



January 2016

Novelty Detection And Cluster Analysis In Time Series Data Using Variational Autoencoder Feature Maps

Sophine Clachar

Follow this and additional works at: <https://commons.und.edu/theses>

Recommended Citation

Clachar, Sophine, "Novelty Detection And Cluster Analysis In Time Series Data Using Variational Autoencoder Feature Maps" (2016). *Theses and Dissertations*. 2004.
<https://commons.und.edu/theses/2004>

This Dissertation is brought to you for free and open access by the Theses, Dissertations, and Senior Projects at UND Scholarly Commons. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of UND Scholarly Commons. For more information, please contact zeinebyousif@library.und.edu.

NOVELTY DETECTION AND CLUSTER ANALYSIS IN TIME
SERIES DATA USING VARIATIONAL AUTOENCODER
FEATURE MAPS

by

Sophie A. Clachar

Bachelor of Science, University of Technology, Jamaica, 2007

Master of Science, University of North Dakota, 2011

A Dissertation

Submitted to the Graduate Faculty

of the

University of North Dakota

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

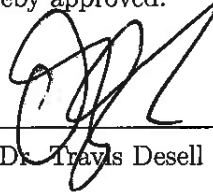
Grand Forks, North Dakota

December

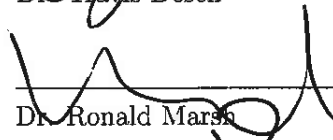
2016

© 2016 Sophie A. Clachar

This dissertation, submitted by Sophie A. Clachar in partial fulfillment of the requirements for the Degree of Doctor of Philosophy from the University of North Dakota, has been read by the Faculty Advisory Committee under whom the work has been done and is hereby approved.



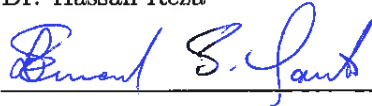
Dr. Travis Desell



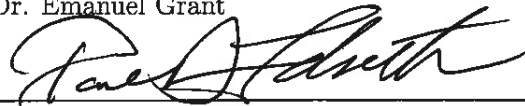
Dr. Ronald Marsh



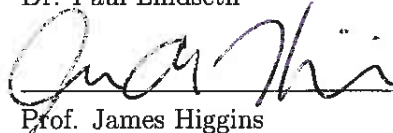
Dr. Hassan Reza



Dr. Emanuel Grant



Dr. Paul Lindseth



Prof. James Higgins

This dissertation is being submitted by the appointed advisory committee as having met all of the requirements of the School of Graduate Studies at the University of North Dakota and is hereby approved.



Dr. Grant McGimpsey
Dean of the School of Graduate Studies

December 5, 2016

Date

PERMISSION

Title Novelty Detection and Cluster Analysis in Time Series Data using
 Variational Autoencoder Feature Maps

Department Department of Computer Science

Degree Doctor of Philosophy

In presenting this dissertation in partial fulfillment of the requirements for a graduate degree from the University of North Dakota, I agree that the library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by the professor who supervised my dissertation work or, in his absence, by the Chairperson of the department or the dean of the School of Graduate Studies. It is understood that any copying or publication or other use of this dissertation or part thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of North Dakota in any scholarly use which may be made of any material in my dissertation.

Sophine A. Clachar
December 2016

CONTENTS

LIST OF FIGURES	viii
LIST OF TABLES	xiii
ACKNOWLEDGEMENTS	xv
ABSTRACT	xvi
CHAPTER	
I INTRODUCTION.	1
1 Scope & Objectives	4
2 Motivation & Contributions	5
II BACKGROUND	7
1 K-Means Clustering	7
2 Density Based Spatial Clustering of Applications with Noise (DBSCAN)	8
2.1 Parallel DBSCAN	9
3 Artificial Neural Networks (ANN)	10
3.1 Kohonen Self-Organizing Map (SOM)	11
3.1.1 Parallel SOM	12
3.2 Variational AutoEncoders (VAE)	13
4 Evaluating Distance/Similarity	16
4.1 Euclidean Distance	17
4.2 Mahalanobis Distance.	17
4.3 Dynamic Time Warping (DTW)	18
III RELATED RESEARCH	19
1 Clustering Time Series Data	19
1.1 Integrating Curve Similarity via Dynamic Time Warping.	19

2	Dimensionality Reduction using Machine Learning & Statistical Approaches	21
IV	THE DATA REPOSITORY	23
1	Overview & Architecture.	23
2	Data Preprocessing & Transformation	24
2.1	Phase of Flight Identification and Designation	25
V	RESEARCH METHODOLOGY.	29
VI	RESULTS & EVALUATION	32
1	Serial Clustering Results	32
1.1	K-Means	32
1.2	DBSCAN	33
1.3	SOM	34
2	Distributed Clustering Results	37
2.1	DBSCAN	37
2.2	SOM	40
2.3	Variational Autoencoder Feature Map (VAEFM)	41
2.3.1	Performance Evaluation	44
2.3.2	Evaluation of the Data Manifold	46
2.3.3	Cluster Evaluation	47
2.3.4	Summary	85
VII	CONCLUSION & FUTURE WORK	87
1	Future Work	88
APPENDIX		
A	PSEUDOCODE ALGORITHMS	89
1	K-Means	89
2	Density Based Spatial Clustering of Applications with Noise (DBSCAN)	90

3	Kohonen's Self Organizing Map (SOM)	91
4	Parallel SOM	92
5	Variational Autoencoder Feature Map	93
	BIBLIOGRAPHY	95

LIST OF FIGURES

Figure		Page
1	The DBSCAN algorithm (a) showing density-reachability and (b) density-connectivity of data points p and q, as described by Ester et. al. [1]	8
2	The mathematical representation of a neuron, which is the building block of a neural network. It has three inputs, and their respective weight vector; the weighted sum of inputs and the bias are passed into the activation function f , producing an output y	10
3	An 8X8 SOM with input layer $x_0 - x_5$. The input data traverses the entire network to identify the best matching unit (BMU). After the BMU is identified, its weights and those of its neighboring neurons $N_0 - N_7$, are updated during training.	12
4	The unfolding of the SOM's lattice during training as the neurons weight vectors are updated to model the data.	12
5	The basic autoencoder network which learn representations of its input data by minimizing the reconstruction error between the encoder and decoder layers in an unsupervised manner. The use of regularization, prevents overfitting and places constraints on the number of hidden neurons that are required to learn useful features. The bias has been excluded from the illustration.	14
6	Variational autoencoders further regularizes the basic autoencoder by learning the joint distribution over the input data and its latent representations. It calculates the weighted sum of its inputs and bias terms – similar to the basic autoencoder. Assuming this produces a gaussian distribution, a third layer is integrated which samples μ and σ from the previous layer. The network then integrates an auxiliary random variable ϵ to generate gaussian noise, $\epsilon \sim \mathcal{N}(0, I)$ and computes $\mu + \sigma\epsilon$. Training the VAE is the process of measuring how much information is lost and minimizing that using gradient descent.	15
7	The DTW alignment of two time-dependent sequences (the aligned points are depicted by arrows) [2]	18

8	Flight data is recorded using an airborne recording device which may encrypt the data. At the end of each flight the data is manually retrieved from the aircraft, decrypted and converted into a CSV format before it is uploaded to the NGAFID.	24
9	A geometric fence, indicated in red, that was created around the Grand Forks International Airport’s runway 35R.	26
10	A sample of flight trajectories, i.e. the approach configuration, which made their final approach on runway 35R.	27
11	The temporal flight data was obtained from the NGAFID and underwent preprocessing and transformation steps. The data was saved into a text file which was transferred to the high performance computing (HPC) center, where the various clustering algorithms were performed; after which their respective results were gathered. .	29
12	Graphical illustration of three clusters identified in 1500 flights when using the k-means algorithm. The choice of distance metric or scale of the data did not influence the selection of intuitive clusters; however it determined how the dataset was partitioned. . .	33
13	Graphical illustration of the one cluster and various outliers found using DBSCAN. The choice of distance metric influenced the cluster formations and the number of outliers that were detected.	34
14	The U-Matrix of a trained SOM, which is color coded; red indicates regions of large dissimilarity and green shows cohesion between neurons.	36
15	The computation time of the DBSCAN algorithm, as the number of processors increased, and its serial counterpart. The performance was assessed by executing the algorithm multiple times. The average runtime, i.e. the solid line, was calculated and the shaded regions depicts variations in the computation time, which occurs due to bottlenecks at the master processor. On average, the slowest computation time was 2060 minutes when using 2 processors and the fastest was 1507 using 16 processors.	38
16	The computation time of the asynchronous SOM algorithm, as the number of processors increased, and its serial counterpart. Multiple runs were also performed for the asynchronous algorithm, however variations in the computation time was miniscule.	41

17 The computation time of the serial and asynchronous SOM algorithm compared with DBSCAN’s serial and asynchronous algorithms. The serial SOM performed slower than both serial and async DBSCAN because the algorithm endures more computation, when the data is compared to each node in the neural network. However, the async SOM algorithm outperformed its serial counterpart, as well as both the serial and async DBSCAN algorithms; its fastest computation time was 239 minutes when using 16 processors, as opposed to DBSCAN’s 1507 minutes. 42

18 Graphical illustration of the VAE (a) which accepts 9 flight parameters, and (b) depicts the block diagram of (a) which is stacked sequentially to model each second in the temporal data. 44

19 The computation time of the VAEFM, Serial SOM, Async SOM, Serial DBSCAN and Async DBSCAN algorithms as the number of processors increased. VAEFM demonstrated an average speedup of 70%, and its scalability was demonstrated up to 256 processors – where it analyzed over 2500 flights in under 5 minutes. 45

20 The manifold of the approach configurations which are color coded based on their neuron mappings. 46

21 The color map for the neurons which shows the number of flights that were mapped to each. 46

22 Cluster 1 - Aerial view of the trajectories for the excessively high and fast approaches; 10% of these were aborted. 49

23 Cluster 1 - Line graph showing the changes in altitude, indicated airspeed, vertical speed, position (left/right of centerline), roll and pitch. This cluster contained flights with excessively high altitudes, airspeeds and vertical speeds while on final approach. Approximately 10% of these flights performed a go-around maneuver. . . . 50

24 Cluster 1 - Pie Chart showing the validation metrics and their respective percentages of occurrence. All 70 flights had unsafe events which are contributory factors for unstabilized approaches. 51

25 Cluster 2 - Aerial view of approaches that were on average 7 knots faster than UND’s SOP; however they were stabilized. 52

26 Cluster 2 - Line graph showing the changes in altitude, airspeed, vertical speed, position (left/right of centerline), pitch and roll for the stabilized approaches. 53

27	Cluster 2 - Pie chart showing the evaluated approach configurations using the inter-cluster validation criteria. There were 69% of flights with airspeeds greater than 65 knots (the maximum airspeed was 75 knots); 27% did not trigger any events and the remaining 3% were slightly high/wide approaches.	54
28	Cluster 3 - Aerial view of the flight trajectories.	55
29	Cluster 3 - Line graph showing the altitude, airspeed, vertical speed, position, roll and pitch while on final. These approaches were approximately 7 knots faster than UND's SOP, however there were no unusual deviations in the flight trajectories during their respective descent.	56
30	Cluster 3 - Pie chart showing 55% of flights with high airspeeds (which did not surpass 75 knots) and 1% of flights which initiated their approach slightly wide.	57
31	Cluster 4 - Aerial view of the flight tracks showing 60 approaches, some of which were not fully aligned with the runway when they initiated their descent.	58
32	Cluster 4 - Line graph showing changes in altitude, airspeed, vertical speed, position, pitch and roll.	59
33	Cluster 4 - Pie chart showing the frequency of occurrence for each validation criteria.	60
34	Cluster 5 - Aerial view of the flight trajectories.	61
35	Cluster 5 - High altitude and airspeed, excessive vertical speed and bank angle as contributing factors for these unstable approaches. . .	62
36	Cluster 5 - Pie chart showing the percentage of events that were detected by the validation criteria.	63
37	Cluster 6 - Aerial view of the flight trajectories	64
38	Cluster 6 - Changes in altitude, airspeed, vertical speed, position, pitch and roll while on final. These flights were not only high and fast but also had huge changes in roll during their descent.	65
39	Cluster 6 - Pie chart showing a higher occurrence of excessive roll events in approaches that were predominantly fast.	66
40	Cluster 7 - Aerial view of the approach trajectories which contains a higher frequency of aborted approaches.	67

41	Cluster 7 - Line graph showing changes in altitude, airspeed, vertical speed, position, pitch and roll. This cluster had multiple aborted approaches, possibly due to awareness of factors that would result in an unstabilized approach.	68
42	Cluster 7 - Pie chart showing over 99 % of flights in this cluster triggered multiple unsafe events from the validation criteria.	69
43	Cluster 8 - Aerial view of the flight trajectories.	70
44	Cluster 8 - Line graph showing changes in the altitude, airspeed, vertical speed, position, pitch and roll for each approach configuration.	71
45	Cluster 8 - Pie chart showing airspeeds as high as 90 knots as unsafe events in this cluster.	72
46	Cluster 9 - Aerial view of the flight tracks.	73
47	Cluster 9 - Line graph showing changes in the altitude, airspeed, vertical speed, position, pitch and roll for the unstable approaches.	74
48	Cluster 9 - Pie chart showing five contributory events for the unstable approaches.	75
49	Cluster 10 - 1274 densely populated approaches.	77
50	Cluster 11 - 34 high/fast approaches.	77
51	Cluster 12 - 41 high/fast approaches.	77
52	Cluster 10 - Line graph showing changes in altitude, airspeed, vertical speed, position, pitch and roll while on final.	78
53	Cluster 11 - Line graph of the parameter values, while on final.	79
54	Cluster 12 - Line graph showing changes in altitude, airspeed, vertical speed, position, pitch and roll.	80
55	Pie charts for clusters 10, 11 and 12 which shows the evaluated results using the validation metrics.	81
56	Outliers - Aerial view of the trajectories for the outlier approaches.	82
57	Outliers - Line graph showing changes in the altitude, airspeed, vertical speed, position, pitch and roll for the outlier flights.	83
58	Outliers - Pie chart showing the outlier approaches that were evaluated based on the validation criteria.	84

LIST OF TABLES

Table		Page
1	An overview of the selected algorithms, their respective distance functions and processing strategy.	31
2	The sample data contained approximately 38% of stabilized approaches, the remaining flights had contributory factors for unstabilized approaches.	35
3	The silhouette coefficient of DBSCAN’s six clusters which revealed overlapping cluster memberships.	39
4	The inter-cluster validation criteria for detecting unsafe practices.	47
5	Cluster 1 - Summary statistics for each flight parameter. The highest altitude, airspeed and vertical speed during the descent was 401 ft, 95 kts and -2035 fpm respectively.	51
6	Cluster 2 - Summary statistics showing the respective range of parameter values. This cluster did not contain excessively high values or unusual changes in the descent profile.	54
7	Cluster 3 - Summary statistics for the stable approaches, although slightly fast, did not contain unusual deviations in their descent profiles.	57
8	Cluster 4 - Summary statistics showing variations in flight parameters while on final	60
9	Cluster 5 - Summary statistics for the unstable approaches showing excessive Altitude, VSI, IAS and Roll as contributory factors.	63
10	Cluster 6 - Summary statistics showing high and fast approaches some of which had a maximum roll exceeding 30 degrees during the descent.	66
11	Cluster 7 - Summary statistics showing the range of flight parameters. This cluster had the fastest VSI at -2060.03 feet per minute.	69
12	Cluster 8 - Summary statistics for the approach configurations; which shows a wider range of values for gravitational forces on the aircraft, i.e. vertical acceleration, than other clusters.	72

13	Cluster 9 - Summary statistics of flight parameters which shows the highest mean, and maximum airspeed among all clusters.	75
14	Cluster 10 - Summary statistics for the densely populated high/fast cluster which had 10+ degree changes in roll, and approaches that were high, fast, and wide with rapid descent profiles.	78
15	Cluster 11 - Summary statistics showing the range of values for the high/fast/long approaches.	79
16	Cluster 12 - Summary statistics showing the range of values for the descent profiles.	80
17	Outliers - Summary statistics for the outlier flights, which contains the highest altitude in the dataset; most of these approaches were aborted.	84
18	The performance assessment of the various algorithms.	86

ACKNOWLEDGEMENTS

First and most importantly I am grateful to my biggest cheerleader. He has been my rock, even when circumstances seemed unfavorable. Yet, words cannot express my gratitude to the Almighty God. Thank you for giving me good health, strength, inspiration and endurance to complete this research; and for blessing me with an exceptional committee.

I would like to express my sincere gratitude to my advisors Dr. Travis Desell and Prof. James Higgins. Dr. Travis Desell's guidance and expertise shaped the course of this research. I appreciate that he challenged me to explore new areas in Computer Science, because this had a profound impact on the knowledge I acquired during my studies.

I am forever grateful to Prof. James Higgins for the privilege of working on his research project and for the invaluable experience that I gained. Most importantly, thank you for your time, patience, encouragement and keen insight throughout the course of my Ph.D. studies.

I wholeheartedly would like to thank my committee members Dr. Ronald Marsh, Dr. Hassan Reza, Dr. Emanuel Grant and Dr. Paul Lindseth for their guidance throughout my graduate studies. I consider it an honor that they agreed to work with me and their collective feedback was instrumental to the overall success of this dissertation. I am also grateful to Brandon Wild and John Walberg because their immense knowledge and expertise had a significant impact on shaping my overall understanding of Flight Data Analysis.

Last but not the least, I would like to thank my family for their prayers, support and encouragement throughout writing this dissertation and my life in general.

To everyone who motivated me and believed that I possess the academic acumen to achieve this milestone – a heartfelt thank you.

ABSTRACT

The identification of atypical events and anomalies in complex data systems is an essential yet challenging task. The dynamic nature of these systems produces huge volumes of data that is often heterogeneous, and the failure to account for this will impede the detection of anomalies. Time series data encompass these issues and its high dimensional nature intensifies these challenges.

This research presents a framework for the identification of anomalies in temporal data. A comparative analysis of Centroid, Density and Neural Network-based clustering techniques was performed and their scalability was assessed. This facilitated the development of a new algorithm called the Variational Autoencoder Feature Map (VAEFM) which is an ensemble method that is based on Kohonen's Self-Organizing Maps (SOM) and Variational Autoencoders. The VAEFM is an unsupervised learning algorithm that models the distribution of temporal data without making a priori assumptions. It incorporates principles of novelty detection to enhance the representational capacity of SOMs neurons, which improves their ability to generalize with novel data.

The VAEFM technique was demonstrated on a dataset of accumulated aircraft sensor recordings, to detect atypical events that transpired in the *approach* phase of flight. This is a proactive means of accident prevention and is therefore advantageous to the Aviation industry. Furthermore, accumulated aircraft data presents *big data* challenges, which requires scalable analytical solutions.

The results indicated that VAEFM successfully identified temporal dependencies in the flight data and produced several clusters and outliers. It analyzed over 2500 flights in under 5 minutes and identified 12 clusters, two of which contained stabilized approaches. The remaining comprised of aborted

approaches, excessively high/fast descent patterns and other contributory factors for unstabilized approaches. Outliers were detected which revealed oscillations in aircraft trajectories; some of which would have a lower detection rate using traditional flight safety analytical techniques. The results further indicated that VAEFM facilitates large-scale analysis and its scaling efficiency was demonstrated on a High Performance Computing System, by using an increased number of processors, where it achieved an average speedup of 70%.

CHAPTER I

INTRODUCTION

The exponential growth of data has presented several challenges in the information era. One of which is the ability to perform large scale analysis on high dimensional data while accounting for heterogeneity. Time series data encompass these issues, and their analysis is prevalent in many disciplines including: Finance & Economics, Medicine, Neuroscience, Aerospace, Hydrology and Speech Processing [3, 4].

Definition 1 *A time-series T is an ordered sequence of n real-valued observations that are recorded based on a given sampling rate [3].*

$$T = (t_1, \dots, t_n), t_i \in \mathbb{R}$$

Temporal observations are comprised of univariate or multivariate measurements which span a given timeframe. Time series data mining is the process of analyzing the shape of the data to identify similarities between patterns on various time scales [3].

Temporal data has several characteristics which makes them difficult to manipulate in their original structure; these include: high volume, high dimensionality, heterogeneity and susceptibility to noise [3, 5, 6, 7]. Consequently, transformation steps are often employed to reduce the dimensions and extract salient features – while ensuring that the data integrity is unaffected [5, 6, 7]. Furthermore, identifying appropriate similarity measures between time series data, is essential for pattern discovery and cluster analysis, as they rely on a notion of distance to reflect underlying similarity within the data [3, 6, 7].

These challenges adds to the complexity of time series data mining, thereby making dependencies difficult to model. This affects the representational

capacity of the trained models which limits their ability to learn meaningful information [5,8]. However, there is still an increased interest in pattern detection techniques that effectively describe temporally varying phenomenon [9].

This dissertation entails the development of a new algorithm called the Variational Autoencoder Feature Map (VAEFM) which is an ensemble method that is based on Kohonen's Self-Organizing Maps (SOM) and Variational Autoencoders. VAEFM is an unsupervised learning algorithm which performs dimensionality reduction and models the distribution of data without making a priori assumptions. It is comprised of multiple stacked Variational Autoencoders which are trained to learn the manifold of time series data. After which the accuracy of the expected data is compared with the observed/predicted data. This produces a score for novelty detection, which is then clustered in a topologically preserving scheme that is based on the Self Organizing Map.

This research applied VAEFM to the analysis of flight data, which is temporal by nature and encompass the aforementioned challenges. Aircraft that are equipped with airborne flight data recorders (FDR) record flight and engine data that were retrieved from various sensors that monitors its performance. The volume of recorded data varies based on the number of observed parameters, the duration of the flight, and the sampling rate. Flight data contains critical information on the aircraft's operation during flight and can be used to identify unsafe practices, violations of standard operating procedures (SOP) and maintenance issues.

The traditional approach to flight data analysis was predominantly reactive, which by nature is performed after an accident or incident has occurred. Industry experts have advocated for proactive and predictive methods which routinely analyze flight data to identify accident precursors and mitigate risks associated with unsafe practices [10,11,12]. However, many approaches that are currently employed rely on strict threshold criteria called *exceedances*; which, albeit useful, are rigid and incapable of detecting events that oscillate near their

thresholds. They are also unable to detect anomalies due to faulty sensors, unpredictable flight patterns, or any criteria that is difficult to quantify statistically/numerically.

Consequently, there is a demand for innovative and standardized tools that are able to augment current analytical methods. Machine learning via data mining is advantageous for anomaly detection due to their proficiency in exploring data to predict new situations, discover meaningful patterns and detect trends [13, 14, 15]. They are able to explore the intricacies of complex data and explain underlying phenomena to promote the identification of anomalies that would have been unidentified using traditional methods.

However, mining flight data presents several challenges, some of which include:

- *The heterogeneous nature of flight data presents dimensionality concerns.* The volume of recorded data varies based on the FDR's sampling rate, the aircraft's make/model and the duration of flight. Further, this data is an amalgamation of continuous and categorical parameters which are recorded in various units.
- *Accumulated flight data requires scalable solutions.* Historical flight data is invaluable as they contain indicators on accident precursors which are useful for predictive analysis. Therefore, its accumulation requires scalable solutions to analyze large volumes of data and provide results in a feasible amount of time.
- *Data representation and feature extraction.* Obtaining hand-engineered features can be a tedious, and potentially error-prone process due to the volume of data. Consequently, performing this task for several hundred, or thousand, flights can be very challenging if done manually.

