.

The visualization of the latent space shows evidence that the model is mapping sequences aligned in time onto the same region of the z-space and, more interestingly, it reveals similar weather matching period across the weather data.

In other words, the model has learned the seasonal property of the data without being told of it. It is important to recall that no prior information regarding the seasonality property is provided, all the windows were shuffled during training and in this experiment each window of observations has 8 time steps. A more detailed view of the content of each cluster we observe that each cluster is represented by certain years. To sum up, as observed, the latent space for the weather dataset is structured and expressive.

6 Conclusions and Future Work

In this thesis we have developed a framework to enhance the environmental research and provide a complete clustering framework. This model is using measurements including atmospheric pressure and could be extended by using various weather features. This work is inspired from times series and image processing applications, deep learning and data mining extending our knowledge in atmospheric data. In addition, this works represents an effort at applying unsupervised learning and explanatory analysis (data clustering) to weather data.

To cope we the challenging high data complexity, we applied deep auto-encoders to obtain latent features in order to aid the clustering objective to retrieve robust and meaningful weather patterns to use them for inverse source estimation models. Also we put forward a kind of temporal and spatial dependencies feature extraction methods, which select the most similar characteristics data based on the temporal and spatial correlation and clustering methods.

The proposed framework for deep weather clustering makes the quest for possible lines of future work. First, even though the proposed model was applied multivariate time series data using 3 GHT pressures, it is also suitable for inserting more metadata information such as wind information.

To reduce the training time, the FC-LSTM auto-encoder could be trained using *Teacher Forcing* [62]. Hence, during training the decoder was provided with the ground truth sequence from the previous time step. During inference, the decoder

output of the previous time step was fed back into the current decoding step. In theory, this speeds up the training in the beginning, where the model outputs more or less random values.

An extension of the current work is to design a semi-supervised setting that would allow to take possibly available labels into consideration. Moreover, in this scenario, it would be interesting to exploit a transfer learning approach that would allow to transfer knowledge between a source task where enough labels are available to a target task with fewer labels. This extension would allow to leverage the features learned across different datasets in other similar tasks.

Finally, applying end-to-end optimization of the network for reconstruction loss objective and clustering loss efficiently could lead to extract more representative spatio-temporal features that are better suited to separate the input sequences into categories (clusters). With the rise of unsupervised learning by deep generative we are able to enhance dimension reduction methods to capture high dimensional probability distributions in order to deal with high dimensional data. The latent space of GANs not only provides dimensionality reduction, but also gives rise to novel applications. Perturbations in the latent space could be used to determine adversarial examples that further help build robust classifiers/predictors [63]. Since GANs have outperformed auto-encoders in generating high fidelity samples, we had a strong intuition in favour of the powerful latent representations of GAN providing improved clustering performance as well [30] [64].

7 References

[1] Klampanos, Iraklis & Davvetas, Athanasios & Andronopoulos, Spyros & Pappas, Charalampos & Ikonomopoulos, Andreas & Karkaletsis, Vangelis. (2018). Autoencoder-Driven Weather Clustering for Source Estimation during Nuclear Events. Environmental Modelling and Software. 102 (April 2018). 84-93. 10.1016/j.envsoft.2018.01.014.

[2] Goodfellow, I. (2015). *Deep Learning Book*.

[3] Malinowski, A., Cholewo, T. J., & Zurada, J. M. (1995). Capabilities and limitations of feedforward neural networks with multilevel neurons. *Proceedings - IEEE International Symposium on Circuits and Systems*.

[4] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*. https://doi.org/10.1162/neco.1997.9.8.1735

[5] Gers, Felix & Schmidhuber, Jürgen & Cummins, Fred. (2000). Learning to Forget: Continual Prediction with LSTM. Neural computation. 12. 2451-71. 10.1162/089976600300015015.

[6] Springenberg, Jost & Dosovitskiy, Alexey & Brox, Thomas & Riedmiller, Martin. (2014). Striving for Simplicity: The All Convolutional Net.

[7] Lecun, Yann & Bottou, Leon & Bengio, Y. & Haffner, Patrick. (1998). Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE. 86. 2278 - 2324. 10.1109/5.726791.