Due to the aforementioned concerns, unsupervised learning provides techniques for addressing them to enhance the analysis of flight data beyond the use of exceedance monitoring.

1 Scope & Objectives

This dissertation presents the development of a framework for novelty detection in time series data – i.e. the identification of new/unknown atypical events. The objective was to identify and designate types of unstabilized approaches that transpired in the *final approach* phase of flight at the Grand Forks International Airport (GFK).

Definition 2 *A phase of flight refers to a period within a flight which begins when a person boards the aircraft with the intention of flight and continues until they have disembarked [16].*

The elements of a stabilized approach involves: maintaining the correct lateral and vertical flight path, adequate energy management and small deviations in the aircraft’s orientation during its final descent. Failure to comply may result in hard landings, loss of control, tail strikes or other unfavorable events [17, 18]. Unstabilized approaches account for approximately 45% of approach and landing accidents [17]; 66% of these events are as a result of high/fast or low/slow approaches [17], which are preventable given the appropriate retraining efforts. However, the elements of unstabilized approaches vary based on the aerodynamic capability of the aircraft, the airport’s location and elevation, and external phenomena such as weather are contributory factors for these occurrences. Therefore, it is impractical to specify these criteria for every make/model aircraft and runway configuration using traditional analytical techniques.

Consequently, unsupervised learning is used to develop representative models of the problem and they are advantageous in cases where the volume of accumulated data is beyond the scope of efficient human analytical capabilities. The specific objectives of this research were to:

- Develop a framework for the extraction, preprocessing and transformation of aircraft data.

- Perform a comparative analysis of unsupervised clustering algorithms to assess their ability to identify patterns in the unlabeled data.
- Utilize the Message Passing Interface (MPI) to facilitate development of the above algorithms for High Performance Computing Systems. After which a comparative analysis of the serial and parallel algorithms evaluated their scaling efficiency.
- Integrate weather data from the National Climatic Data Center's (NCDC) Automated Surface Observing Systems (ASOS) to determine if strong tailwinds or crosswinds were contributory factors in the unstabilized approaches.

2 Motivation & Contributions

The University of North Dakota's (UND) flight training belongs to the General Aviation (GA) sector. GA is one of two branches of civil aviation that pertains to the operation of all non-scheduled and non-military aircraft [19, 20], and comprises 63% of all civil aviation activity within the United States [19, 20, 21, 22]. GA is an invaluable and lucrative industry; however it has the highest accident rates within civil aviation [19, 23].

As of 2009 GA accident and fatality rates were 7.2 and 1.33 per 100,000 flight hours respectively; and eight out of ten accidents were caused by *pilot error* [23, 24, 25, 26]. Pilot error can be attributed to poor human-system integration and for GA operators, they are often unaware of certain risks, incorrectly assess the gravity of the situation or underestimate the aerodynamic capability of the aircraft. Therefore, pilot's decisions are strongly determined by their perception of the risk which influences their risk avoidance and mitigation techniques [27, 28]. Measures are needed to educate them on unsafe practices, as unfavorable events are not random occurrences but a series of active failures facilitated by latent conditions [29, 30]. Consequently, one of the major aspects

in improving the accident and fatality rates requires educating pilots on their unsafe practices.

This research performed a comparative analysis of the following unsupervised machine learning algorithms: K-means, Density Based Spatial Clustering of Applications with Noise (DBSCAN), and Kohonen Self Organizing Map (SOM). The use of various distance functions was performed and the algorithm's scalability was assessed. This facilitated the development of the Variational Autoencoder Feature Map (VAEFM) which was designed to address limitations of the above algorithms. VAEFM facilitates dimensionality reduction and learns the manifold of the aircraft data without making a priori assumptions. VAEFM's design addressed scalability concerns of temporal data by distributing the workload across multiple processors during training, to facilitate parallel processing. The derived model was useful for knowledge discovery on various type of anomalous flight patterns. The results of which were presented using graphical means that can be easily understood to facilitate awareness of unsafe flight practices.

The VAEFM analyzed over 2500 flights, and its scalability was assessed up to 256 processors; where it produced results in under 5 minutes. The algorithm identified correlations in the data, and produced 12 clusters which comprised of stabilized approaches, missed/aborted approaches, excessively high/fast descent profiles and other contributory factors for unstabilized approaches. Outliers were detected which revealed oscillations in aircraft trajectories, some of which would have a lower detection rate using traditional flight safety analytical techniques.

CHAPTER II

BACKGROUND

Unsupervised cluster analysis is the process of exploring data, which is often unlabeled, to discover natural groupings, called clusters. Cluster membership is determined by the use of a distance metric and the selection of an appropriate metric is vital when performing analysis in high dimensional feature space as it can influence the algorithm's sensitivity to neighbors or outliers and skew hidden correlations – *the curse of dimensionality*. This section contains the background on clustering algorithms and distance metrics/measures that were used in this research.

1 K-Means Clustering

K-means [31], is one of the oldest unsupervised learning algorithms that partitions a dataset into a user specified number of clusters, denoted by k [32,33]. Given a set of data points and an integer k , the *k-means* algorithm randomly selects k centroids and seeks to minimize an objective function between each data point and its nearest centroid producing k clusters [32,34,35].

The advantages of *k-means* include its simplicity, ease of implementation and applicability to a wide range of problems [32,36]. The disadvantages include: its inability to detect outliers or the appropriate number of clusters, and inefficiently selecting initial centroids. Research has shown that an adequate choice of centroids can strongly influence both the quality of the solution and the convergence time [32]. Applications of K-means in aerospace systems include NASA's Morning Report [37,38] and Gariel's framework for trajectory clustering [39]. The k-means algorithm can be found in Appendix 1.

2 Density Based Spatial Clustering of Applications with Noise (DBSCAN)

Density Based Spatial Clustering of Applications with Noise (DBSCAN) is an algorithm that clusters unlabeled data based on the density-reachability of each item with respect to its neighbors [1]. The algorithm uses two hyper-parameters: epsilon and minimum points. Epsilon is the radius within which density reachable neighbors can be found; and minimum points is the minimum number of data points that are required to form a cluster. Unlike k-means, DBSCAN does not require the number of clusters to be specified a priori because it is designed to discover clusters of arbitrary shapes and sizes; and unclustered points are identified as outliers [40]. Figure 1 shows how DBSCAN identifies density-reachable points.

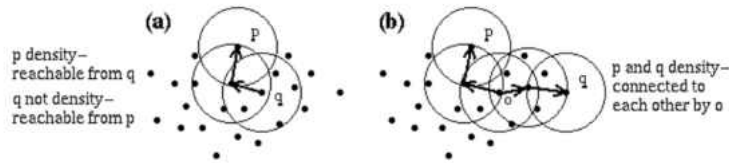


Figure 1: The DBSCAN algorithm (a) showing density-reachability and (b) density-connectivity of data points p and q, as described by Ester et. al. [1]

The advantages of DBSCAN include its ability to: automatically detect the number of clusters, identify clusters of arbitrary shapes and sizes, and detect outliers. Disadvantages include: the quality of the clusters formed depends on the distance measure employed [41], and finding appropriate values for epsilon is very challenging [42]. The DBSCAN algorithm is very difficult to parallelize due to the sequential nature that is used to identify density-reachability and density-connectivity. The pseudocode for DBSCAN can be found in Appendix 2.

2.1 Parallel DBSCAN

Parallel clustering with DBSCAN has been the focus of many research endeavors, as the algorithm's efficacy is renowned; however its computation time is expensive and the best method to improve its performance is via parallel analysis [43]. The Master-Worker paradigm, MapReduce, and disjoint-sets are some of the most prominent methods to achieve this [43, 44, 45, 46, 47, 48, 49, 50].

The master-worker model as described in [43, 44, 48], implements the DBSCAN algorithm sequentially with the master acting as a central coordinator. The master assigns tasks to each worker, performs dynamic load balancing, and merge results from workers to determine cluster assignment. One of the most computationally expensive task in DBSCAN is the region queries, requiring as much as 95% of the computation time [44]. Consequently, the workers seek to alleviate this problem by performing this task in isolation of each other, and provides the cluster(s) and/or outlier(s) to the master. However, this approach may result in bottlenecks at the master processor and incur high communication overhead between the master and workers [51].

A disjoint set structure was implemented in [51], as a way to forgo the sequential processing steps of DBSCAN by using a tree-based bottom-up approach to identify cluster membership. The algorithm treats each datapoint as a tree and merges those that belong to the same cluster until all clusters have been identified. The disjoint set structure allows the algorithm to maintain non-overlapping sets, to reveal cluster membership. Their approach is applicable on shared and distributed memory. The results indicated they were able to achieve improved computation time by using over 40 cores, on shared memory architecture; and 8,192 cores on distributed memory [51].

3 Artificial Neural Networks (ANN)

Artificial Neural Networks (ANN) are mathematical models that are inspired by the structure and behavior of a biological neural network [52, 53, 54]. ANNs are represented as a directed graph of interconnected neurons which are influenced by weighted connections. ANNs can be effectively used for classification, clustering, forecasting, pattern recognition and dimensionality reduction [55]. They possess several advantages, which include: high level of accuracy, efficiency, adaptability, and noise tolerance [52, 56]. Their disadvantages include: the inability to determine the optimal number of neurons, and identifying adequate training sets which encompass the problem domain [56]. Types of ANNs include: Feed Forward Networks, Kohonen Self-Organizing Maps, Recurrent Neural Networks, Restricted Boltzmann Machines, and Autoencoders. Figure 2 shows the mathematical representation of a neuron.

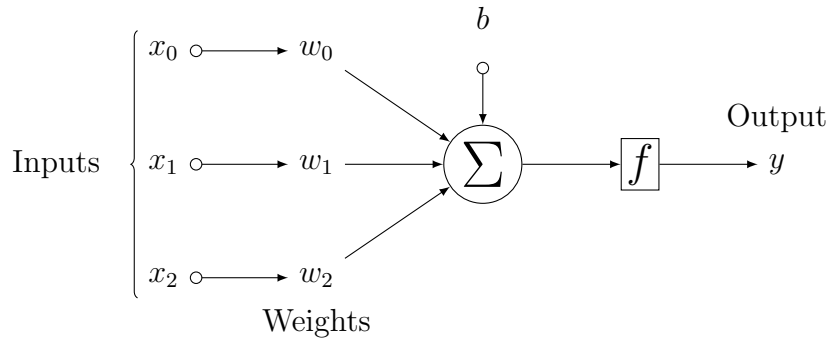


Figure 2: The mathematical representation of a neuron, which is the building block of a neural network. It has three inputs, and their respective weight vector; the weighted sum of inputs and the bias are passed into the activation function f , producing an output y .

Neural networks can be trained using supervised, unsupervised, and reinforcement techniques. Supervised learning requires previously labeled, hand-engineered data which accurately represent the problem space. However, for many real-world tasks obtaining hand-engineered features can be a tedious, and potentially error-prone process, especially for high dimensional and disparate data [57]. Their unsupervised counterpart aim to learn labels directly

from the data by using distance metrics to identify cohesion within the data (i.e. small distances which indicate similarity).

This research used unsupervised learning techniques, due to the limited availability of labeled data. ANNs provide the added benefit that they can be trained to model a problem and unlike their counterparts, DBSCAN and k-means, the ANN model can be used to generalize about new or unseen cases – which is essential for novelty detection.

3.1 Kohonen Self-Organizing Map (SOM)

The Kohonen Self Organizing Map (SOM), is a type of artificial neural network that was developed by Teuvo Kohonen. SOMs consist of an input and an output layer – which is organized in a lattice [52] and is based on an unsupervised competitive learning technique which has proven effective for exploratory data mining [52, 58, 59, 60]. SOMs advantages include their ability to project high dimensional data into low dimensional feature space [52, 60]; and identifying hidden correlations within the data while preserving their topological relationships [55, 61, 62]. Therefore data with high similarity will be mapped within close proximity and their neighboring neurons will be sensitive to similar input data [62].

In the basic SOM algorithm, each neuron’s weights are randomly initialized and during training, vectors of data are fed into the neural network. The algorithm iteratively determines which neuron’s weights are more representative of the data by minimizing an objective function; the Euclidean distance metric (shown in equation 4) is often used in literature. The winning neuron is identified and called the Best Matching Unit (BMU). After the BMU is found, each neuron’s weight vector is updated based on its proximity to the BMU (see figure 3) – which is based on a Gaussian function. This process is performed for a maximum number of epochs, during which the neighborhood radius and the learning rate monotonically decrease over time. Figure 4 shows changes to the

SOM's architecture that occurs during training. After training completes, the neural network will classify new/unseen data and clusters can be identified using a U-Matrix [63]. The SOM algorithm can be found in Appendix 3.

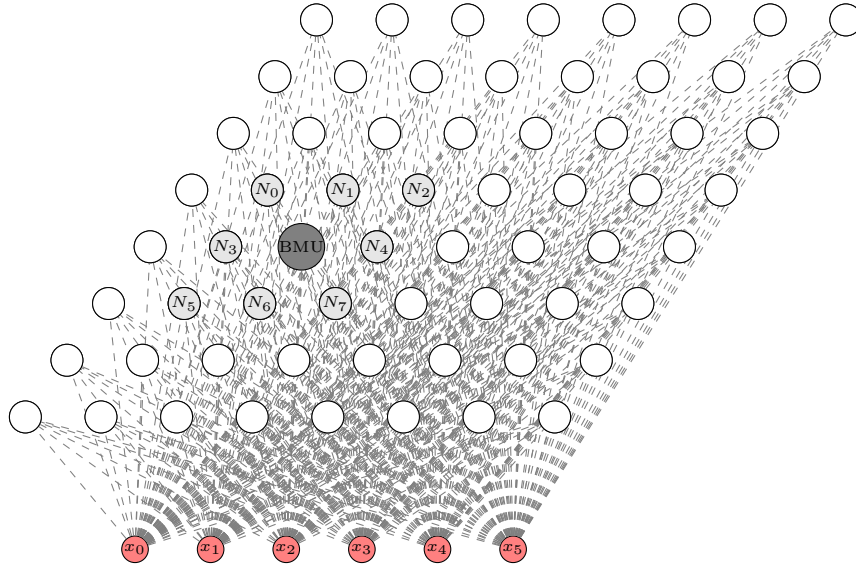


Figure 3: An 8X8 SOM with input layer $x_0 - x_5$. The input data traverses the entire network to identify the best matching unit (BMU). After the BMU is identified, its weights and those of its neighboring neurons $N_0 - N_7$, are updated during training.

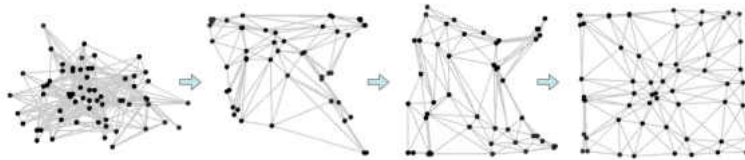


Figure 4: The unfolding of the SOM's lattice during training as the neurons weight vectors are updated to model the data.

3.1.1 Parallel SOM

Parallel implementations of self organizing maps are traditionally performed via network partition or data partition. In network partition, the map is divided among the number of processors, and each work independently to train the map. However, in data partition, the dataset is distributed among the processors and each work independently using different vectors of data to train identical copies of the same map [64, 65, 66]. In both network and data partition, a master processor is required to synchronize the results from each worker.

Network partition suffers from latency issues as each epoch requires communication between processors to determine the best matching unit; additionally much effort is needed to accurately join the map at the end of training; thereby limiting the scalability of this technique [64,67,68]. However its main advantage is that it uses the weight update rule from the basic SOM algorithm which accurately preserves the topological ordering when compared to its counterpart [64].

Data partition has the advantage of improved scalability because the parallel granularity is determined by the volume of data [64], thereby allowing one to scale up or down the number of processes when required. However, it often uses the batch update algorithm which delays the weight updates until the end of each epoch and requires that the entire dataset be available during training [64,65,68]. Research has shown that delaying the weight updates is an effective technique to improve the training time; the results are comparable and it eliminates a source of potential bias and poor convergence because it does not require the use of a learning rate parameter to be specified a priori [64].

3.2 Variational AutoEncoders (VAE)

The autoencoder is a type of unsupervised neural network which is useful for learning the manifold of unlabeled data. It also performs dimensionality reduction by enforcing a bottleneck on the size of the hidden layer and applies various regularizers, such as tied weights, to prevent overfitting the data and trivially learning the identity function [57,69,70]. The trained network, if it appropriately models the structure of the data, are useful representations for discriminative techniques [71]. Figure 5 shows the basic structure of an autoencoder which consists of an encoder function, which maps input data to a hidden representation and a decoder function which reconstructs the output from the hidden representation [69,71,72].

Data is fed into the network in a feed-forward manner and the weighted

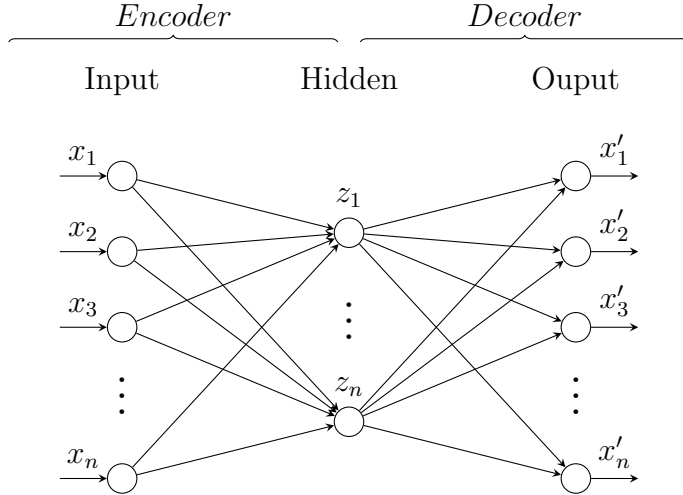


Figure 5: The basic autoencoder network which learn representations of its input data by minimizing the reconstruction error between the encoder and decoder layers in an unsupervised manner. The use of regularization, prevents overfitting and places constraints on the number of hidden neurons that are required to learn useful features. The bias has been excluded from the illustration.

product of the data, its respective weights and biases are sought and a non-linearity (i.e. activation function) is applied to restrict the range of the outputs (the sigmoid function is depicted in equations 1). This is the encoding process which produces vectors at the hidden layer, called latent variables.

$$z = \sigma \left(\sum_{i=1}^n W_{ij} x + b \right) \quad (1)$$

$$x' = \sigma \left(\sum_{i=1}^n W_{ij}^T z + b \right) \quad (2)$$

Unlike traditional feed-forward neural networks which are trained to map to a target output, autoencoders are trained to reconstruct the input data; this is the decoding process. The weighted sum of the hidden values and their respective biases are applied to an activation function to produce an output (see equation 2). The decoder uses tied weights, which is the transpose of the encoders weight matrix and is a form of regularization to prevent learning the identity function.

Training an autoencoder is the process of minimizing the reconstruction error which is achieved by optimization methods, such as gradient descent, which propagates the error through the network until the cost is minimized. After which, the encoder should contain meaningful representations provided that the decoder learned to output accurate reconstructions of the input.

Types of autoencoder networks include: Sparse, Denoising and Variational techniques. This research used variational autoencoders which incorporates probabilistic inference to further regularize the basic autoencoder so that its low dimensional features extract factors of variations in the data [73, 74, 75].

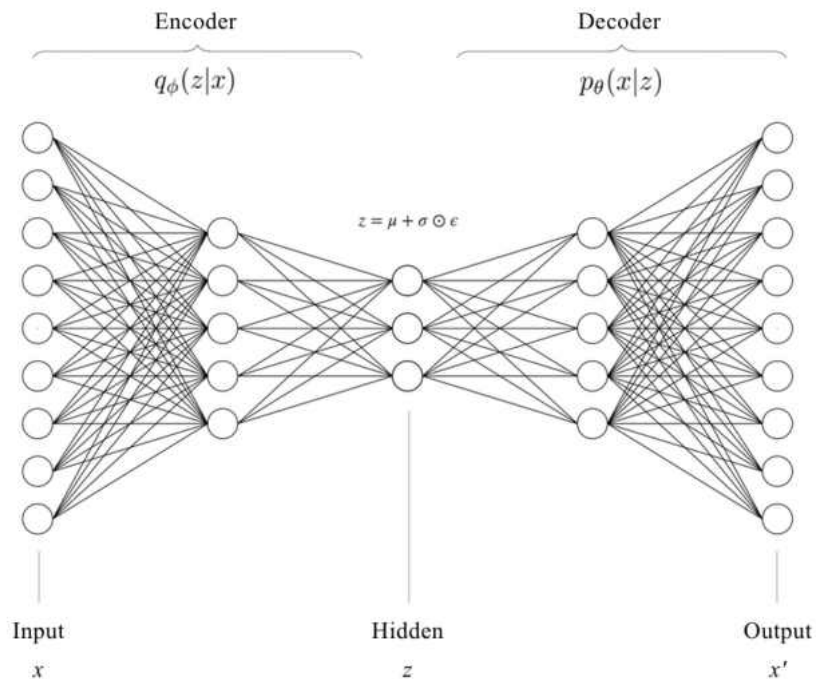


Figure 6: Variational autoencoders further regularizes the basic autoencoder by learning the joint distribution over the input data and its latent representations. It calculates the weighted sum of its inputs and bias terms – similar to the basic autoencoder. Assuming this produces a gaussian distribution, a third layer is integrated which samples μ and σ from the previous layer. The network then integrates an auxiliary random variable ϵ to generate gaussian noise, $\epsilon \sim \mathcal{N}(0, I)$ and computes $\mu + \sigma\epsilon$. Training the VAE is the process of measuring how much information is lost and minimizing that using gradient descent.

Variational Autoencoders (VAE), shown in figure 6, learns the joint distribution over the input data and its latent representations by sampling from

a unit gaussian distribution. For a given input, the encoder derives the mean and variance of the gaussian from which the latent variables are sampled. The decoder then uses the latent variables to reconstruct the original data. VAEs seek to minimize the cost function in equation 3 which is comprised of a reconstruction loss and a latent loss. The reconstruction loss assesses the similarity between the VAE’s generated data and the input data. The latent loss is used to penalize latent variables that do not follow a unit gaussian; this is based on the Kullback-Leibler Divergence [76] between the input distribution and its approximation.

$$\mathcal{L}(\phi, \theta; x) = \mathbb{E}_{z \sim q_\phi(z|x)}[\log(p_\theta(x|z))] - \mathcal{D}_{KL}(q_\phi(z|x)||p_\theta(z)) \quad (3)$$

VAEs are both inference (i.e. the encoder function) and generative (i.e. the decoder function) models [77]; as the decoder can be used to generate new data from the latent vector.

4 Evaluating Distance/Similarity

Cluster analysis relies on the use of distance functions that adequately measure similarity to determine cluster memberships. However, the selection of an appropriate distance function will enhance or deter the algorithm’s ability to identify neighbors and/or outliers. Therefore this research compared three similarity measures to identify any improvements in the quality of the cluster formations.

Definition 3 *A distance function $d(x,y)$ measures the distance between elements x and y . A metric satisfies the following:*

- $d(x, y) \geq 0$
- $d(x, y) = d(y, x)$
- $d(x, z) \leq d(x, y) + d(y, z)$

- $d(x, y) = 0 \iff x = y$.

If any of the above properties are not upheld, the distance function is not a metric (and is referred to as a distance/similarity measure).

4.1 Euclidean Distance

The Euclidean distance metric [78] measures the similarity between two vectors by calculating the sum of squared difference between one to one mappings of their elements. Equation 4 represents the Euclidean distance between two vectors x and y .

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \dots + (x_n - y_n)^2} \quad (4)$$

4.2 Mahalanobis Distance

The Mahalanobis [79] distance metric is a form of z-score which measure the distance between vectors of data, while measuring the correlation of each parameters and preserving their magnitude. The distance between two vectors x and y is shown in equation 5, and \mathbf{S}^{-1} is the inverse of the covariance matrix.

$$d(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T \mathbf{S}^{-1} (\vec{x} - \vec{y})} \quad (5)$$

4.3 Dynamic Time Warping (DTW)

Dynamic Time Warping (DTW) [80] is a dynamic programming approach to the problem of aligning time series data in such a way that an optimal path can be detected which minimizes the distance between each data point. It produces a cost, which represents the similarity between the two curves. Figure 7 shows how DTW identifies time-independent similarity between two curves.

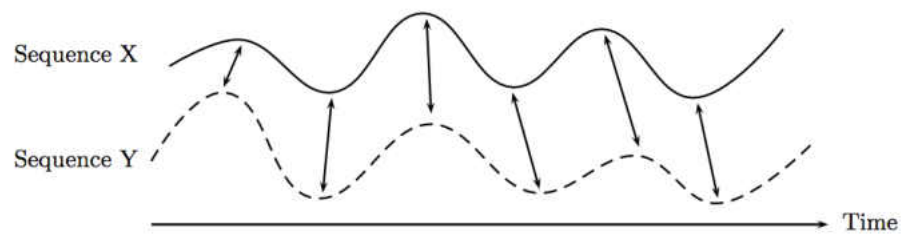


Figure 7: The DTW alignment of two time-dependent sequences (the aligned points are depicted by arrows) [2]

CHAPTER III

RELATED RESEARCH

1 Clustering Time Series Data

Analyzing time series data is a complex task due to the temporal dependencies in the data and the variability of their sampling rates. Identifying clusters in temporal data requires normalizing the dataset to a fixed range, an appropriate distance measure/metric and a clustering algorithm [3, 4, 81]. Previous research have explored many methods of analyzing time series data; some approaches employ rigorous mathematical and statistical techniques to reduce the dimensions of the data while others integrate curve similarity analysis. However, their respective approaches often employ an amalgamation of various techniques which often includes dimensionality reduction prior to machine learning. This is an important preprocessing step because there is a need to identify appropriate representations of the data as the success of machine learning algorithms depend on such [82]. Subsequently, the derived features, if they appropriately model the structure of the dataset, are useful for discriminative or predictive tasks [71]. However, the choice of data representation can potentially skew the variability in the raw data [57, 82]. Consequently, dimensionality reduction techniques are often used meticulously to minimize loss of information so their predicted classification will not be negatively affected.

1.1 Integrating Curve Similarity via Dynamic Time Warping

Many researchers have used Dynamic Time Warping (DTW) [80] (see section II 4.3) as a similarity measure for comparing temporal sequences of varying lengths [4, 83, 84, 85, 86].

Rakthanmanon et. al. [4] used dynamic time warping as the similarity measure to search trillions of electrocardiogram (ECG) data which was sampled at 256Hz. Parshutin & Kuleshova analyzed time warping techniques for time series clustering [83]. Their research analyzed the influence of DTW [80] and Derivative Dynamic Time Warping (DDTW) [86] on the topological preservation of the Self Organizing Map (SOM) [58, 59]; SOM is a popular dimensionality reduction algorithm (see section II 3.1). Their results indicated that the topology of the neurons influence the precision of the SOM's results; the use of DTW produces results with lower Mean absolute Error and DDTW produces results with lower logical error. However, their research only used the DTW cost in the SOM algorithm which treated the time series data as a single data point, which is not the most descriptive representation of such data. Romano & Scepi [84] also used DTW and SOM to classify curves. However, their experiments were on simulated data which do not contain any of the irregularities of real-world temporal data.

Clachar [87] identified anomalies in flight data using DTW's warp cost and the average rate of change to obtain an overall cost for time series subsequences which were then clustered using Self-Organizing Maps. Further analysis was then performed to identify the effect of external factors, such as weather, on the detected outliers. Somervuo & Kohonen [85] used DTW with SOM and Learning Vector Quantization (LVQ) to analyze sequences of variable lengths and rates. In their research each SOM's node is a vector sequence which allows adaptation during training to allow variable length sequences. They also use the DTW warping path as a means of averaging the difference in updating the input pattern with the weights. However, they mentioned that an invalid warping path can strongly influence the results. The DTW algorithm attempts to find an optimal warp path between two time series, however it has the potential to provide unintuitive alignments [86], therefore directly integrating the warp path into the weight update may provide misleading results during training.

Although DTW is a good distance measure, its performance is limited to certain constraints and its results cannot be directly indexed as it does not obey the triangular inequality [88]. For some time sequences it may produce unintuitive alignments, by mapping one time series to a subsection of another time series or it may not find obvious natural alignments between two sequences because their axes are scaled differently [86].

Other researchers have employed dimensionality reduction techniques, to obtain a low dimensional representation of the data prior to clustering/classification.

2 Dimensionality Reduction using Machine Learning & Statistical Approaches

Singular Value Decomposition (SVD) and Principal Component Analysis (PCA) were used by [37, 38] to identify atypical flights. These methods are geared towards expert analysts to provide insight on parameters that contribute to atypical behavior [38]. Their research indicated that analysis can be performed in one of two ways:

1. Analyzing routine events: i.e. snapshots of data parameters at critical phases of flight. This involves calculating an atypicality score using Mahalanobis [79] distance metric (see section II 4.2). The scores were then used to detect outliers – i.e. atypical flights.
2. Analyzing time intervals: this step transforms the time series data for a given phase of flight into a mathematical signature using a quadratic least squares model. A statistical summary was obtained for the least squares coefficients and atypicality scores were calculated using Mahalanobis distance [37, 38].

The above calculations are then useful for cluster analysis; the k-means algorithm was used by the authors.

Other approaches used classifiers to identify abnormalities in the descent phase of commercial jet aircraft [89, 90]. Support Vector Machines and combination rules, i.e. a statistical summary of the data, were used to rank the level of abnormality at various heights during the descent. This approach was performed on real and artificial data. Results indicated that their technique worked well for the artificial data. However, real world data presented unfavorable results because it is unlikely to have multiple abnormal descent patterns at various heights.

The application of cluster analysis on flight data has been explored by [39, 42, 87, 91]. Their research focuses on density-based, hierarchical clustering and neural network-based techniques. Li [42] applied PCA on the flight data and used Density-based spatial clustering of applications with noise (DBSCAN) [1] to cluster airline flight data from 365 flights. Their results indicated that the choice for DBSCAN's hyperparameters strongly affected the number of outliers detected. Their results also showed that the algorithm was able to identify clusters in the data and anomalies (which represent abnormal behavior). However, their research did not mention any performance implications of using DBSCAN on their dataset. One of the main disadvantages of the DBSCAN algorithm is its poor convergence time when analyzing large volumes of data.

CHAPTER IV

THE DATA REPOSITORY

The University of North Dakota's (UND) fleet of Cessna 172S model aircraft are equipped with the Garmin G1000 system, which contains many integrated features. One of which is its airborne flight data recorder (FDR) which has a 1 hertz frequency and records 64 flight and engine parameters from sensors onboard the aircraft. On average, the duration of a typical flight is 80 minutes. For the 1 hertz time frequency this produces a 64-dimensional vector size of 4800×64 (i.e. 4800 seconds, 64 flight parameters). Flight data holds key information on the aircraft's operation during various phases of flight; and can be used to identify unsafe practices, violations of standard operating procedures (SOP) and maintenance issues which are integral aspects of aviation safety management. Consequently, at the end of each flight, the data is retrieved from the aircraft and uploaded to a data repository for further analysis. Figure 8 depicts this process.

1 Overview & Architecture

The National General Aviation Flight Information Database (NGAFID) is a joint university-industry-FAA initiative that is responsible for the curation, dissemination and analysis of flight data for the General Aviation (GA) sector. The NGAFID is supported by an Intel *x86* – 64 architecture with 12 physical cores, 12 threads of execution, 284 GB of RAM and 24 CPUs. It uses a LAMP software bundle with a MySQL relational database. As of December 2016, the database contains de-identified data from over 370,000 flights totaling 650,000 flight hours. This produced over 1.5 billion database rows which accumulates over 2 terabytes of storage. The University of North Dakota (UND) is the major

contributor to the NGAFID and their data comprises over 95% of the aforementioned statistics.



Figure 8: Flight data is recorded using an airborne recording device which may encrypt the data. At the end of each flight the data is manually retrieved from the aircraft, decrypted and converted into a CSV format before it is uploaded to the NGAFID.

UND has routinely contributed to the NGAFID since march 2011 to date. Even though the NGAFID was designed for the GA sector, which UND flight operations belong to, the methodology in this research is not confined to GA. However, this data source is useful in demonstrating the utility of the analytical techniques for identifying atypical flight patterns which contribute to unstable approaches. Furthermore, the growing size of the repository presents several *big data* challenges requiring scalable solutions. Consequently, any useful algorithmic design should facilitate distributed analysis in order to provide results in a feasible about of time to address scalability issues.

The NGAFID is used as the data source for this research and the analytical process is initiated by extracting samples of the raw data and preprocessing them prior to cluster analysis.

2 Data Preprocessing & Transformation

The preprocessing and transformation steps encompassed: sanitizing the data, feature extraction, data transformation and phase of flight identification. Samples of raw data were first obtained from the NGAFID and sanitized to remove noise and invalid/incomplete data. After which the features that are useful for the analysis are selected. There are approximately thirteen parameters

which contain useful indicators of the aircraft’s approach configuration; they are: Mean Sea Level Altitude, Radio Altitude, Indicated Airspeed, Vertical Speed, Course, Heading, Roll, Pitch, Vertical Acceleration, Lateral Acceleration, Engine RPM, Latitude, and Longitude.

The extracted data undergoes transformation steps, which is an integral aspect due to the various scales of flight parameters (e.g. knots, degrees, feet, etc). This ensures that data has the same scale so that features which are orders of magnitude larger than others do not thwart the analysis. The residuals of the extracted data were derived, see equation (6), by obtaining estimates of the population mean \bar{X} using bootstrap resampling [92]. The residuals were then divided by the range (equation 7) and rescaled to [0.01, 0.9]. The geographic coordinates (e.g. latitude and longitude) were normalized in a different manner, by creating a geometric fence (see figure 9) around the airport and applying max-min normalization to rescale the coordinates within the specified region.

$$X' = X - \bar{X} \tag{6}$$

$$\frac{X'}{MAX(X') - MIN(X')} \tag{7}$$

Subsequently, the data was preprocessed to extract the approach phase of flight.

2.1 Phase of Flight Identification and Designation

Amidan & Ferryman [37, 38] did pioneering research with developing representative mathematical signatures of flight data and concluded that the analysis requires preprocessing the raw data to extract the various phases of flight, prior to performing any statistical or machine learning techniques. This will reduce the dimension and allow fine-grained analysis; because analyzing the entire duration could obscure anomalies. This research incorporates their



Figure 9: A geometric fence, indicated in red, that was created around the Grand Forks International Airport's runway 35R.

suggestions on preprocessing flight data to extract the phases of flight and analyzing each phase independently to exude a higher degree of accuracy.

Aircraft undergoes approximately ten distinct phases of flight, with each phase having sub-phases [16]. This research focused on abnormal descent patterns in the *final approach phase*.

UND's fleet predominantly operates from the Grand Forks International Airport (GFK). The flight data was preprocessed to identify the final approach phase for GFK's runway 35R as follows:

1. Obtain the geographic coordinates for the runway under analysis.
2. Create a search space of approximately 200 feet around the anticipated descent point; the location of the Visual Approach Slope Indicator (VASI) or Precision Approach Path Indicator (PAPI) can be used.
3. Identify when the aircraft has entered the airfield's traffic pattern. This occurs around 800 feet above ground level (AGL) and when the aircraft is within a 5 mile radius from the centroid of the airport.
4. Verify if the aircraft is descending and its power is decreasing (i.e. the vertical speed is less than zero and the engine's power has reduced below

2000 revolutions per minute).

5. Calculate the course differential; this is the difference between the aircraft's course and the runway's course. If the difference is ± 15 degrees, the aircraft has completed its turn and is in alignment with the runway; this is indicative of an approach. At this point the altitude should be below 500 feet AGL.
6. Obtain the flight data starting from (5), until the aircraft has surpassed the anticipated descent point, and/or the airspeed has decreased below 30 knots.

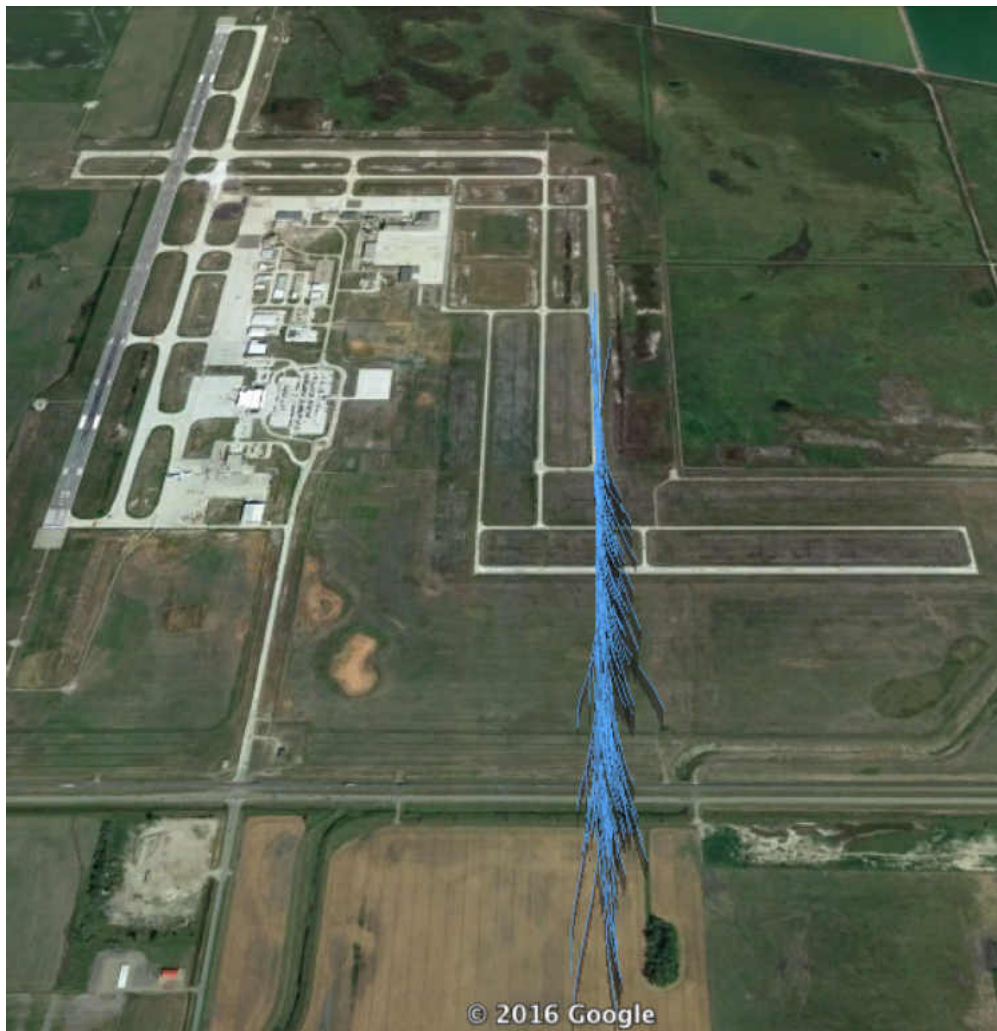


Figure 10: A sample of flight trajectories, i.e. the approach configuration, which made their final approach on runway 35R.

The aforementioned steps reduced the vector space, extracting only the

approach from each flight and the average duration is 40 seconds. However, only the first thirty seconds is needed, to prevent obtaining the landing/taxiing data. From here on, this extracted data is referred to as the snapshot of the aircraft's approach configuration. Each snapshot will be analyzed to identify anomalous descent patterns using machine learning techniques. Figure 10 shows an aerial view of the extracted trajectories for each aircraft's approach configuration.

CHAPTER V

RESEARCH METHODOLOGY

Flight data was obtained from the NGAFID and underwent several preprocessing and transformation steps to identify and extract the approach configurations from the data. The results were then normalized and saved to a text file which was subsequently transferred to UND's High Performance Computing (HPC) center where the analysis was performed. Figure 11 illustrates the process flow.

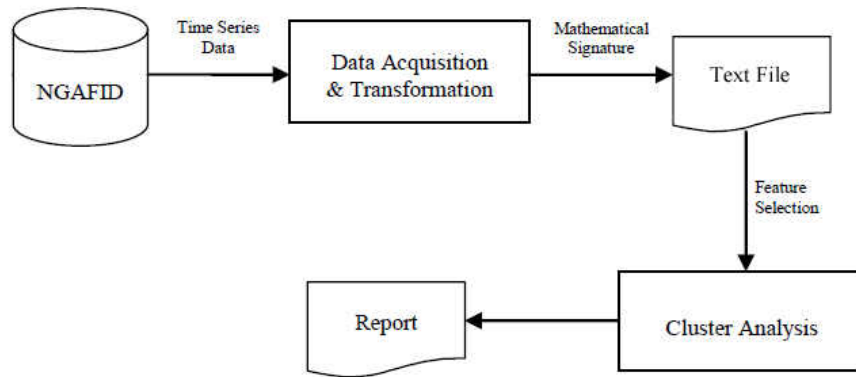


Figure 11: The temporal flight data was obtained from the NGAFID and underwent preprocessing and transformation steps. The data was saved into a text file which was transferred to the high performance computing (HPC) center, where the various clustering algorithms were performed; after which their respective results were gathered.

The algorithm's results were gathered using a Beowulf HPC cluster with 32 dual quad-core compute nodes (for a total of 256 processing cores). Each compute node has 64GBs of 1600MHz RAM, two mirrored RAID 146GB 15K RPM SAS drives, two quad-core E5-2643 Intel processors which operate at 3.3Ghz, and run the Red Hat Enterprise Linux (RHEL) 6.2 operating system. All 32 nodes within the cluster are linked by a private 56 gigabit (Gb) InfiniBand (IB) FDR 1-to-1 network. The code was compiled and run using

Python, Open MPI [93] and MPI4Py (MPI for Python) [94, 95, 96], to allow highly optimized use of this network infrastructure.

The analysis was performed in two phases: 1) the preliminary data exploration efforts and 2) the detailed analysis which includes the scalability assessment. All flights were obtained from UND's Cessna 172S model aircraft which made their descent at the Grand Forks International Airport's runway 35R.

In the preliminary analysis, a random sample of 1500 flights were selected and their respective approach configurations were sought. Subsequently, the average values for each parameter in the approach configurations were obtained, as the preliminary efforts sought to analyze the efficacy of select algorithms and distance functions for analyzing flight data. Consequently this is a form of average case analysis which proved useful for visualizing clusters in the data in lower dimensions. Three parameters were used: Mean Sea Level Altitude (MSL), Vertical Speed (VSI) and Indicated Airspeed (IAS). The algorithms that were useful for identifying clusters and outliers in the three dimensional data, were then optimized for the detailed analysis in high dimensional feature space.

The detailed analysis sought to identify clusters and outliers in the time series data (i.e. the entire approach configurations). A new sample of 2582 flights that occurred in June 2015 was obtained from the NGAFID and the following 9 flight parameters were used: radio altitude, indicated airspeed, vertical speed, pitch, roll, engine RPM, vertical acceleration, latitude and longitude. A comparative analysis was performed of the serial and asynchronous implementations of the algorithms and their respective distance functions. However, unlike the preliminary analysis, the Mahalanobis distance metric was not used because it is very time-consuming; comparable results can be obtained by calculating a z-score which standardizes the data. Dynamic Time Warping was integrated as an alternative similarity measure and its results were compared with Euclidean distance metric. Table 1 summarizes the various algorithms, their respective distance functions and processing strategy.

Table 1: An overview of the selected algorithms, their respective distance functions and processing strategy.

Algorithm	Distance Function	Serial / Parallel Processing	Summary
K-Means	Euclidean	Serial	A random sample of 1500 flights were selected. The average values of each parameter in the approach configurations were calculated. The average case analysis was only performed in the initial data exploration attempts to gain insight into the data. The selected parameters were: Mean Sea Level Altitude (MSL), Vertical Speed (VSI) and Indicated Airspeed (IAS).
	Mahalanobis		
DBSCAN	Euclidean		
	Mahalanobis		
DBSCAN	Euclidean	Serial	A sample of 2582 flights, which occurred in June 2015 was obtained. The detailed analysis used 9 flight parameters: radio altitude, indicated airspeed, vertical speed, pitch, roll, engine RPM, vertical acceleration, latitude and longitude. The entire sequence of time series data was analyzed (not their averages).
		Parallel	
	DTW	Serial	
Parallel			
SOM	Euclidean	Serial	
	DTW & Avg. Rate of Change		
	DTW	Parallel	
VAEFM ^a	Reconstruction-based	Parallel	

^aVariational Autoencoder Feature Map

CHAPTER VI

RESULTS & EVALUATION

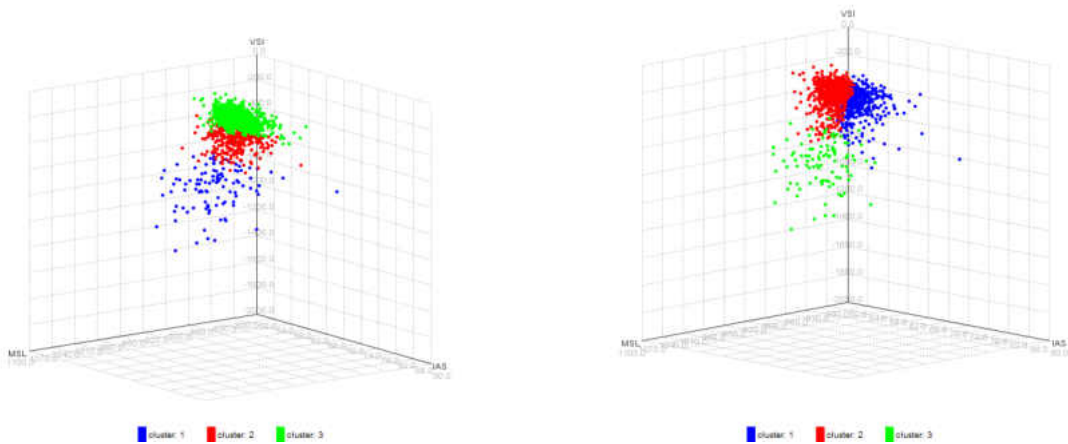
This section contains the results for the serial and distributed clustering algorithms. The intent was to analyze their performance when the dimensions increase and assess the scalability of their distributed implementations.

1 Serial Clustering Results

A random sample of 1500 flights was obtained from the NGAFFID, for all approaches that made their descent on runway 35R at the Grand Forks International Airport (GFK). The preliminary analysis sought to compare the efficacy of each algorithm and was performed in three dimensional feature space (each dimension corresponds to a flight parameter). The selected parameters were: Mean Sea Level Altitude (MSL), Vertical Speed (VSI) and Indicated Airspeed (IAS); and the average values of each parameter in the approach snapshots were calculated. Performing the analysis with the averages was only done in the initial data exploration attempts to gain insight into the data and to visualize the quality of the cluster formations.