[8] Bengio, Y. (2009). Learning Deep Architectures for AI. Foundations. 2. 1-55. 10.1561/2200000006.

[9] Hinton, G.E. & Salakhutdinov, R.R.. (2006). Reducing the Dimensionality of Data with Neural Networks. Science (New York, N.Y.). 313. 504-7. 10.1126/science.1127647.

[10] Hervé Bourlard and Yves Kamp. “Auto-association by multilayer per- ceptrons and singular value decomposition”. In: Biological cybernetics 59.4-5 (1988), pp. 291–294.

[11] Nathalie Japkowicz, Stephen Jose Hanson, Mark Gluck, et al. “Nonlinear autoassociation is not equivalent to PCA”. In: Neural computation 12.3 (2000), pp. 531–545.

[12] Ranzato, Marc'Aurelio & Boureau, Y-Lan & Lecun, Yann. (2007). Sparse feature learning for deep belief networks. Advances in Neural Information Processing Systems. 20.

[13] Masci, Jonathan & Meier, Ueli & Ciresan, Dan & Schmidhuber, Jürgen. (2011). Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction. 52-59. 10.1007/978-3-642-21735-7_7.

[14] Rifai, Salah & Vincent, Pascal & Muller, Xavier & Glorot, Xavier & Bengio, Y.. (2011). Contractive Auto-Encoders: Explicit Invariance During Feature Extraction. Proceedings of the 28th International Conference on Machine Learning, ICML 2011.

[15] Vincent, Pascal & Larochelle, Hugo & Bengio, Y. & Manzagol, Pierre-Antoine. (2008). Extracting and composing robust features with denoising autoencoders. Proceedings of the 25th International Conference on Machine Learning. 1096-1103. 10.1145/1390156.1390294.

[16] Liu, Zhijian & George, Roy. (2005). Mining Weather Data Using Fuzzy Cluster Analysis. Fuzzy Modeling with Spatial Information for Geographic Problems. 105-119. 10.1007/3-540-26886-3_5.

[17] Sozer, Aziz & Yazici, Adnan & Oğuztüzün, Halit. (2015). Indexing Fuzzy Spatiotemporal Data for Efficient Querying: A Meteorological Application. Fuzzy Systems, IEEE Transactions on. 23. 1399-1413. 10.1109/TFUZZ.2014.2362121.

[18] J. A. Hartigan and M. A. Wong, ‘‘Algorithm AS 136: A k-means clustering algorithm,’’ J. Roy. Stat. Soc. C, Appl. Stat., vol. 28, no. 1, pp. 100–108,1979.

[19] Nutakki, Gopi & Abdollahi, Behnoush & Sun, Wenlong & Nasraoui, Olfa. (2019). An Introduction to Deep Clustering. 10.1007/978-3-319-97864-2_4.

[20] MacQueen, J. (1967). Some Methods for Classification and Analysis of MultiVariate Observations. Proc. 5th Berkeley Symp. Math. Stat. Probab. 1. 281-297.

[21]Christopher M. Bishop. 2006. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag, Berlin, Heidelberg.

[22] Steinbach, Michael & Ertöz, Levent & Kumar, Vipin. (2003). The Challenges of Clustering High Dimensional Data. Univ. Minnesota Supercomp. Inst. Res. Rep.. 213. 10.1007/978-3-662-08968-2_16.

[23] T. Kohonen, ‘‘The self-organizing map,’’ Neurocomputing, vol. 21,nos. 1–3, pp. 1–6, 1998.

[24] Ester, Martin & Kriegel, Hans-Peter & Sander, Joerg & Xu, Xiaowei. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. KDD. 96. 226-231.

[25] Arthur, David & Vassilvitskii, Sergei. (2007). K-Means++: The Advantages of Careful Seeding. Proc. of the Annu. ACM-SIAM Symp. on Discrete Algorithms. 8. 1027-1035. 10.1145/1283383.1283494.

[26] De la Torre, Fernando & Kanade, Takeo. (2006). Discriminative cluster analysis.. 241-248.