1.1 K-Means

The *k-means* algorithm was used as one of the first data exploration efforts in this research. Figure 12 shows the results from using this algorithm to identify three clusters in the sample data. The analysis was performed using two distance metrics: Euclidean distance [78] (equation 4), as recommended by the original *k-means* algorithm, and Mahalanobis [79] distance metric (equation 5) to determine which metric accurately represents distance in three dimensional data.



(a) Three clusters that were found using the Euclidean distance metric.

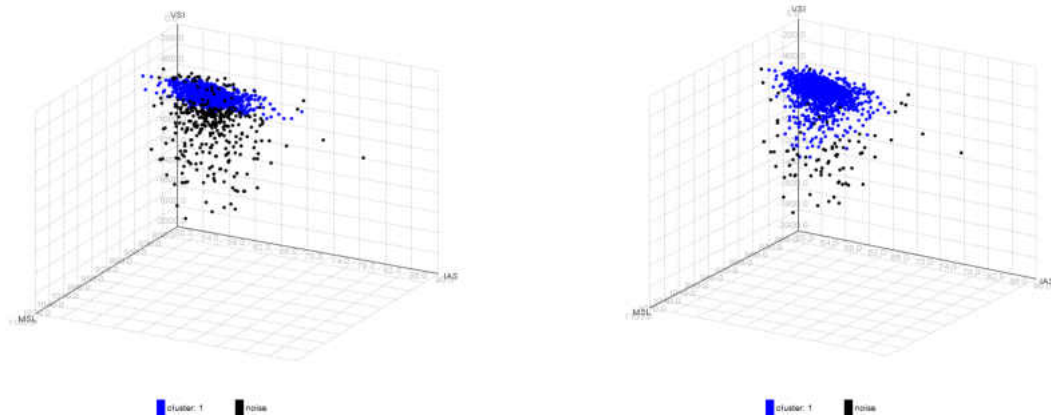
(b) Three clusters that were found using the Mahalanobis distance metric.

Figure 12: Graphical illustration of three clusters identified in 1500 flights when using the k-means algorithm. The choice of distance metric or scale of the data did not influence the selection of intuitive clusters; however it determined how the dataset was partitioned.

The graphical illustrations in figure 12 reveals that k-means identified k-clusters, even if the selected value for k was not optimal. The success or failure of k-means relies strongly on choosing appropriate values for k, which is very challenging in high dimensional data, unless one has the ability to visualize it. Further, the distance metric and scale of the data can strongly influence the outcome the data partitions. Due to this algorithm’s difficulty in detecting meaningful clusters (without additional heuristics); and its inability to identify outliers, which may be indicative of abnormal flights, the algorithm was not used further nor was it optimized. The initial data exploration indicates that k-means may perform poorly in high dimensional feature space (i.e. beyond the 3 dimensional case as depicted in this example).

1.2 DBSCAN

DBSCAN was subsequently used on the aforementioned sample of 1500 flights and its results was compared to its centroid-based counterpart, *k-means*. Figure 13 shows the results of applying DBSCAN, and as with k-means, the Mahalanobis and Euclidean distance metrics were used. The results illustrate



(a) The results from DBSCAN using the Euclidean distance metric; one cluster was identified and 355 outliers.

(b) The results from DBSCAN using the Mahalanobis distance metric; one cluster was identified and 84 outliers.

Figure 13: Graphical illustration of the one cluster and various outliers found using DBSCAN. The choice of distance metric influenced the cluster formations and the number of outliers that were detected.

that DBSCAN was able to identify one cluster of high density within the dataset and several outliers. The total number of elements in the cluster and the number of outliers was influenced by the scale of the data and the distance metric. However, there was a consistent identification of one cluster irrespective of the distance metric that was used.

From the initial analysis, the DBSCAN algorithm demonstrated improved results over k-means. However, the algorithm’s computational time is very demanding. Further use will require parallel analysis to produce results in a feasible amount of time.

1.3 SOM

The self organizing map required two types of data: 1) a training set and 2) a test set. The first is used to train the network so that it is able to generalize about data that it was not trained with; and the latter is used to demonstrate the predictive capability of the trained network. However, due to the limited availability of labeled data, the training set was obtained by using bootstrap sampling on the test set (i.e. the 1500 flights). Subsequently, the vectors of

Table 2: The sample data contained approximately 38% of stabilized approaches, the remaining flights had contributory factors for unstabilized approaches.

Cluster	% Occurrence
High	14.8
Long	8.5
Short	17.02
Steep	8.5
Fast	12.7
Stable/Normal	38.5

training data was fed into the network based on the steps outlined in algorithm 3. The network was trained until the average training error was less than 10^{-7} . After which, the test set was projected unto the network.

The preliminary results were manually validated by safety experts at UND because, unlike DBSCAN and k-means, SOM produced mappings of flight data that were highly correlated; therefore, it would be difficult to determine the utility of the results or optimize the neural network without assistance from experts. The initial validation detected flights with 10-15 degree changes in roll just above touchdown, unsafe low-level maneuvers and exceptionally fast approaches with rapid deceleration by touchdown.

However, the rate of false-positive flights were very high; false-positives are flights that were considered normal and were mapped to neurons with anomalous flights. The high rate of false positives could be as a result of obtaining a poor selection of flights for the training set. Consequently, the neural network was retrained and the validated flights were used to augment the new training set and was used to fine-tune the network during the latter stages of training. The results underwent a second round of human validation to identify improvements in the SOM’s mappings. The results indicated that SOM’s rate of false positives decreased to less than 5% of the dataset; which is a significant improvement from the initial validation efforts.

Table 2 shows each anomaly with their respective occurrence; and the results indicate that there were approximately 38.5% of stabilized approaches.

Figure 14 shows the Unified Distance Matrix (U-Matrix) of a trained SOM. The U-Matrix is a visualization method that was suggested by Ultsch [63] as an alternative method of identifying clusters of neurons based on their relative distance to each other. The U-Matrix is a hexagonal lattice that displays high dimensional data into a 2-dimensional manner that can be easily understood by humans. It is color coded on a red-green scale, where red depicts a large distance (high dissimilarity) and green is indicative of small distances.

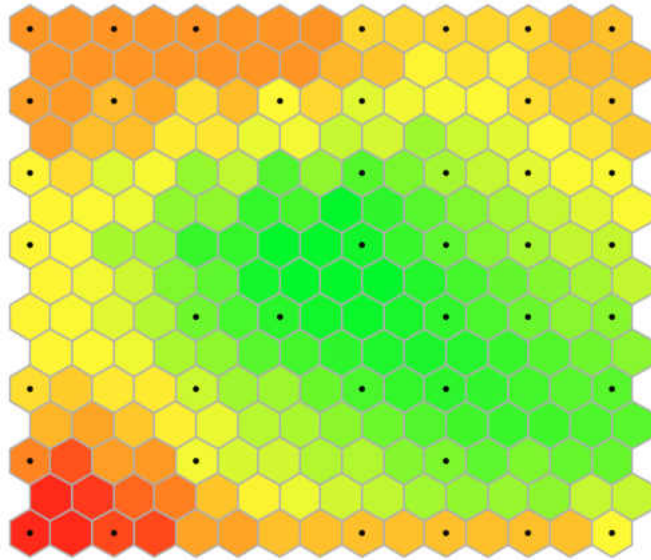


Figure 14: The U-Matrix of a trained SOM, which is color coded; red indicates regions of large dissimilarity and green shows cohesion between neurons.

Training the Self-Organizing Map is a time consuming process, due to the frequency that the training data is presented to the network. Also, the size of the network adds to the computation time, as the data is presented to each neuron for comparison. Consequently, SOMs computation time is a function of the number of epochs, the size of the training set and the network size (each of which varies based on the problem). Therefore, there is a need for parallel analysis to train the neural network to facilitate the required scalability.

2 Distributed Clustering Results

The k-means algorithm did not produce effective quality clusters in lower dimensions as it is very difficult to identify appropriate values for k . It was also unable to identify outliers therefore it was not parallelized.

A new sample of 2582 flights, which occurred in June 2015, was selected and preprocessed to obtain their respective approach configurations. The following flight parameters were used in the analysis: radio altitude, indicated airspeed, vertical speed, pitch, roll, engine RPM, vertical acceleration, latitude and longitude. The previously used average case analysis was solely to demonstrate preliminary research efforts in identifying clusters and visualizing the results. However, from here on, the entire sequence of temporal data will be analyzed. The Dynamic Time Warping, which is advantageous in comparing the alignment of time series data, was used as an alternative to Euclidean or Mahalanobis distance metrics.

This section evaluates the scalability of the asynchronous algorithms on the actual approach configuration (not the averages).

2.1 DBSCAN

The parallel DBSCAN algorithm, as described in [46], was implemented and its computation time was compared to the serial algorithm.

The serial DBSCAN algorithm had an average runtime of 2013 minutes (33 hours), whereas its asynchronous equivalent, i.e. the master worker model with two processors, had an average runtime of 2060 minutes (34 hours). Therefore, the serial algorithm is 3% faster than its asynchronous counterpart on a single processors. However this time difference is expected due to additional overhead for the MPI communication calls and the master process synchronizing the results.

As the number of processors increased, to 4 processors, the asynchronous DBSCAN algorithm demonstrated a 23% improvement in computation time over

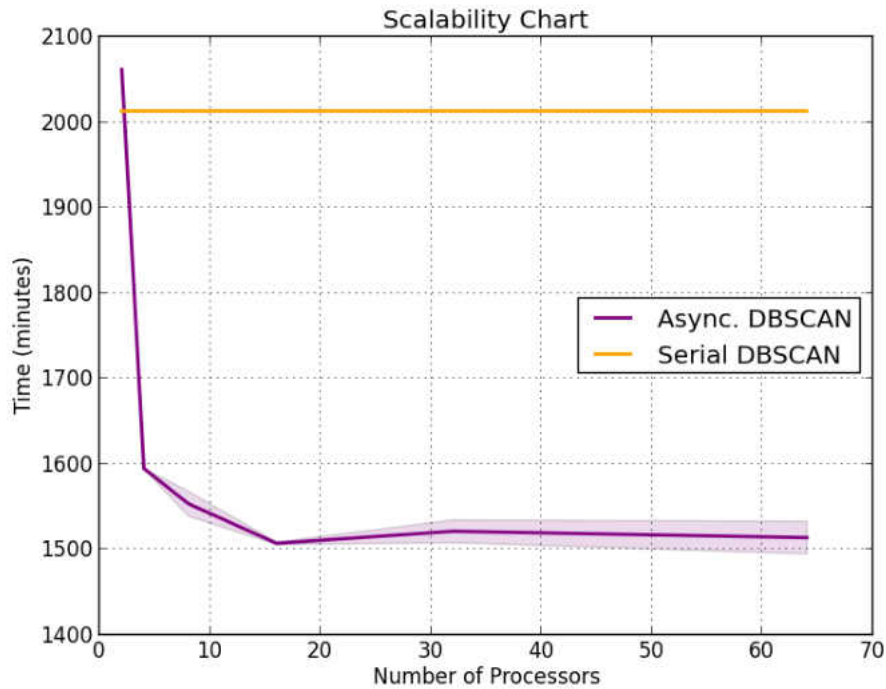


Figure 15: The computation time of the DBSCAN algorithm, as the number of processors increased, and its serial counterpart. The performance was assessed by executing the algorithm multiple times. The average run-time, i.e. the solid line, was calculated and the shaded regions depicts variations in the computation time, which occurs due to bottlenecks at the master processor. On average, the slowest computation time was 2060 minutes when using 2 processors and the fastest was 1507 using 16 processors.

its two process counterpart and it gained a 21% improvement over the serial algorithm. When using 8 and 16 processors, there was a 23% and 25% respective improvement over the serial algorithm. Figure 15 shows there is a decrease in processing time as the number of processors increased. However, on average, the speedup is sluggish as it is very difficult to perform DBSCAN in parallel because the workload became imbalanced when the master synchronizes the data. Consequently, it is very challenging to adjust the parallel granularity to achieve paramount speedup with this technique. The slowest computation time achieved was 2060 minutes, when using 2 processors (i.e. one master and one worker) and the fastest was 1507, using 16 processors. The mean and standard deviation of the computation time across all processors was 1625 and 215 minutes respectively.

Table 3: The silhouette coefficient of DBSCAN’s six clusters which revealed overlapping cluster memberships.

Cluster	Average Cohesion	Average Separation
1	0.009	0.008
2	-0.054	-0.001
3	0.248	0.001
4	0.206	0.001
5	0.587	0.003
6	0.126	0.0006

Figure 15 shows that the best computation time occurs around 14 and 18 processors; beyond that, there would not be a significant improvement in computation time based on the resource utilization/overhead. For example, 64 processors took an average of 1514 minutes which is not a significant improvement when compared to 16 processors as it is using 4 times the number of processors.

However, both the serial and asynchronous algorithms consistently identified six clusters and 1669 outliers. Due to the high dimensions of the data, visualizing the clusters was very challenging. They were instead validated by using an internal cluster validation metric called the silhouette coefficient [97] (see equation 8). The silhouette values range between -1 and 1; values that approach 1 are indicative of good inter-cluster membership or intra-cluster separation. Table 3 shows that the average cohesion and separation are centered around 0 which indicate overlapping cluster memberships. Cluster 5, contained the best inter-cluster results.

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \quad (8)$$

As the number of dimensions increased, it was very challenging to find appropriate values for DBSCAN’s hyperparameters. Consequently, the clusters that were identified were not fully separable.

2.2 SOM

The data partition based SOM algorithm was used and it can be found in appendix 4. The objective function, which was previously based on the Euclidean distance metric, was replaced with Dynamic Time Warping due to its innate ability to identify similarity in temporal data.

Figure 16 shows the performance of the algorithm's scalability as the number of processors increased. The asynchronous algorithm running on two processors took approximately 27 hours, as opposed to its serial counterpart which took approximately 39 hours. Using twice the number of processors resulted in an average computation time of 10 hours, which is a 75% speedup over the serial algorithm; and 62% improvement over the asynchronous 2 processor version. Further increasing the number of processors to 8 and 16 reduced the average computation time by 55%. However, when the number of processors surpassed 16, the computation time increased due to the bottleneck at the master. On average, the fastest computation time was 239 minutes when using 16 processors and the slowest was 1627 minutes using 2 processors. The mean and standard deviation across all processors were 572 and 530 respectively. The standard deviation is fairly high, due to huge variations in the computation time across processors.

The async SOM algorithm demonstrated the fastest computation time when compared with async DBSCAN (see figure 17), and its performance is over 6 times faster. The evaluated results indicated that SOM identified strong correlations in the data. However, one of its disadvantages lie in the possibility of overfitting and overtraining the data which may occur when the neighborhood radius decreases [98, 99]; and adjusting the size of the network does not alleviate this issue.

Consequently, additional heuristics are needed to reduce the bottleneck at the master processor and prevent overfitting the training set. A new algorithm was developed which is based on Self Organizing Maps and Stacked Variational

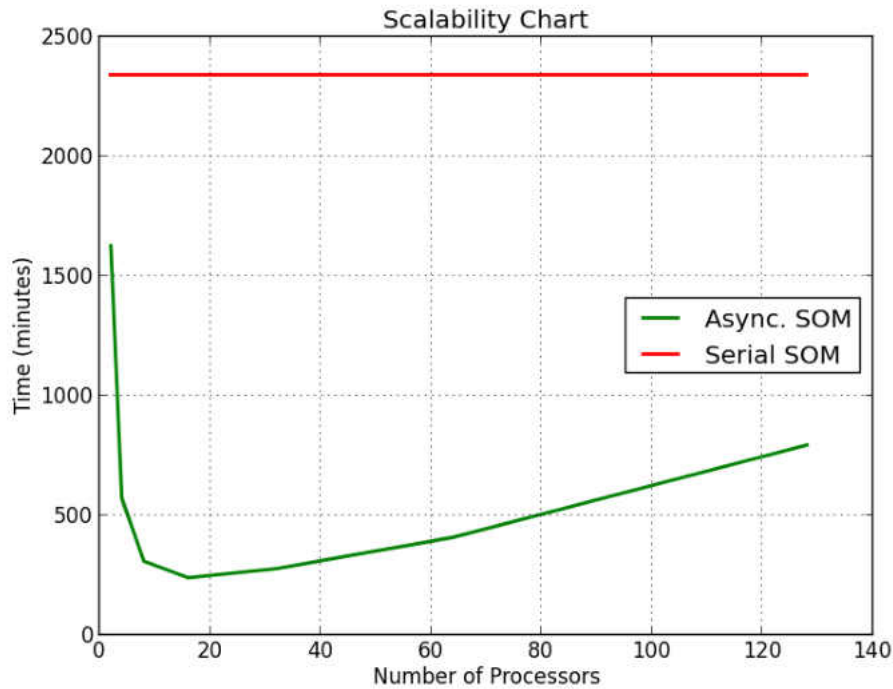


Figure 16: The computation time of the asynchronous SOM algorithm, as the number of processors increased, and its serial counterpart. Multiple runs were also performed for the asynchronous algorithm, however variations in the computation time was miniscule.

Autoencoders; together they learn the data manifold to prevent overfitting. They also facilitate improved scalability by merging the benefits of both data partition and network partition to train the neural network.

2.3 Variational Autoencoder Feature Map (VAEFM)

The new Variational Autoencoder Feature Map algorithm, is the combination of two neural networks: 1) the variational autoencoder, and 2) the self-organizing map. This methodology seeks to leverage the weights of trained variational autoencoders to pre-train a self-organizing map. MPI was used to facilitate the development of an asynchronous implementation and the computation time was compared with that of SOM to identify any improvements in scalability, because the parallel SOM algorithm was not scalable beyond 16 processors. The steps of the algorithm are outlined below and the pseudocode can be found in Appendix 5.

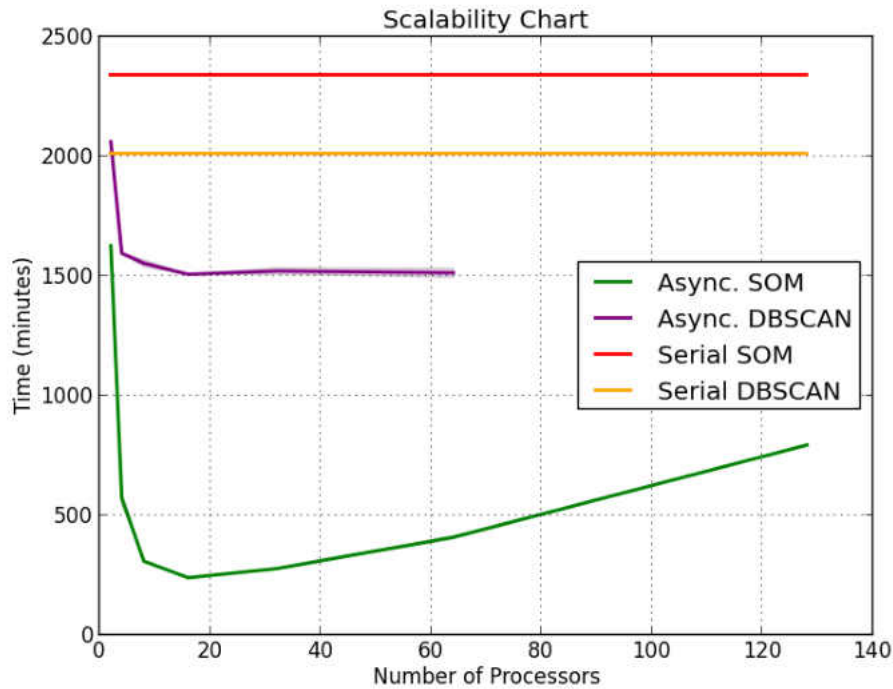


Figure 17: The computation time of the serial and asynchronous SOM algorithm compared with DBSCAN’s serial and asynchronous algorithms. The serial SOM performed slower than both serial and async DBSCAN because the algorithm endures more computation, when the data is compared to each node in the neural network. However, the async SOM algorithm outperformed its serial counterpart, as well as both the serial and async DBSCAN algorithms; its fastest computation time was 239 minutes when using 16 processors, as opposed to DBSCAN’s 1507 minutes.

1. Initialize the architecture of a SOM network; excluding the weight vectors.
2. Obtain samples of approach configurations which have been preprocessed and normalized.
3. Derive a training set from the above data by using random sampling techniques. Previously validated/labeled flights can also be used. This research used the previously validated SOM results.
4. Generate stacked variational autoencoder networks to model each flight in the training set; each layer in the stack is indicative of one time step in the flight’s approach configuration. Therefore, if an approach spans 30 seconds, there will be 30 autoencoders which are stacked sequentially to model each

second in the approach (see figure 18). Furthermore if the training set is comprised of 100 flights/approaches, there will be 100 autoencoder networks; each of which is a stacked VAE to model each respective approach.

5. Train each stacked autoencoder network in a layer-wise manner using gradient descent, with momentum, to minimize the reconstruction error.
6. After training is completed, and the error is minimized, extract the weight vectors from each layer in the stacked VAE. These weights will be used to pre-train the SOM.
7. Assign the aforementioned weight vector to one neuron in the uninitialized network (in step 1 above). Apply the neighborhood function of the SOM algorithm to ensure that neighboring neurons update their weights based on their proximity to the selected neuron.
8. Perform steps 5 - 7 for all flights in the training set.
9. A test set is fed into the trained SOM network and the data traverses all neurons to identify which has the smallest prediction error (i.e. the BMU). Each flight in the test set is mapped to their respective BMUs.

The steps outlined above was applied to each approach configuration in the training set. This new algorithm allows for improved scalability as each stacked autoencoder network can be trained independently which allows the algorithm to scale the number of processors to the size of the training set. After which the weight vectors are obtained from each autoencoder and undergoes an adaptation step to integrate their weights into the uninitialized self organizing map.

When adaptation is completed, copies of the trained VAEFM is shared with all processors and the test set is distributed evenly among them. Each process works independently to identify the BMU for each approach in the test set.

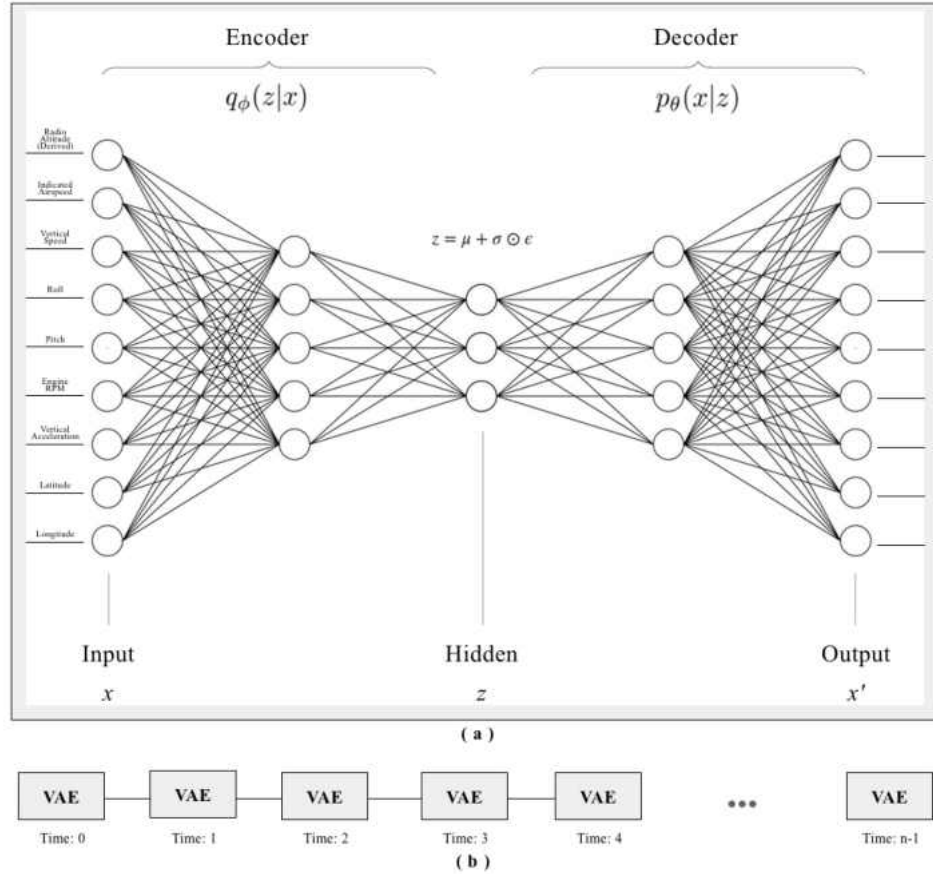


Figure 18: Graphical illustration of the VAE (a) which accepts 9 flight parameters, and (b) depicts the block diagram of (a) which is stacked sequentially to model each second in the temporal data.

After which, the results are sent to the master process which merges them and creates visualizations of the data manifold.

The above steps are major enhancements over the original SOM algorithm which only maps inputs to outputs. Also, the new architecture is representative of a deep network, and the depth is based on the duration of the approach configurations.

2.3.1 Performance Evaluation

The purpose of VAEFM was twofold: 1) improve the scalability of async SOM beyond 16 processors and 2) enhance the training algorithm to minimize overfitting. Figure 19 shows the computation time of the VAEFM algorithm as the number of processors increased to 256. The computation times of serial and

async SOM were superimposed to demonstrate the point at which VAEFM outperformed the algorithms. Initially VAEFM performs slower than async SOM because its algorithm contains longer analytical steps. Consequently, when using 8 and 16 processors, the respective computation time was approximately 85% and 58% slower than async SOM. However, VAEFM demonstrates improved computation time as the number of processors increased (even though it has not surpassed SOM at this point).

VAEFM outperformed SOM when using 32 processors and had an average computation time of 214 minutes as opposed to async SOM's 278 minutes; this is a 23% improvement. Furthermore, as the number of processors increased to 64, 128, 256, VAEFM demonstrated an average speedup of 70%.

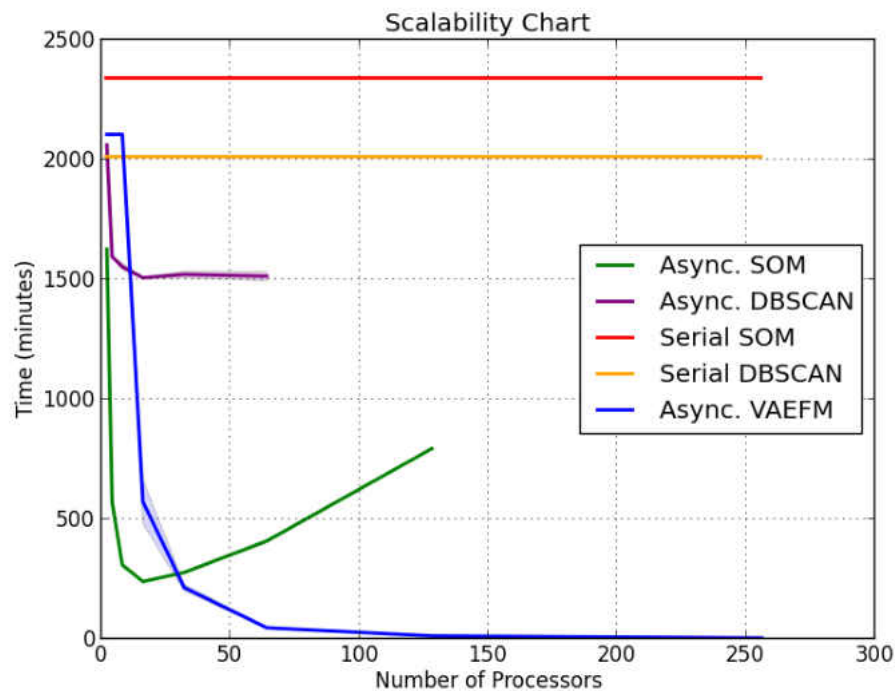


Figure 19: The computation time of the VAEFM, Serial SOM, Async SOM, Serial DBSCAN and Async DBSCAN algorithms as the number of processors increased. VAEFM demonstrated an average speedup of 70%, and its scalability was demonstrated up to 256 processors – where it analyzed over 2500 flights in under 5 minutes.

VAEFM's maximum computation time was 2106 minutes and its fastest was 5 minutes when using 8 and 256 processors respectively. There were major

improvements in performance, as the number of processors increased, when compared with the computation time of SOM and DBSCAN. This algorithm has demonstrated improved computation time, beyond 16 processors. The results will be validated in the subsequent section to verify its utility for novelty detection.

2.3.2 Evaluation of the Data Manifold

Figure 20 shows the data manifold of the approach configurations that was produced by VAEFM, and figure 21 shows the color scheme for each neuron and the number of approaches that were mapped to each¹. The graphical illustration of the manifold shows that there are clusters present in the data; some of which were densely populated.

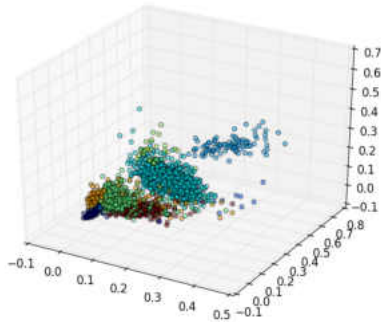


Figure 20: The manifold of the approach configurations which are color coded based on their neuron mappings.

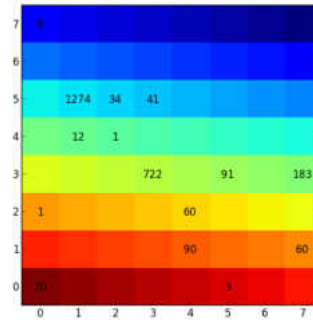


Figure 21: The color map for the neurons which shows the number of flights that were mapped to each.

Further evaluations revealed that, flights that are mapped to adjacent neurons had similar approach configurations and their levels of dissimilarity increased based on their distance in the grid. Consequently, factors that distinguish one cluster of flights from another is often a very subtle change in a few flight parameters, unless they were mapped further away from each other. Each neuron will be analyzed independently in the subsequent sections and the results will be displayed graphically with accompanying tabular information on

¹The color scheme for the neuron mapping is not related to the color-coding for the validation criteria

Table 4: The inter-cluster validation criteria for detecting unsafe practices.

Name	Definition	Legend		
		Red	Yellow	Green
Rapid VSI	Vertical acceleration (VSI), greater than 1000 feet per minute (fpm).	≤ -1000	≤ -900	> -900
High/Low IAS	Airspeed greater than 61 knots (or less than 55 knots for low airspeed).	> 75 or < 50	> 65 or < 55	$\sim 61 \pm 5$
High/Low Altitude	Derived radio altitude above 200 feet AGL (or less than 120 feet for low altitudes).	≥ 230 or ≤ 120	> 200 or ≤ 140	≤ 200 & > 140
Excessive Roll	10+ degree change in roll attitude while on final.	$\geq 10^\circ$	n/a	$< 10^\circ$
Excessive Pitch	10+ degree change in pitch while on final.	$\geq 10^\circ$	n/a	$< 10^\circ$

the range of parameter values.

2.3.3 Cluster Evaluation

This section automates the manual validation efforts by using UND’s SOP criteria to identify the frequency of violations that occur in each cluster. This seeks to demonstrate the forensic capability of the VAEFM algorithm for identifying correlated flight parameters that contribute to atypical descent patterns. Table 4 shows the criteria, which are not exclusive means of validation. However, they serve as guidelines for comparing the results in each cluster, and will be used to color code line graphs to show excessively high/low values in the approach configuration.

Weather conditions, such as precipitation and wind velocity, are contributory factors for unstabilized approaches. Therefore, weather data will be obtained to identify the presence of strong tailwinds and/or crosswinds.

The National Climatic Data Center (NCDC) provides information on weather

conditions across the USA by obtaining data from Automated Surface Observing Systems (ASOS) [100]. The system provides weather data from the automated sensors, at participating airports, and publishes them in 1-minute, 5-minute, and hourly observations. The dataset contains reporting on wind speed and direction, visibility, runway visual range, obstructions and various weather phenomena. It also includes sky conditions and cloud coverage, temperature, dew point, and the altimeter setting (i.e. pressure) [100, 101].

Weather data is reported using the METAR format, which collectively describes the above conditions for a given airport. This research obtained the 1-minute ASOS data and calculated the wind components (see equation 9), for GFK 35R to identify correlations in tailwinds or crosswinds for the selected approaches.

$$\begin{aligned} \text{Angle} &= \text{wind}_{\text{direction}} - \text{runway}_{\text{direction}} \cdot \pi/180 \\ \text{Left/Right Crosswind} &= \sin(\text{Angle}) \cdot \text{wind}_{\text{speed}} \\ \text{Tailwind/Headwind} &= \cos(\text{Angle}) \cdot \text{wind}_{\text{speed}} \end{aligned} \quad (9)$$

Cluster One

Cluster one comprised of 70 approaches which were excessively high and fast. There were approximately 36% with airspeeds exceeding 65 knots; the maximum detected was 95 knots. 33 % of the approaches were high (i.e. above 200 feet AGL); and 17% had rapid decelerations of 1000 fpm or more. Approximately 10% of these flights aborted their approach, possibly due to conditions that would result in it being unstabilized. These high, fast and rapid decelerations comprised over 87% of this cluster. Further analysis of the wind velocity shows 15% of these approaches had left or right crosswinds greater than 5 knots, which were contributory factors in the variation in roll during the descent.

Figure 22 shows an aerial view of the aircraft trajectories and figure 23 shows changes in flight parameters during their respective descent.



Figure 22: Cluster 1 - Aerial view of the trajectories for the excessively high and fast approaches; 10% of these were aborted.

The clustered results were further analyzed using the inter-cluster validation criteria, and figure 24 shows the pie chart of the percentage of occurrences; table 5 contains summary statistics for this cluster.

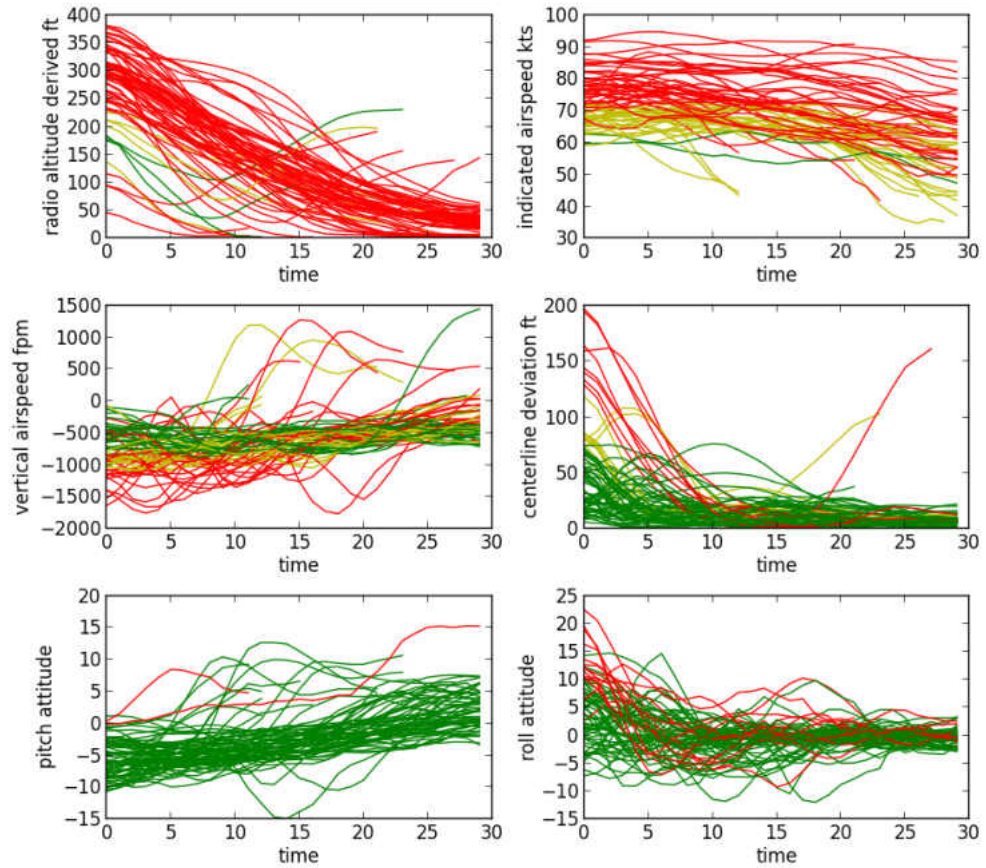


Figure 23: Cluster 1 - Line graph showing the changes in altitude, indicated airspeed, vertical speed, position (left/right of centerline), roll and pitch. This cluster contained flights with excessively high altitudes, airspeeds and vertical speeds while on final approach. Approximately 10% of these flights performed a go-around maneuver.

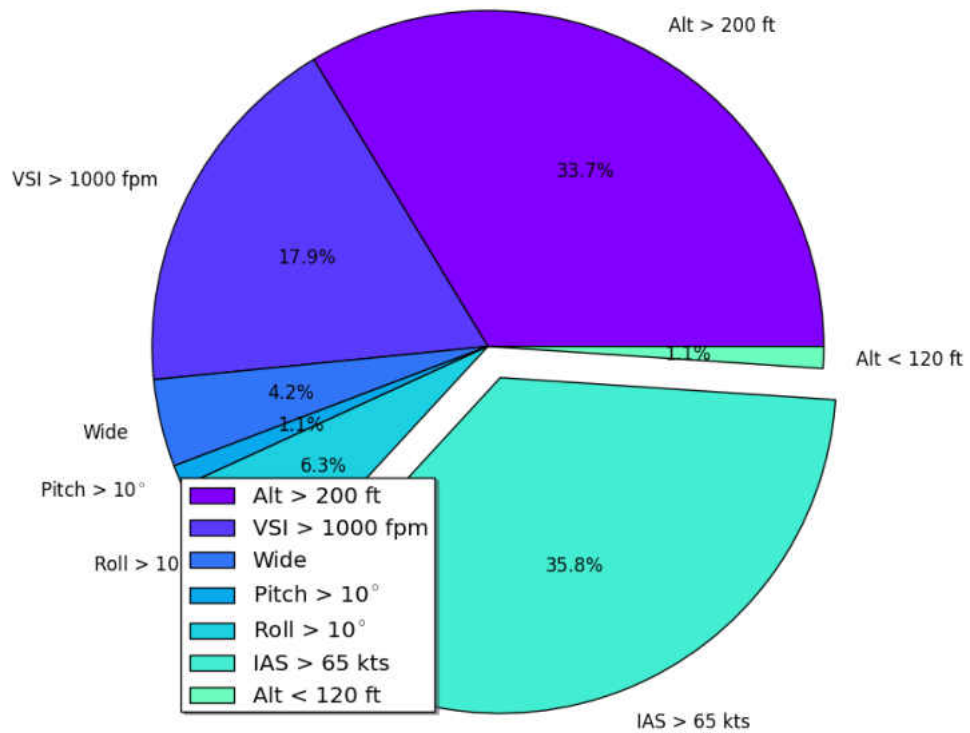


Figure 24: Cluster 1 - Pie Chart showing the validation metrics and their respective percentages of occurrence. All 70 flights had unsafe events which are contributory factors for unstabilized approaches.

Table 5: Cluster 1 - Summary statistics for each flight parameter. The highest altitude, airspeed and vertical speed during the descent was 401 ft, 95 kts and -2035 fpm respectively.

Parameter	Max	Min	Mean	Standard Deviation
Altitude	401.0	1.0	136.02	96.86
Airspeed	95.40	30.99	69.95	9.78
Vertical Speed	1446.84	-2035.17	-591.27	409.77
Pitch	15.96	-15.84	-2.15	4.31
Roll	26.03	-14.77	0.42	4.58
Eng RPM	2561.60	0.00	1284.47	457.70
Vertical Acceleration	0.59	-0.65	0.009	0.096
Position	222.26	0.31	21.07	27.25

Cluster Two

Cluster two contained 60 flights, 69% of which had high airspeed with the highest being 75 knots, which was triggered by one flight (depicted in red in figure 26). The remaining IAS were on average 7 knots faster than UND's standard operating procedures (SOPs) for IAS on final approach. However, the approaches in this cluster were considered stable as there were no unusual changes in each aircraft's descent profile.



Figure 25: Cluster 2 - Aerial view of approaches that were on average 7 knots faster than UND's SOP; however they were stabilized.

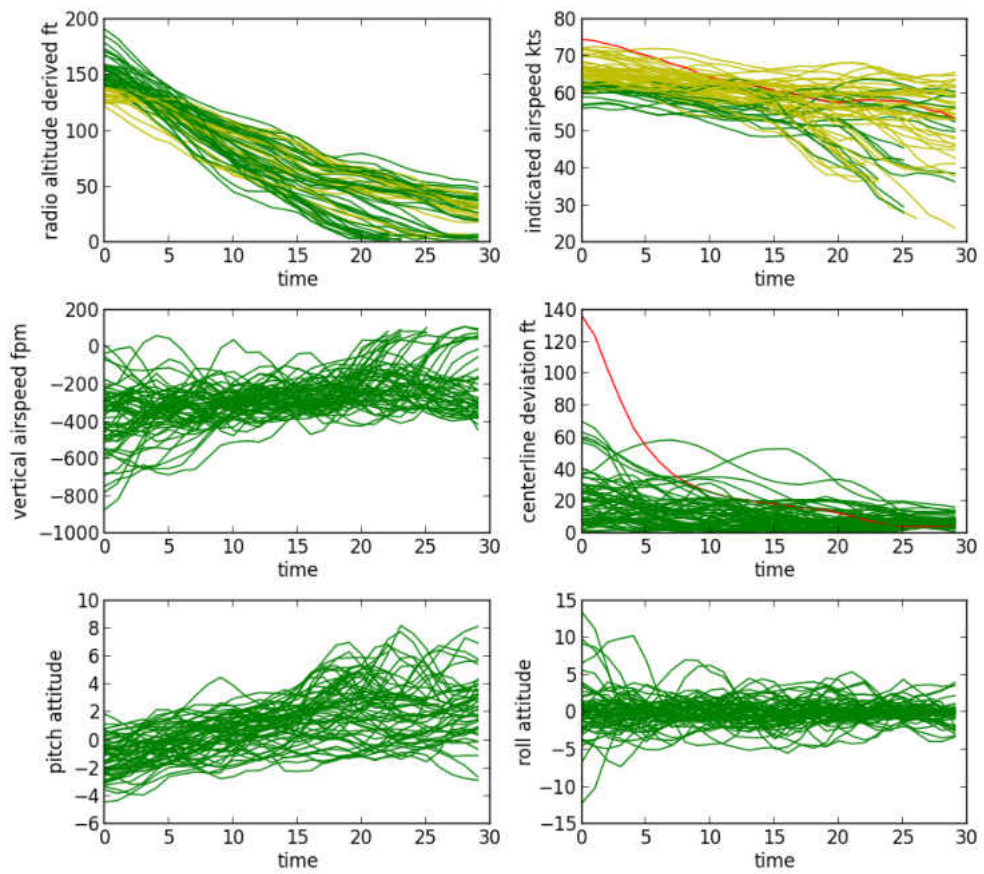


Figure 26: Cluster 2 - Line graph showing the changes in altitude, air-speed, vertical speed, position (left/right of centerline), pitch and roll for the stabilized approaches.

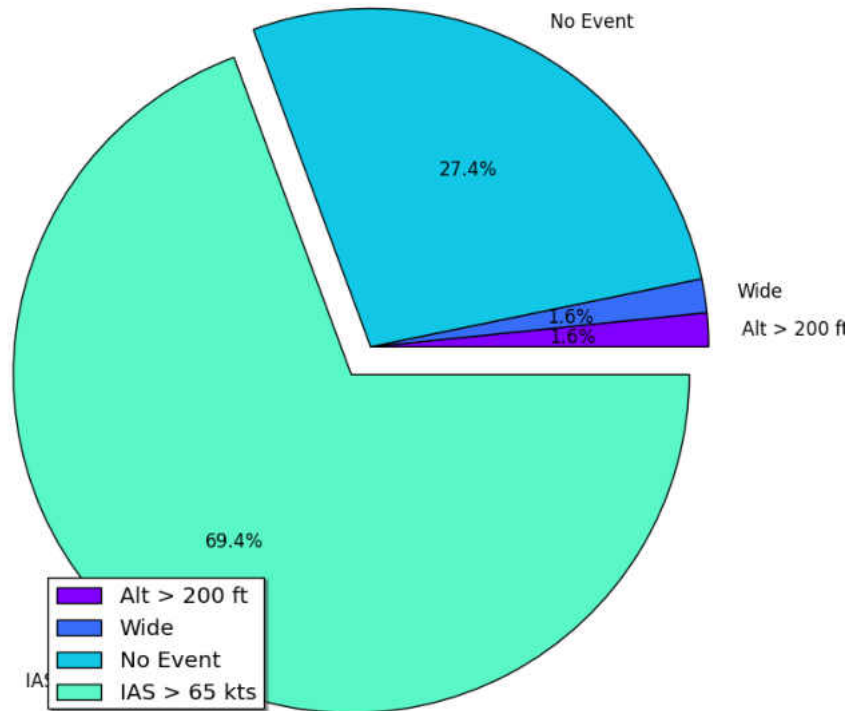


Figure 27: Cluster 2 - Pie chart showing the evaluated approach configurations using the inter-cluster validation criteria. There were 69% of flights with airspeeds greater than 65 knots (the maximum airspeed was 75 knots); 27% did not trigger any events and the remaining 3% were slightly high/wide approaches.

Table 6: Cluster 2 - Summary statistics showing the respective range of parameter values. This cluster did not contain excessively high values or unusual changes in the descent profile.

Parameter	Max	Min	Mean	Standard Deviation
Altitude	206.0	1.0	77.83	43.13
Airspeed	75.38	22.10	59.00	7.20
Vertical Speed	154.12	-918.47	-279.78	151.81
Pitch	10.34	-5.16	0.92	2.25
Roll	17.64	-14.62	0.07	2.68
Eng RPM	2348.10	668.50	1561.36	278.10
Vertical Acceleration	0.49	-0.40	0.0008	0.082
Position	154.92	0.31	12.96	12.55

Cluster Three

Cluster three comprised of 90 flights, 55% of which had slightly fast airspeeds. As with cluster 2, they were on average 7 knots faster than UND's SOP. The descent rate, roll and pitch did not indicate any unexpected changes in the approach configuration (see figure 29).

Both clusters 2 and 3 did not contain major deviations in their flight trajectories (see figure 25 and 28); The most distinguishing factors between them are very subtle (see tables 6 and 7). All approaches in cluster 3 were initiated below 200 feet, their respective sink rates did not surpass 750 feet per minute, and there was less variability in roll and pitch during these descent. However, both clusters 2 and 3 are considered stabilized approaches, with cluster 2 being slightly faster, but not excessive.



Figure 28: Cluster 3 - Aerial view of the flight trajectories.