[27] Ye, Yunming & Huang, Joshua & Chen, Xiaojun & Zhou, Shuigeng & Williams, Graham & Xu, Xiaofei. (2006). Neighborhood Density Method for Selecting Initial Cluster Centers in K-Means Clustering. 189-198. 10.1007/11731139_23.

[28] Yang, Bo & Fu, Xiao & Sidiropoulos, N.D. & Hong, Mingyi. (2016). Towards K-means-friendly Spaces: Simultaneous Deep Learning and Clustering.

[29Springenberg, Jost. (2015). Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks.

[30] Mukherjee, Sudipto & Asnani, Himanshu & Lin, Eugene & Kannan, Sreeram. (2019). ClusterGAN: Latent Space Clustering in Generative Adversarial Networks. Proceedings of the AAAI Conference on Artificial Intelligence. 33. 4610-4617. 10.1609/aaai.v33i01.33014610.

[31] Xie, Junyuan & Girshick, Ross & Farhadi, Ali. (2015). Unsupervised Deep Embedding for Clustering Analysis.

[32] F. Li, H. Qiao, B. Zhang, and X. Xi. (2017). ‘‘Discriminatively boosted image clustering with fully convolutional auto-encoders.’’

[33] Yang, Jianwei & Parikh, Devi & Batra, Dhruv. (2016). Joint Unsupervised Learning of Deep Representations and Image Clusters. 5147-5156. 10.1109/CVPR.2016.556.

[34] Wang, Zhangyang & Chang, Shiyu & Zhou, Jiayu & Wang, Meng & Huang, Thomas. (2016). Learning A Task-Specific Deep Architecture For Clustering. 369-377. 10.1137/1.9781611974348.42.

[35] Guo, Xifeng & Liu, Xinwang & Zhu, En & Yin, Jianping. (2017). Deep Clustering with Convolutional Autoencoders. 373-382. 10.1007/978-3-319-70096-0_39.

[36] F. Li, H. Qiao, B. Zhang, and X. Xi. (2017). ‘‘Discriminatively boosted image clustering with fully convolutional auto-encoders.’’ [Online]. Available: https://arxiv.org/abs/1703.07980

[37] Min, Erxue & Guo, Xifeng & Liu, Qiang & Zhang, Gen & Cui, Jianjing & Long, Jun. (2018). A Survey of Clustering with Deep Learning: From the Perspective of Network Architecture. IEEE Access. PP. 1-1. 10.1109/ACCESS.2018.2855437.

[38] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong. (2016). ‘‘Towards K-means friendly spaces: Simultaneous deep learning and clustering.’’ [Online]. Available: https://arxiv.org/abs/1610.04794

[39] P. Huang, Y. Huang, W. Wang, and L. Wang, ‘‘Deep embedding network for clustering,’’ in Proc. 22nd Int. Conf. Pattern Recognit. (ICPR), Aug. 2014, pp. 1532–1537.

[40] Dahal, Paras. (2018). Learning Embedding Space for Clustering From Deep Representations. 3747-3755. 10.1109/BigData.2018.8622629.

[41] Brockwell, Peter & Davis, Richard. (2019). Times series : theory and methods / Peter J. Brockwell, Richard A. Davis. SERBIULA (sistema Librum 2.0).

[42] Keogh, Eamonn & Chakrabarti, Kaushik & Pazzani, Michael & Mehrotra, Sharad. (2002). Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. Knowledge and Information Systems. 3. 10.1007/PL00011669.

[43] Madiraju, Naveen & Sadat, Seid M & Fisher, Dimitry & Karimabadi, Homa. (2018). Deep Temporal Clustering: Fully Unsupervised Learning of Time-Domain Features.

[44] Tzirakis, Panagiotis & Nicolaou, Mihalis & Schuller, Bjorn & Zafeiriou, Stefanos. (2019). Time-series Clustering with Jointly Learning Deep Representations, Clusters and Temporal Boundaries. 1-5. 10.1109/FG.2019.8756618.

[45] Hua, Yuxiu & Zhifeng, Zhao & Li, Rongpeng & Chen, Xianfu & Liu, Zhiming & Zhang, Honggang. (2019). Deep Learning with Long Short-Term Memory for Time Series Prediction. IEEE Communications Magazine. PP. 1-6. 10.1109/MCOM.2019.1800155.