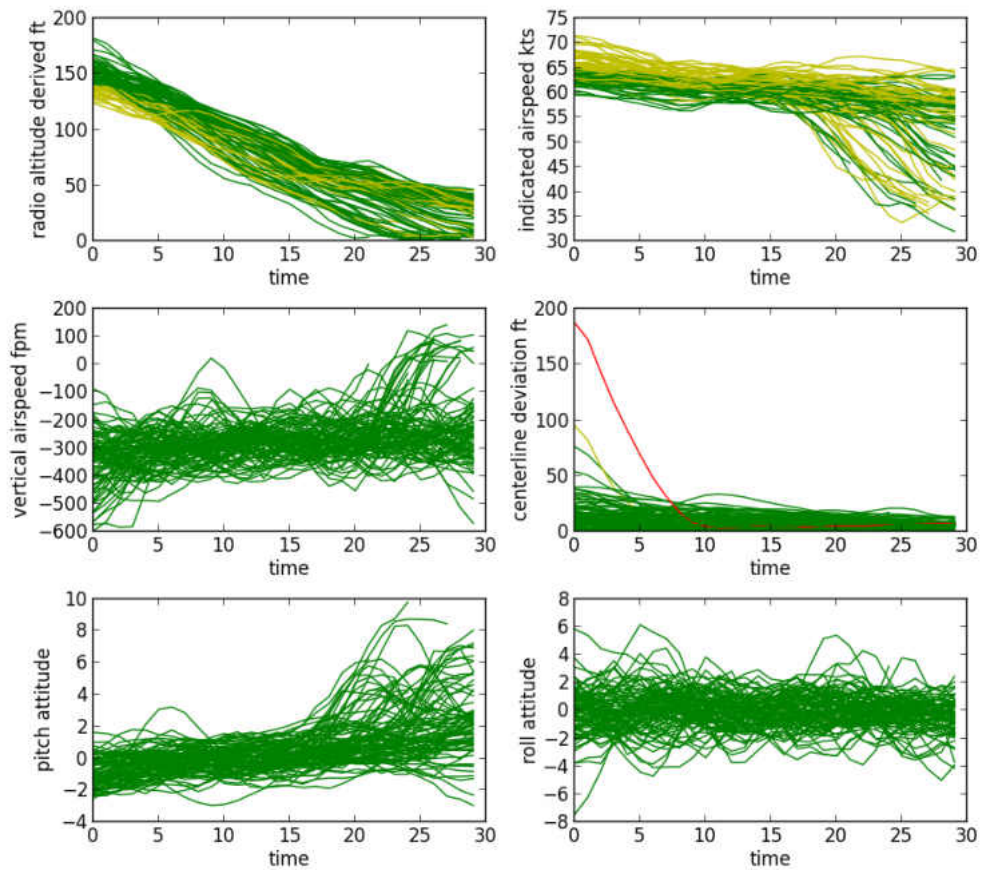


Figure 29: Cluster 3 - Line graph showing the altitude, airspeed, vertical speed, position, roll and pitch while on final. These approaches were approximately 7 knots faster than UND's SOP, however there were no unusual deviations in the flight trajectories during their respective descent.

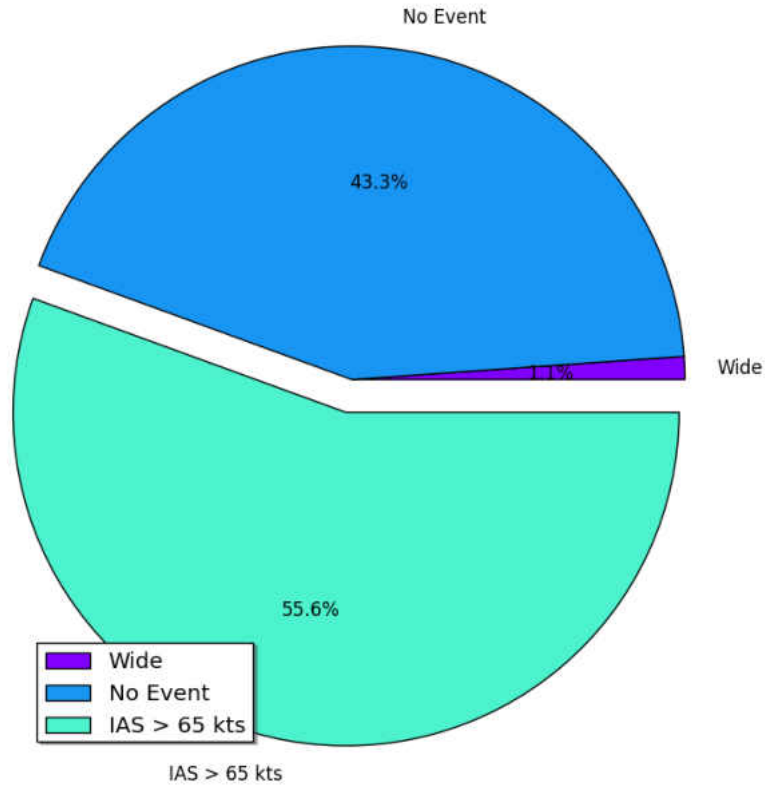


Figure 30: Cluster 3 - Pie chart showing 55% of flights with high airspeeds (which did not surpass 75 knots) and 1% of flights which initiated their approach slightly wide.

Table 7: Cluster 3 - Summary statistics for the stable approaches, although slightly fast, did not contain unusual deviations in their descent profiles.

Parameter	Max	Min	Mean	Standard Deviation
Altitude	192.0	1.0	81.21	42.64
Airspeed	72.63	31.41	60.19	5.33
Vertical Speed	156.78	-745.51	-276.60	107.24
Pitch	9.93	-5.43	0.49	1.78
Roll	10.70	-7.93	-0.038	1.89
Eng RPM	2291.30	654.40	1611.65	234.90
Vertical Acceleration	0.32	-0.38	0.00072	0.064
Position	219.13	0.31	9.50	10.56

Cluster Four

Cluster Four comprised of 60 flights, 47% were fast, 36% high and 10 % had rapid descent rates. This cluster also included flights that were not fully aligned with the runway when they initiated their descent, consequently they were identified as wide.

2% of these flights had right crosswinds greater than 10 knots. However, there was no relationship between crosswinds and wide approaches in this cluster.

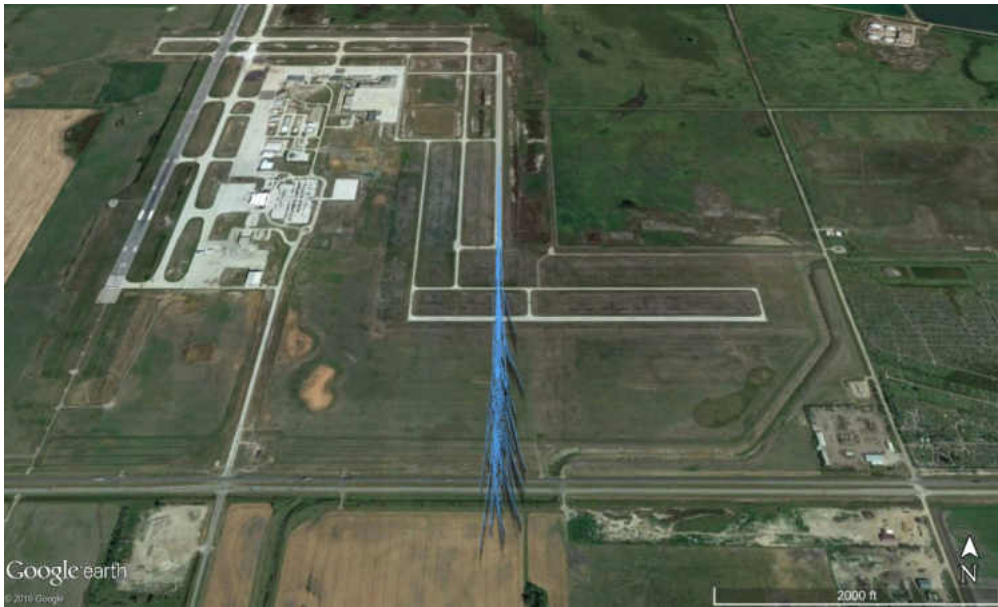


Figure 31: Cluster 4 - Aerial view of the flight tracks showing 60 approaches, some of which were not fully aligned with the runway when they initiated their descent.

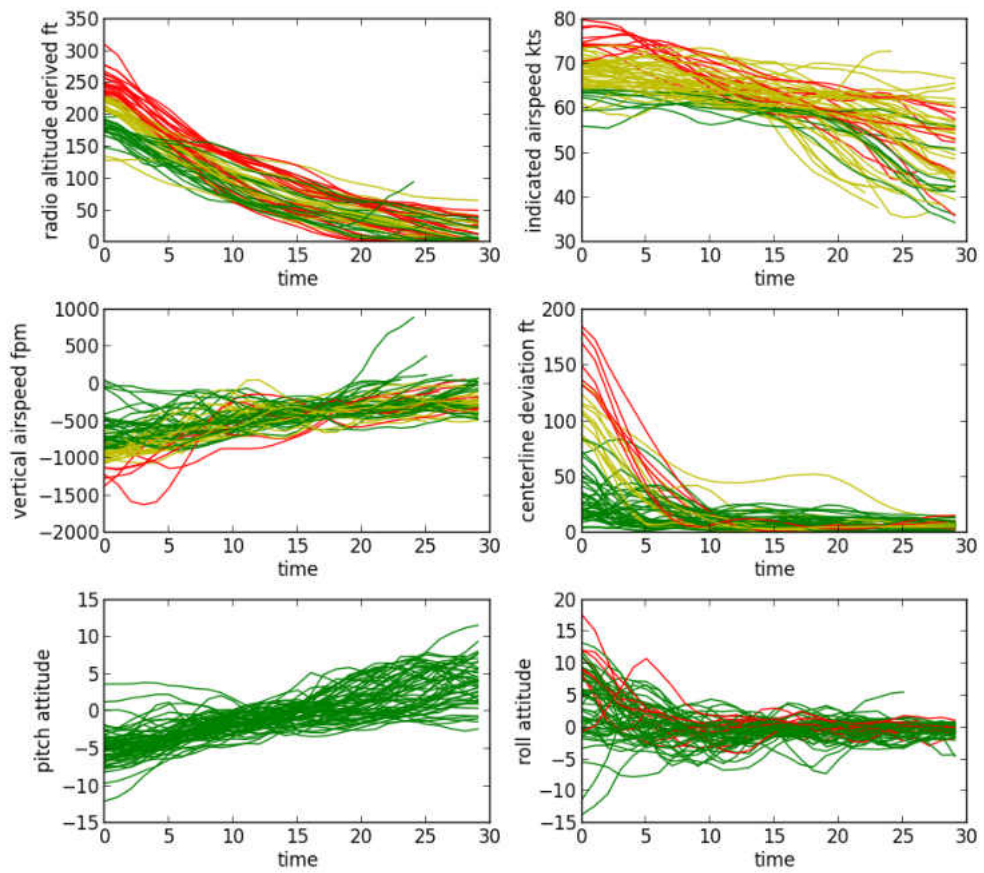


Figure 32: Cluster 4 - Line graph showing changes in altitude, airspeed, vertical speed, position, pitch and roll.

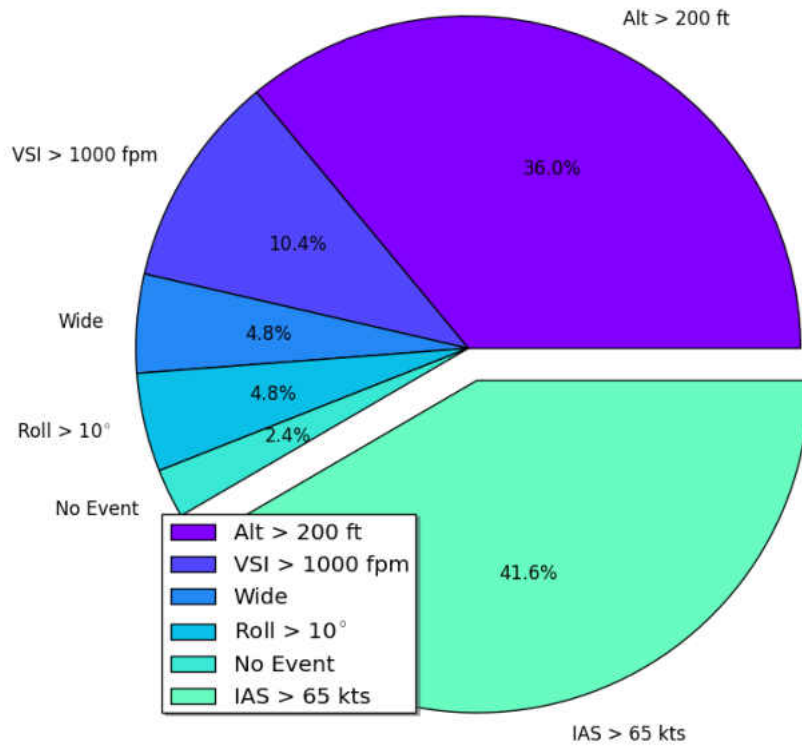


Figure 33: Cluster 4 - Pie chart showing the frequency of occurrence for each validation criteria.

Table 8: Cluster 4 - Summary statistics showing variations in flight parameters while on final

Parameter	Max	Min	Mean	Standard Deviation
Altitude	347.0	1.0	93.31	67.36
Airspeed	81.41	32.25	61.92	7.69
Vertical Speed	1005.21	-1738.48	-454.04	278.02
Pitch	11.89	-13.38	-0.59	3.49
Roll	19.69	-16.57	0.37	3.53
Eng RPM	2449.20	683.50	1322.51	327.58
Vertical Acceleration	0.86	-0.47	0.010	0.087
Position	211.66	0.31	17.44	25.95

Cluster Five

Cluster five contained 12 flights and figure 34 shows that each aircraft's line of descent had oscillations; this is a cluster of unstabilized approaches. All approaches were excessively high (i.e. between 300 and 450 feet AGL). They were all fast, with the exception of one flight. Their descent rates were rapid, possibly due to their high altitude. The remaining 15% of these approaches were misaligned or still attempting to position the aircraft, just 30 seconds before touchdown (see figure 35).



Figure 34: Cluster 5 - Aerial view of the flight trajectories.

The wind component analysis indicated that there was no tailwind greater than 5 knots, however 8% of flights had right crosswinds greater than 10 knots. Further analysis revealed that the crosswind effect was not a strong indicator for the unstable approaches.

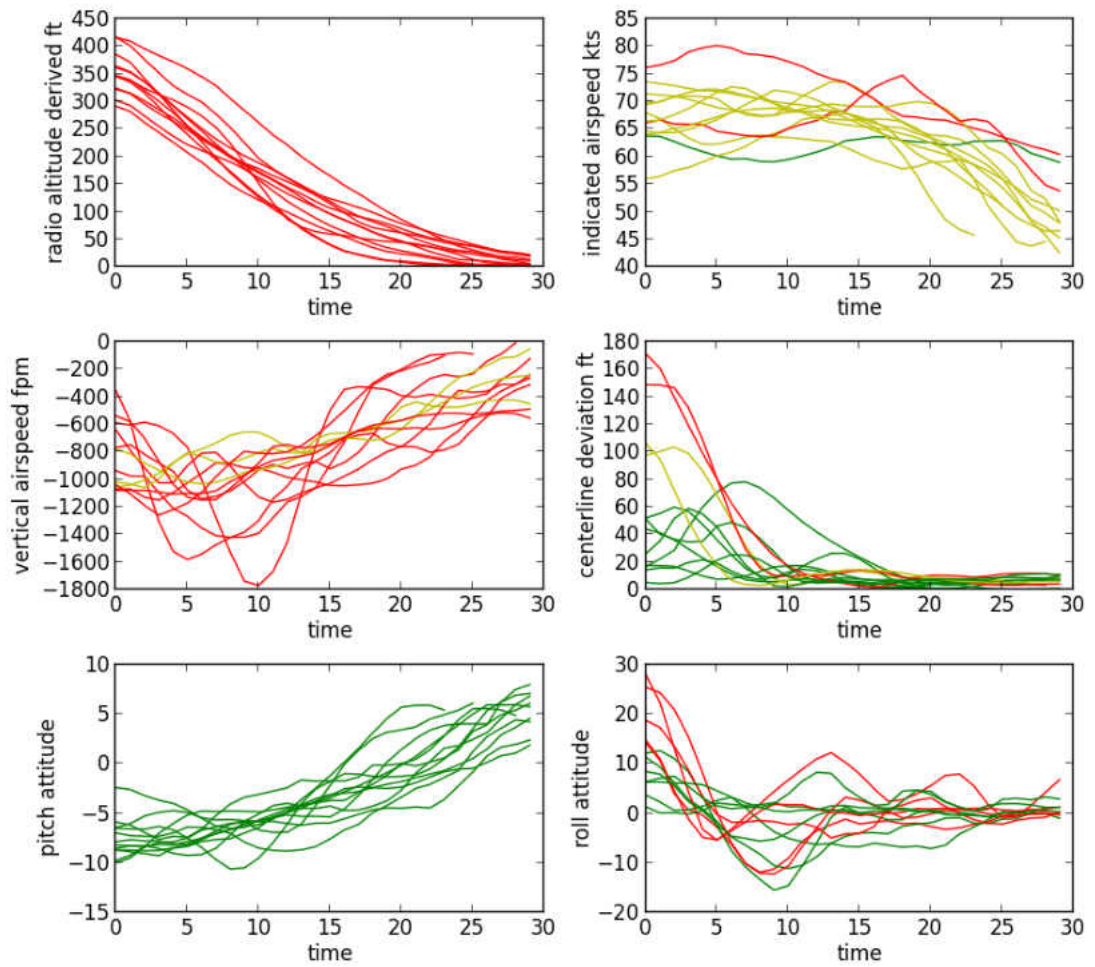


Figure 35: Cluster 5 - High altitude and airspeed, excessive vertical speed and bank angle as contributing factors for these unstable approaches.

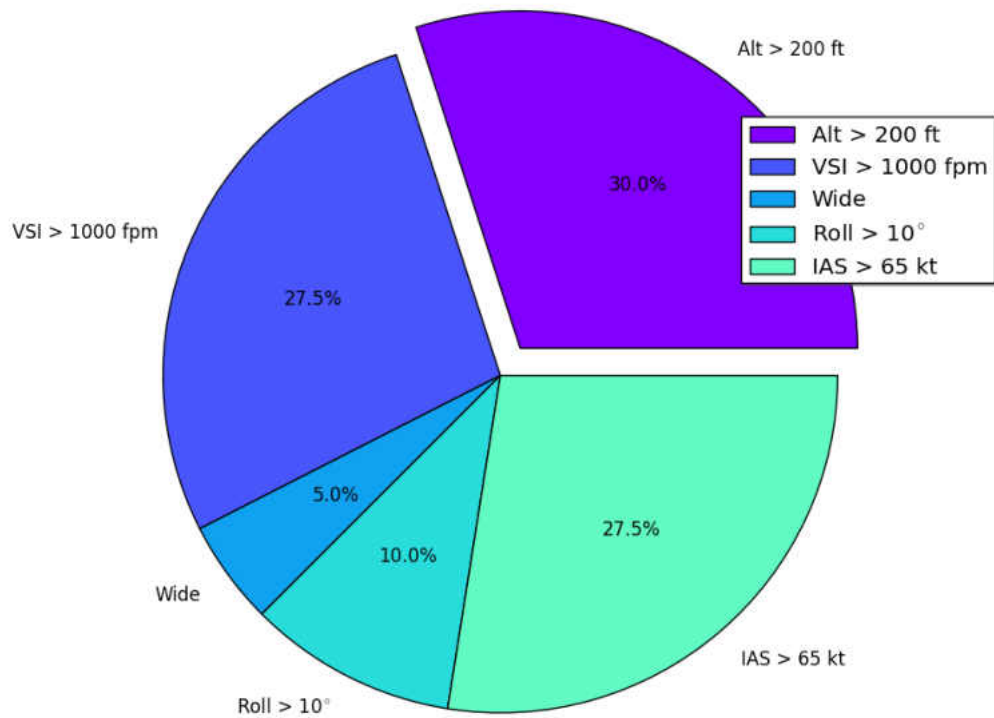


Figure 36: Cluster 5 - Pie chart showing the percentage of events that were detected by the validation criteria.

Table 9: Cluster 5 - Summary statistics for the unstable approaches showing excessive Altitude, VSI, IAS and Roll as contributory factors.

Parameter	Max	Min	Mean	Standard Deviation
Altitude	434.0	1.0	144.33	118.57
Airspeed	81.65	40.52	64.46	7.15
Vertical Speed	111.56	-1892.77	-761.38	351.77
Pitch	10.12	-11.54	-3.25	4.55
Roll	29.16	-17.75	0.91	6.88
Eng RPM	2384.40	763.40	1009.74	145.62
Vertical Acceleration	0.30	-0.30	0.0075	0.089
Position	192.95	0.31	21.29	31.12

Cluster Six

Cluster Six comprised of 91 flights, which were also high and fast. However, this cluster also contained a higher frequency of flights with unsafe low-level turning maneuvers and wide approaches.

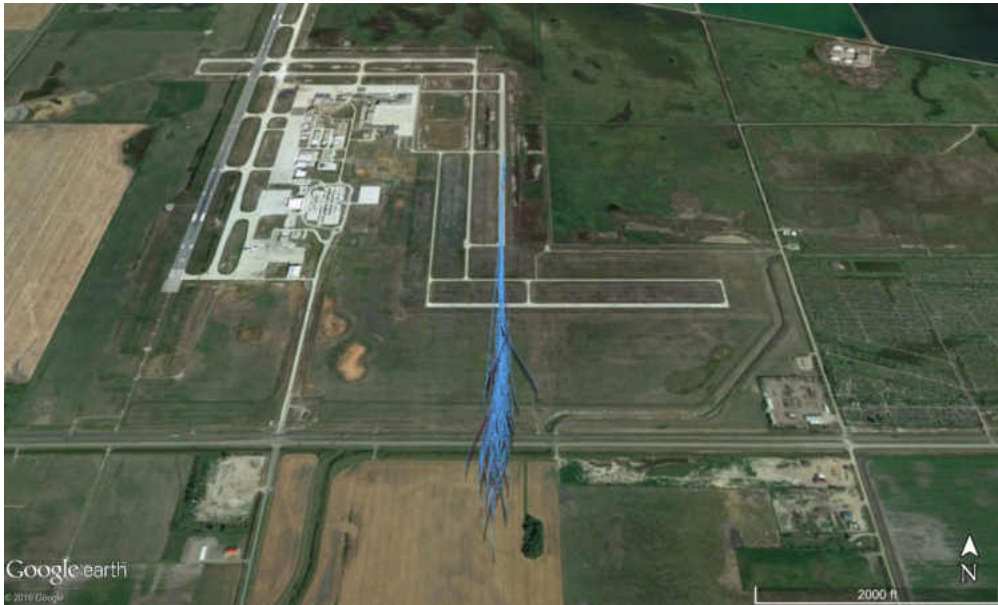


Figure 37: Cluster 6 - Aerial view of the flight trajectories

Further analysis revealed that 5% of these approaches had a tailwind greater than 5 knots, and 7% with left or right crosswinds that were greater than 10 knots. The wind component analysis was performed again only using the flights with unsafe low-level turns and the results indicated that wind was a direct factor for approximately 20% of the excessive roll events.

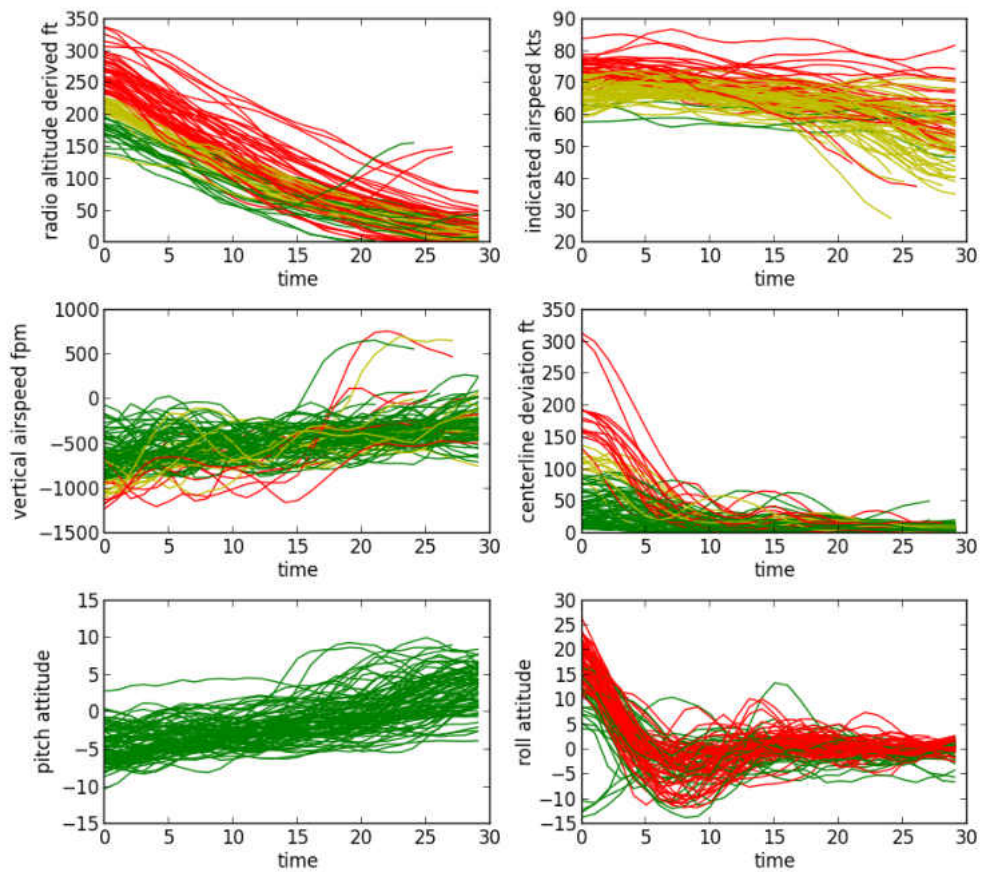


Figure 38: Cluster 6 - Changes in altitude, airspeed, vertical speed, position, pitch and roll while on final. These flights were not only high and fast but also had huge changes in roll during their descent.

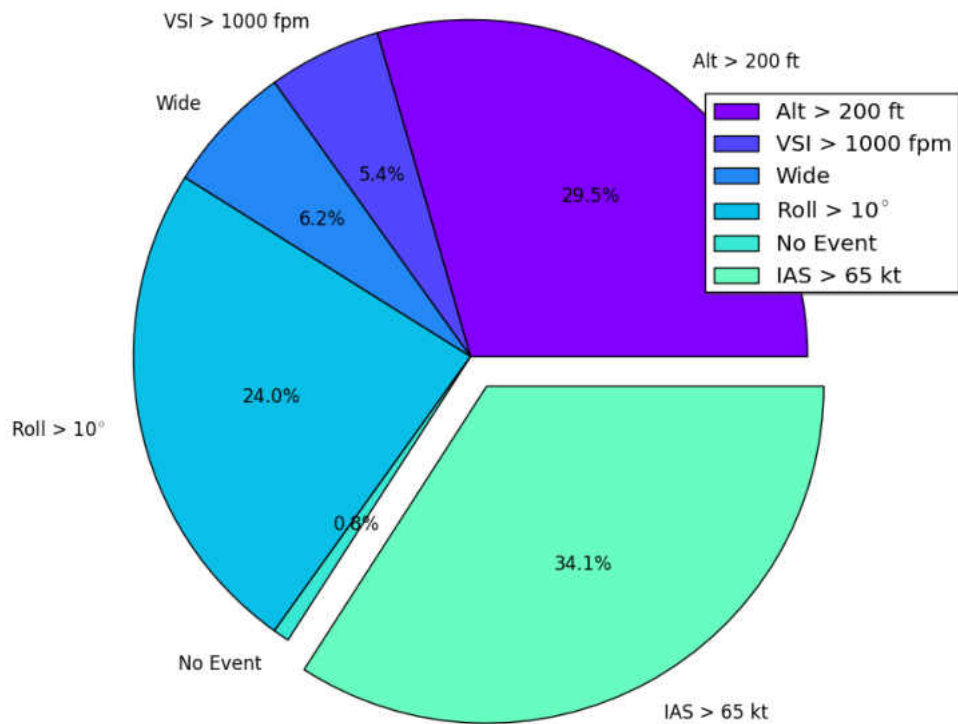


Figure 39: Cluster 6 - Pie chart showing a higher occurrence of excessive roll events in approaches that were predominantly fast.

Table 10: Cluster 6 - Summary statistics showing high and fast approaches some of which had a maximum roll exceeding 30 degrees during the descent.

Parameter	Max	Min	Mean	Standard Deviation
Altitude	353.0	1.0	110.75	75.98
Airspeed	88.68	27.09	65.41	7.53
Vertical Speed	836.07	-1457.49	-477.23	260.85
Pitch	11.46	-11.09	-1.39	3.50
Roll	30.51	-17.36	1.28	6.57
Eng RPM	2486.30	668.20	1363.65	306.72
Vertical Acceleration	0.72	-0.33	0.017	0.086
Position	340.70	0.31	25.13	35.38

Cluster Seven

Cluster seven contained 183 approaches which were shorter than previous clusters; they were either very high and fast, or very low and fast. This cluster had a higher frequency of aborted approaches which indicates that pilots may have performed a go around due to conditions that would result in unstabilized approaches or other unsafe events.

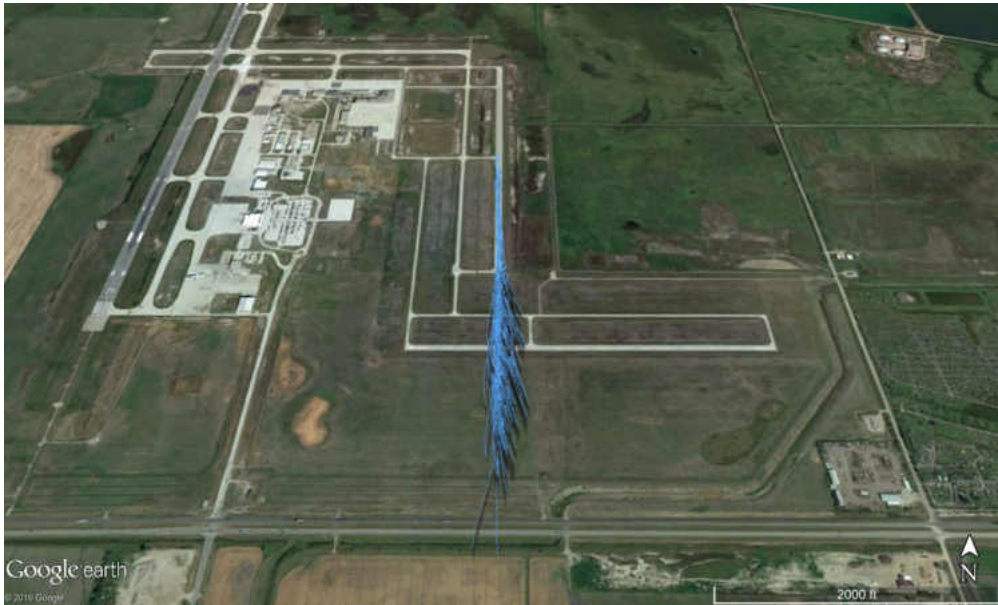


Figure 40: Cluster 7 - Aerial view of the approach trajectories which contains a higher frequency of aborted approaches.

Approximately 4% of the approaches had tailwinds greater than 5 knots and 12% had crosswinds greater than 10 knots. There was a direct correlation between wide approaches and strong crosswinds; some of which exceeded 20 knots.

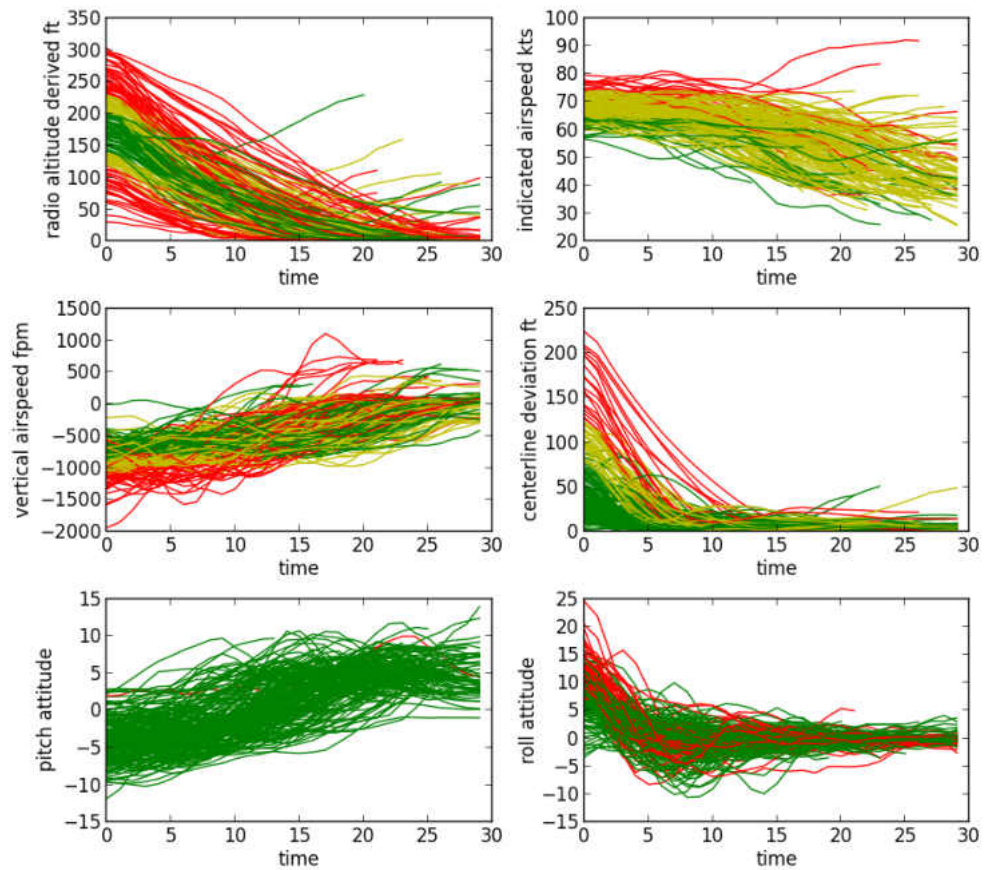


Figure 41: Cluster 7 - Line graph showing changes in altitude, airspeed, vertical speed, position, pitch and roll. This cluster had multiple aborted approaches, possibly due to awareness of factors that would result in an unstabilized approach.

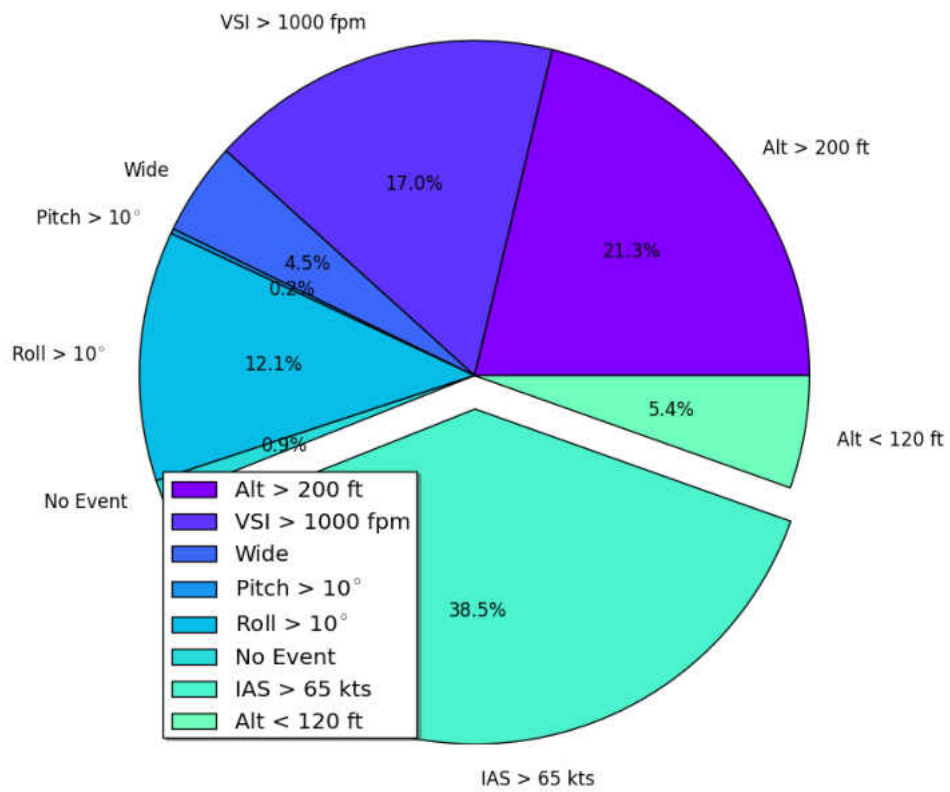


Figure 42: Cluster 7 - Pie chart showing over 99 % of flights in this cluster triggered multiple unsafe events from the validation criteria.

Table 11: Cluster 7 - Summary statistics showing the range of flight parameters. This cluster had the fastest VSI at -2060.03 feet per minute.

Parameter	Max	Min	Mean	Standard Deviation
Altitude	326.0	1.0	74.27	69.81
Airspeed	94.32	22.51	60.28	9.87
Vertical Speed	1256.24	-2060.03	-502.98	387.02
Pitch	15.51	-13.33	0.40	4.497
Roll	28.42	-16.69	1.10	4.32
Eng RPM	2530.50	655.70	1016.02	332.08
Vertical Acceleration	0.96	-0.38	0.018	0.096
Position	250.92	0.31	17.86	28.52

Cluster Eight

Cluster eight comprised of 722 approaches, 70% of which had predominantly high airspeeds, 7% high altitudes and 20% did not trigger any unsafe events from the validation metrics. There were 2% and 3% of approaches with left and right crosswind exceeding 10 knots, and 1% with tailwinds greater than 5 knots. Consequently, the wind velocity had a minuscule effect in this cluster of fast approaches.



Figure 43: Cluster 8 - Aerial view of the flight trajectories.

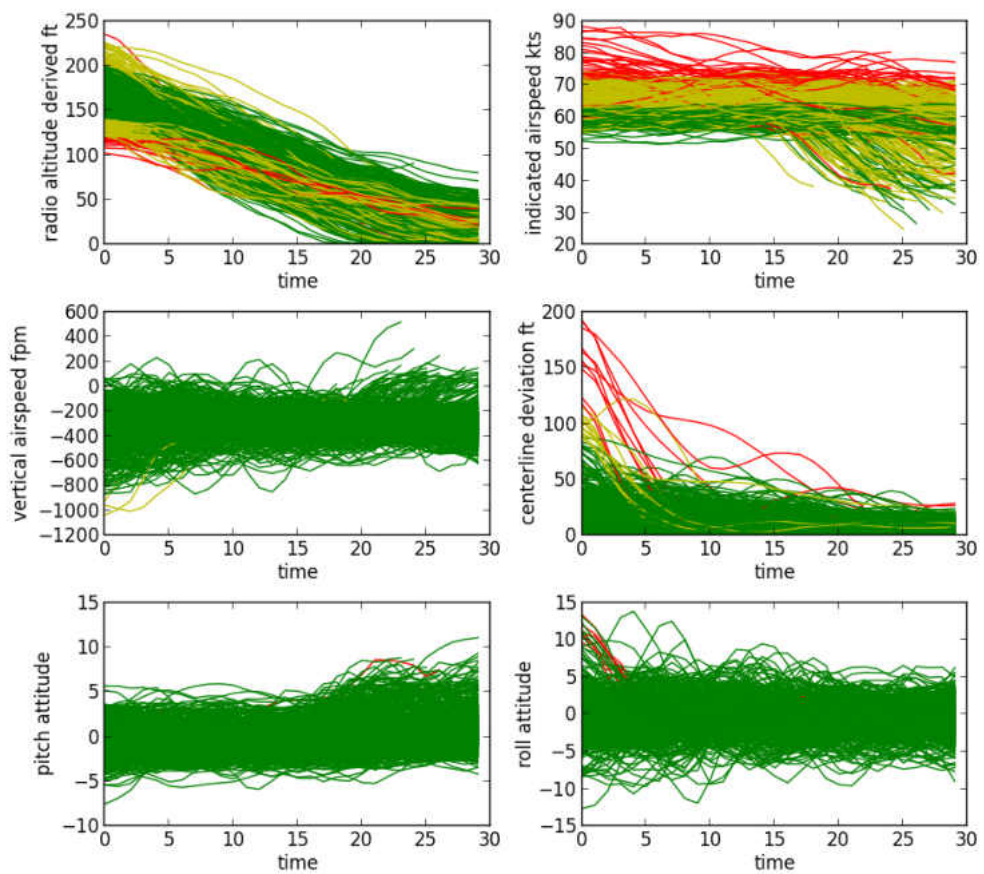


Figure 44: Cluster 8 - Line graph showing changes in the altitude, air-speed, vertical speed, position, pitch and roll for each approach configuration.

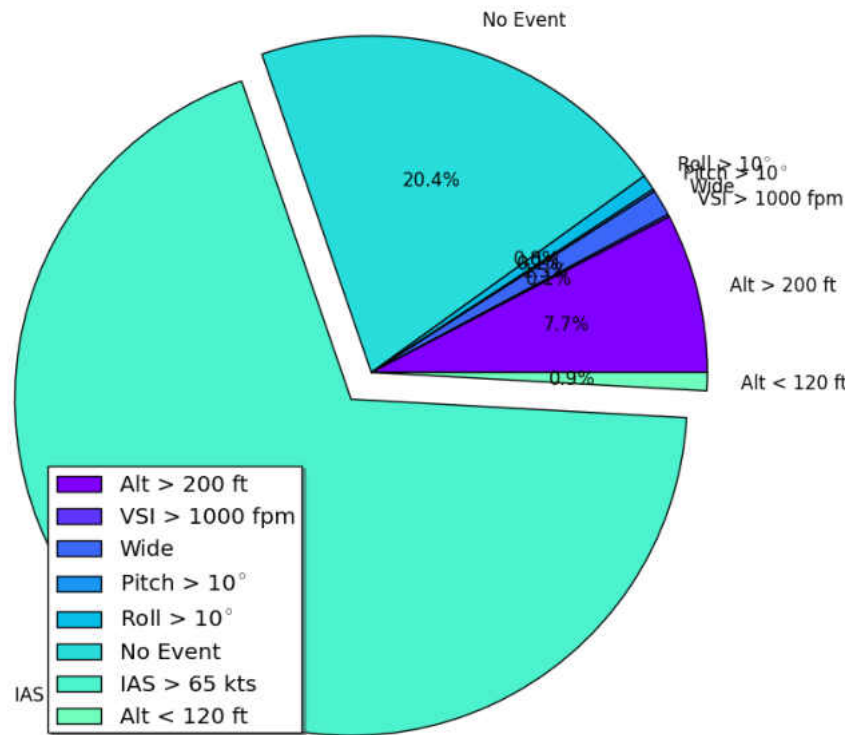


Figure 45: Cluster 8 - Pie chart showing airspeeds as high as 90 knots as unsafe events in this cluster.

Table 12: Cluster 8 - Summary statistics for the approach configurations; which shows a wider range of values for gravitational forces on the aircraft, i.e. vertical acceleration, than other clusters.

Parameter	Max	Min	Mean	Standard Deviation
Altitude	249.0	1.0	92.81	45.01
Airspeed	90.29	22.68	62.95	5.29
Vertical Speed	567.40	-1122.90	-288.14	136.49
Pitch	11.78	-8.36	-0.004	1.901
Roll	17.60	-14.94	-0.18	2.74
Eng RPM	2502.80	618.00	1704.18	266.86
Vertical Acceleration	1.20	-0.44	-0.0003	0.08
Position	224.73	0.31	12.14	12.93

Cluster Nine

Cluster nine comprised of another group of unstable approaches. These 9 flights had excessive altitudes and airspeeds, and fluctuations in the rate of descent. Strong right crosswind components were evident in 40% of the approaches with 10+ degree changes in roll.



Figure 46: Cluster 9 - Aerial view of the flight tracks.

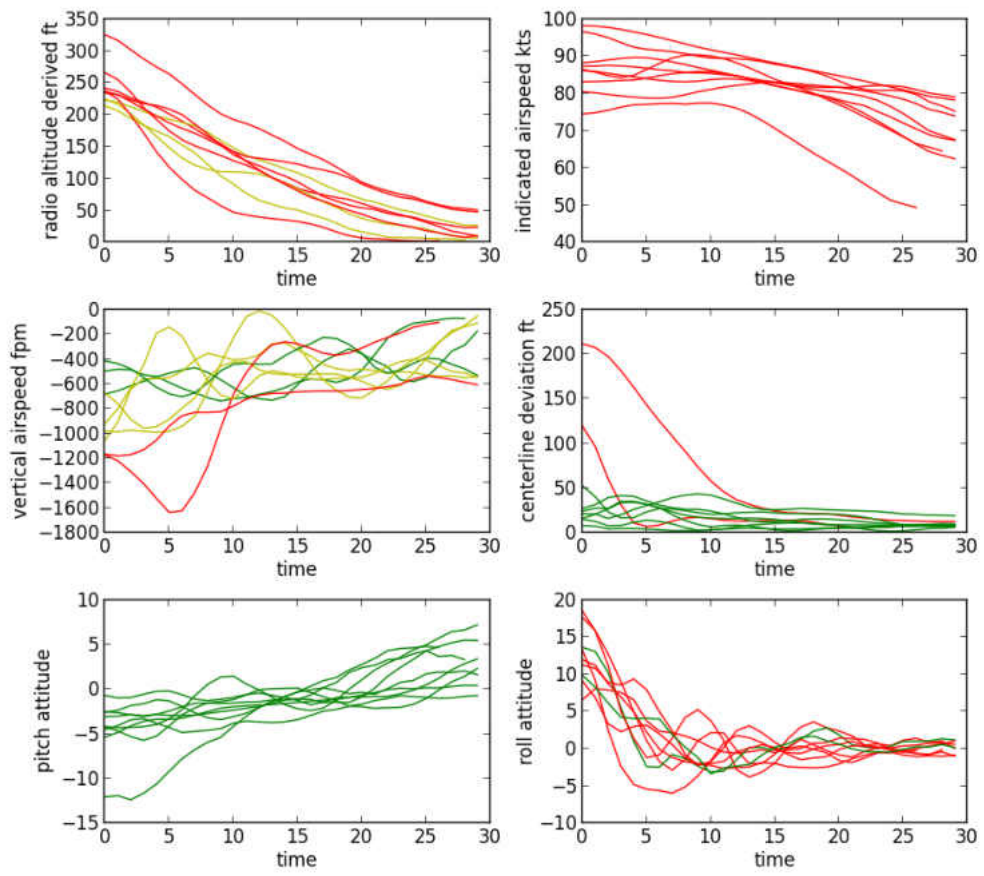


Figure 47: Cluster 9 - Line graph showing changes in the altitude, airspeed, vertical speed, position, pitch and roll for the unstable approaches.