[46] Malhotra, Pankaj & Ramakrishnan, Anusha & Anand, Gaurangi & Vig, Lovekesh & Agarwal, Puneet & Shroff, Gautam. (2016). LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection.

[47] Laptev, N., Yosinski, J., Li, L.E., & Smyl, S. (2017). Time-series Extreme Event Forecasting with Neural Networks at Uber.

[48] Atluri, Gowtham & Karpatne, Anuj & Kumar, Vipin. (2017). Spatio-Temporal Data Mining: A Survey of Problems and Methods. ACM Computing Surveys. 51. 10.1145/3161602.

[49] Le, Trung-Nghia & Sugimoto, Akihiro. (2017). Video Salient Object Detection Using Spatiotemporal Deep Features. IEEE Transactions on Image Processing. PP. 10.1109/TIP.2018.2849860.

[50] Min, W., & Wynter, L. (2011). Real-time road traffic prediction with spatio-temporal correlations. *Transportation Research Part C: Emerging Technologies*. https://doi.org/10.1016/j.trc.2010.10.002

[51] Shi, X., Chen, Z., Wang, H., Yeung, D., Wong, W., & Woo, W. (2015). Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. *ArXiv, abs/1506.04214*.

[52] Han, Jiawei & Kamber, M. & Pei, J.. (2006). Data Mining: Concepts and Techniques. 585-631.

[53] Caliński, T., & Harabasz, J. (1974). “A dendrite method for cluster analysis”. Communications in Statistics-theory and Methods 3: 1-27. doi:10.1080/03610926.2011.560741.

[54] Peter J. Rousseeuw (1987). “Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis”. Computational and Applied Mathematics 20: 53–65. doi:10.1016/0377-0427(87)90125-7.

[55] Srivastava, Nitish & Mansimov, Elman & Salakhutdinov, Ruslan. (2015). Unsupervised Learning of Video Representations using LSTMs.

[56] Huth, R., 1996. An intercomparison of computer-assisted circulation classification methods. Int. J. Climatol. 16 (8), 893e922. https://doi.org/10.1002/(SICI)1097- 0088(199608)16

[57] Huth, Radan & Beck, Christoph & Philipp, Andreas & Demuzere, Matthias & Ustrnul, Zbigniew & Kučerová, Monika & Kyselý, Jan & Tveito, Ole. (2008). Classifications of Atmospheric Circulation Patterns.

[58] Li, Jiandong & Liao, Hong & Hu, Jianlin & Li, Nan. (2019). Severe particulate pollution days in China during 2013–2018 and the associated typical weather patterns in Beijing-Tianjin-Hebei and the Yangtze River Delta regions. Environmental Pollution. 248. 10.1016/j.envpol.2019.01.124.

[59] Sipek, Vaclav. (2013). The influence of large-scale climatic patterns on precipitation, temperature, and discharge in Czech river basins. Journal of Hydrology and Hydromechanics. 61. 278-285. 10.2478/johh-2013-0035.

[60] Michalakes, John & Dudhia, Jimy & Gill, D. & Henderson, Tom & Klemp, J. & Skamarock, W. & Wang, Wei. (2005). The Weather Research and Forecast Model: Software Architecture and Performance. 25. 156-168. 10.1142/9789812701831_0012.

[61] Stein, Ariel & Draxler, R.R. & Rolph, Glenn & Stunder, Barbara & Cohen, M.D. & Ngan, F.. (2016). NOAA’s HYSPLIT atmospheric transport and dispersion modeling system. Bulletin of the American Meteorological Society. 96. 150504130527006. 10.1175/BAMS-D-14-00110.1.

[62] Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., and Shroff, G. (2016). Lstm- based encoder-decoder for multi-sensor anomaly detection. arXiv preprint arXiv:1607.00148

[63] Ilyas, Andrew & Jalal, Ajil & Asteri, Eirini & Daskalakis, Constantinos & Dimakis, Alexandros. (2017). The Robust Manifold Defense: Adversarial Training using Generative Models.

[64] Santurkar, Shibani & Budden, David & Shavit, Nir. (2018). Generative Compression. 258-262. 10.1109/PCS.2018.8456298.