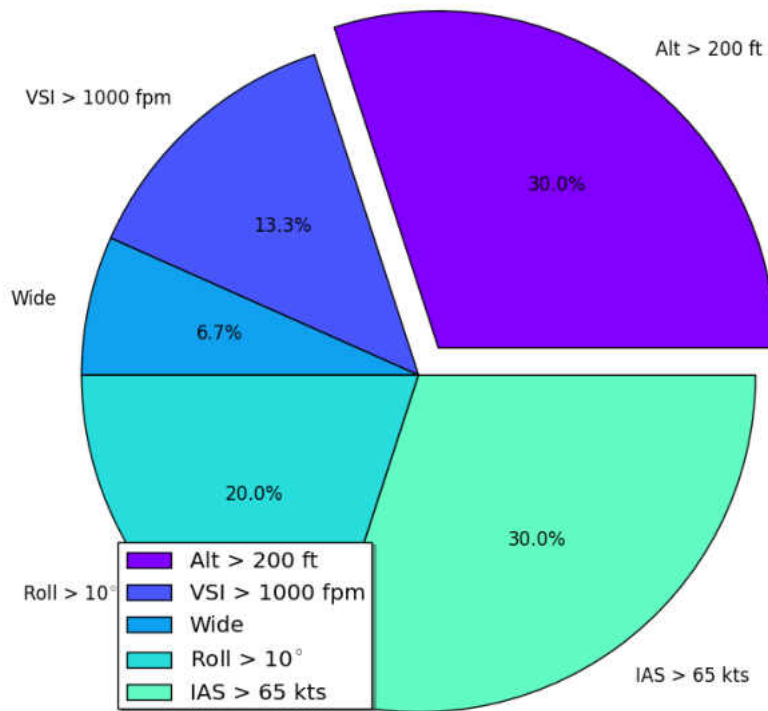


Figure 48: Cluster 9 - Pie chart showing five contributory events for the unstable approaches.

Table 13: Cluster 9 - Summary statistics of flight parameters which shows the highest mean, and maximum airspeed among all clusters.

Parameter	Max	Min	Mean	Standard Deviation
Altitude	341.0	1.0	111.90	78.45
Airspeed	98.32	48.40	81.54	8.21
Vertical Speed	26.93	-1808.61	-561.66	300.34
Pitch	7.43	-13.04	-1.20	3.10
Roll	21.80	-6.93	1.57	5.11
Eng RPM	2615.60	891.10	1931.86	439.17
Vertical Acceleration	0.21	-0.27	0.012	0.07
Position	214.78	0.31	22.77	32.76

Clusters Ten to Twelve

Clusters ten to twelve are neighboring neurons, consequently their results had many similarities and indistinct dissimilarities. Due to the level of cohesion between these three neurons, their results will be presented contiguously to highlight the subtle differences between them.

Cluster ten is the most densely populated neuron, and comprised of 1274 approaches; whereas clusters eleven and twelve contained 34 and 41 respectively. All three clusters had a considerable occurrence of high/fast approaches. However, the excessive roll events in cluster ten had strong right crosswinds with a maximum of 24 knots and tailwinds greater than 11 knots. These are contributory factors for the 10+ degree changes in roll shown in figure 52. Wind velocity were not causal factors in cluster 11, however 11% of the approaches in cluster 12 had tailwinds and crosswinds exceeding 5 and 10 knots respectively.

All three clusters had small changes in pitch as shown in figures 52, 53, and 54, similar engine RPM settings and rapid decelerations of 1100 fpm or greater. Cluster ten contained approaches with the widest distance from the runway centerline, and maximum altitude and airspeed of 347 feet and 86 knots. Cluster eleven represented higher altitudes with the maximum being 365 feet, minimum of 6 feet, and airspeeds up to 78 knots. Cluster twelve represented slightly lower altitudes and airspeed, 337 feet AGL, and 77 knots (see tables 14, 15 and 16).

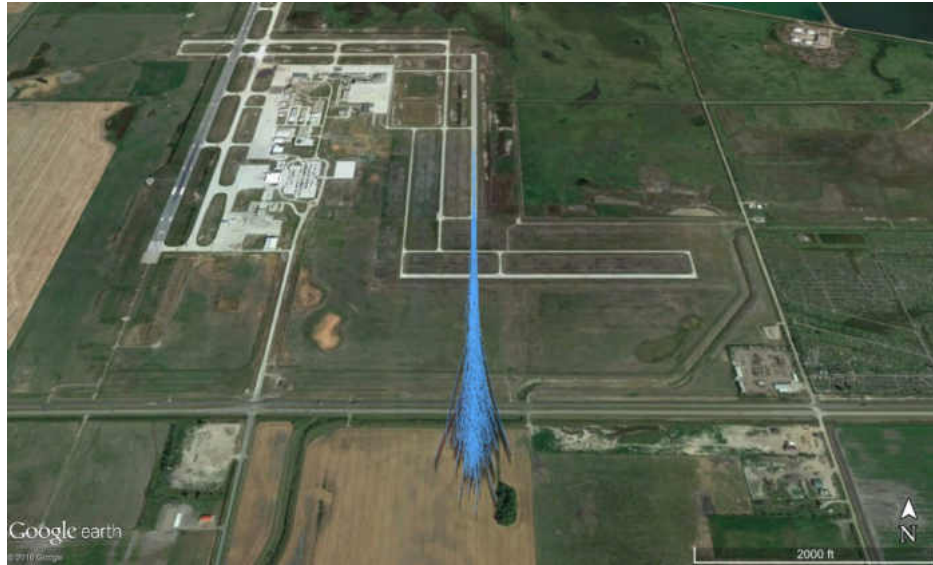


Figure 49: Cluster 10 - 1274 densely populated approaches.



Figure 50: Cluster 11 - 34 high/fast approaches.

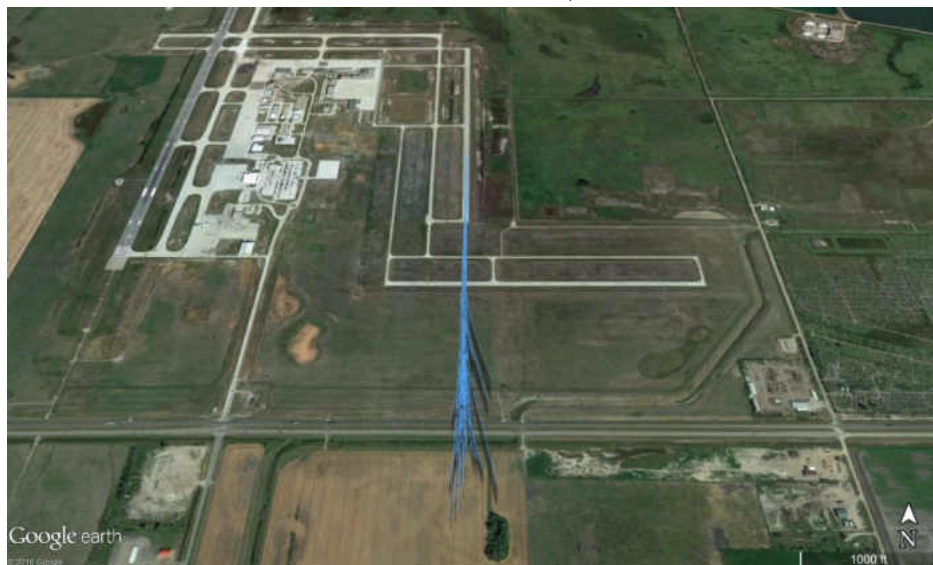


Figure 51: Cluster 12 - 41 high/fast approaches.

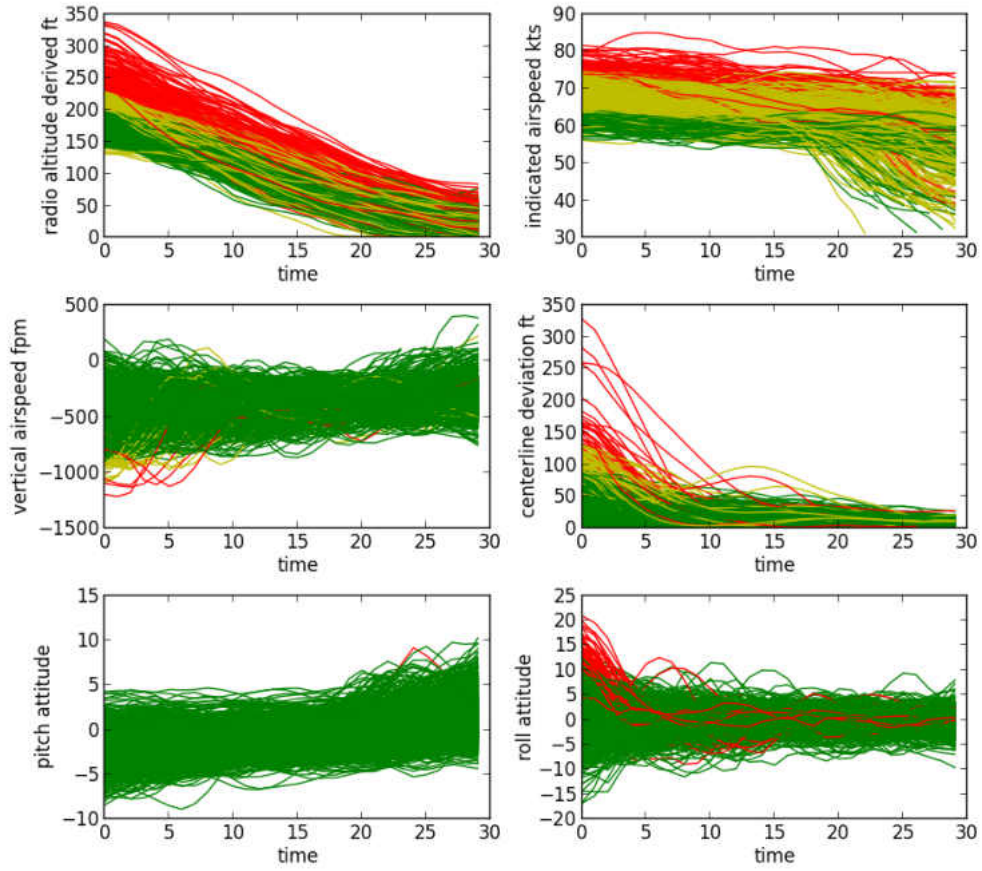


Figure 52: Cluster 10 - Line graph showing changes in altitude, airspeed, vertical speed, position, pitch and roll while on final.

Table 14: Cluster 10 - Summary statistics for the densely populated high/fast cluster which had 10+ degree changes in roll, and approaches that were high, fast, and wide with rapid descent profiles.

Parameter	Max	Min	Mean	Standard Deviation
Altitude	347.0	1.0	107.46	56.58
Airspeed	86.41	29.81	63.48	4.83
Vertical Speed	445.97	-1445.40	-366.34	145.14
Pitch	10.38	-10.03	-0.65	1.97
Roll	25.83	-23.79	-0.039	2.90
Eng RPM	2472.40	606.00	1574.02	204.12
Vertical Acceleration	0.72	-0.46	0.0016	0.072
Position	356.92	0.31	14.08	16.73

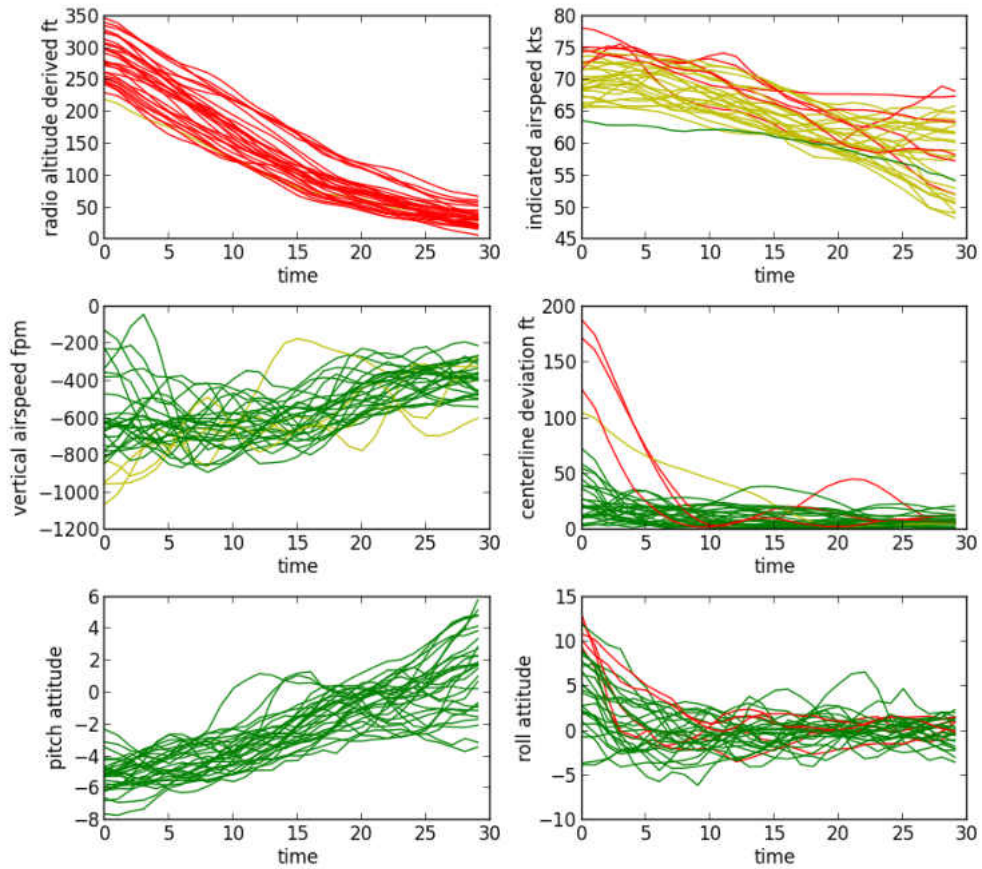


Figure 53: Cluster 11 - Line graph of the parameter values, while on final.

Table 15: Cluster 11 - Summary statistics showing the range of values for the high/fast/long approaches.

Parameter	Max	Min	Mean	Standard Deviation
Altitude	365.0	6.0	136.05	85.54
Airspeed	78.65	45.65	64.75	5.59
Vertical Speed	33.41	-1169.85	-550.02	176.26
Pitch	6.22	-8.34	-2.31	2.59
Roll	20.72	-9.33	0.54	3.19
Eng RPM	2416.60	856.00	1286.04	223.95
Vertical Acceleration	0.45	-0.36	0.0038	0.072
Position	211.05	0.31	14.71	20.76

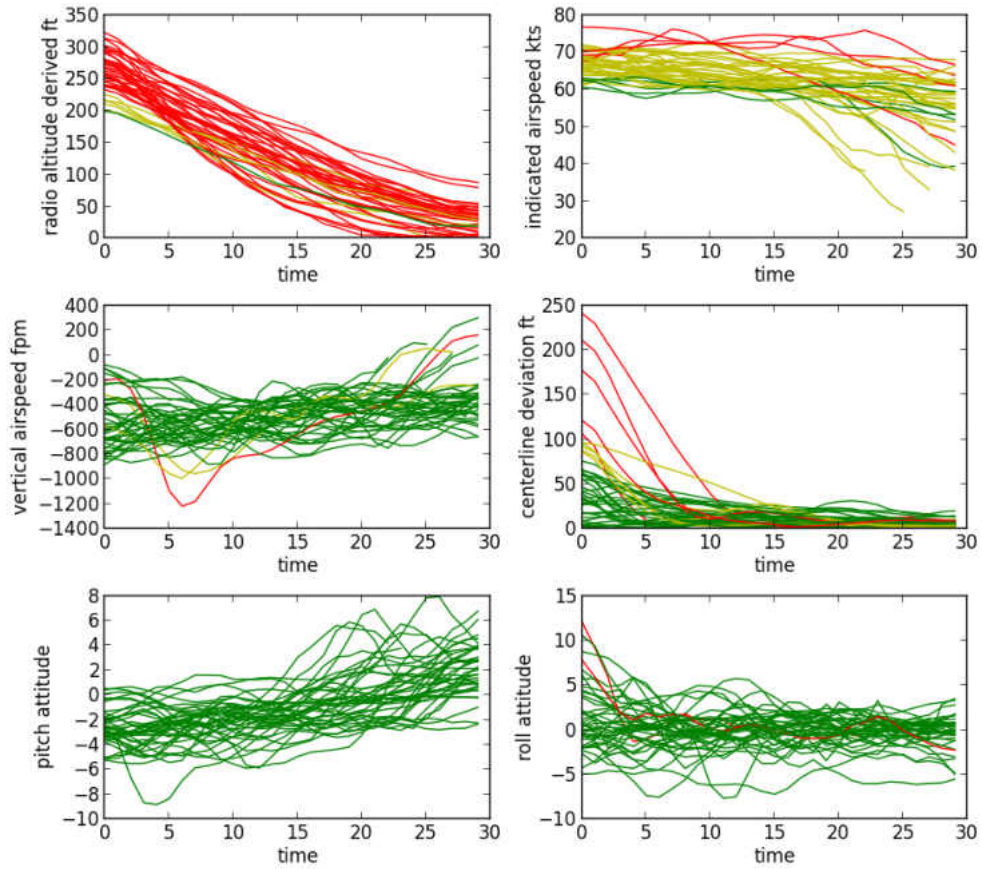


Figure 54: Cluster 12 - Line graph showing changes in altitude, airspeed, vertical speed, position, pitch and roll.

Table 16: Cluster 12 - Summary statistics showing the range of values for the descent profiles.

Parameter	Max	Min	Mean	Standard Deviation
Altitude	337.0	1.0	129.0	79.46
Airspeed	77.32	23.86	63.10	6.30
Vertical Speed	343.38	-1383.33	-481.51	199.53
Pitch	9.67	-11.62	-1.06	2.48
Roll	14.78	-13.75	0.07	2.89
Eng RPM	2459.70	624.10	1332.16	320.06
Vertical Acceleration	0.81	-0.38	0.003	0.078
Position	264.03	0.31	18.65	27.09

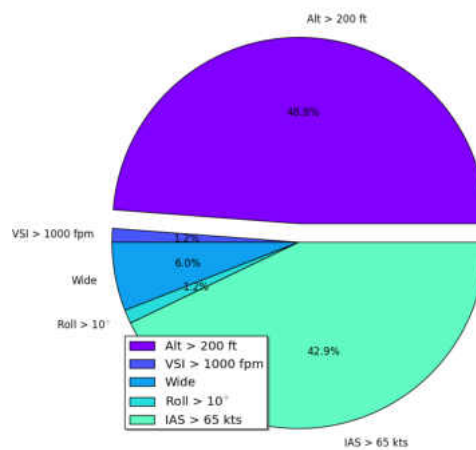
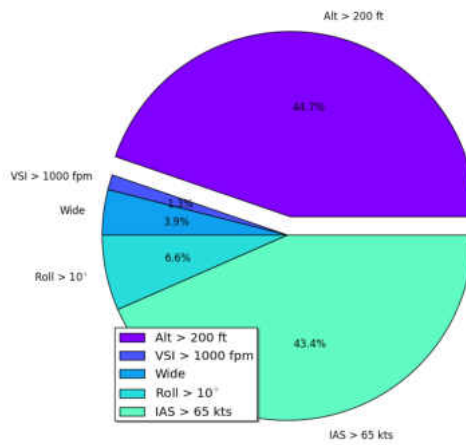
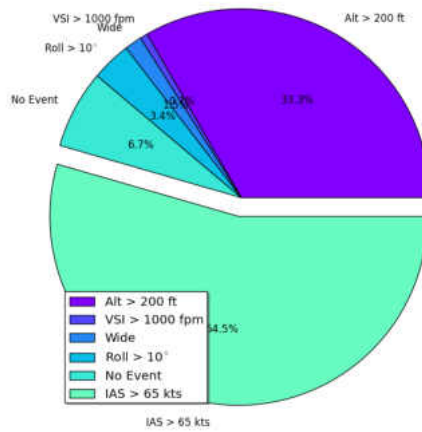


Figure 55: Pie charts for clusters 10, 11 and 12 which shows the evaluated results using the validation metrics.

Outliers

Figure 21 shows various densely populated neurons with similar characteristics, hence they were identified as clusters. Therefore, sparsely populated neurons are identified as outlier flights. Figure 56 shows the aerial view of their trajectories 80% of which were aborted between 50 and 100 feet AGL.



Figure 56: Outliers - Aerial view of the trajectories for the outlier approaches.

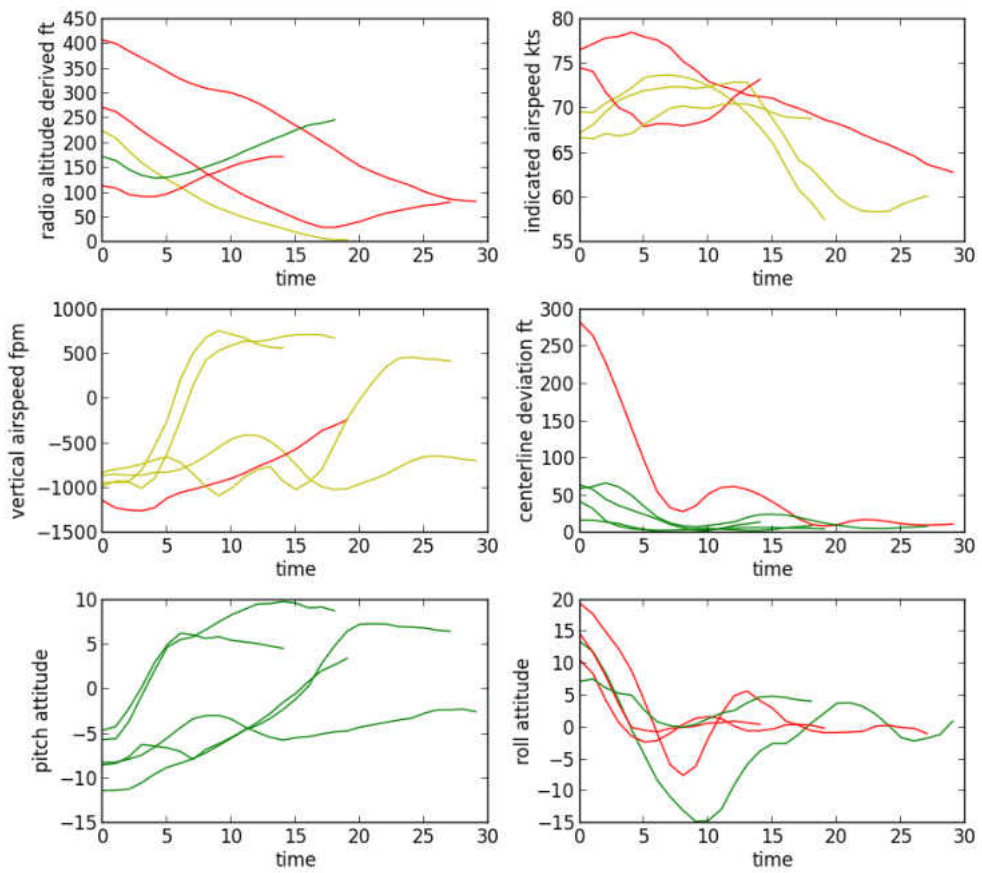


Figure 57: Outliers - Line graph showing changes in the altitude, airspeed, vertical speed, position, pitch and roll for the outlier flights.

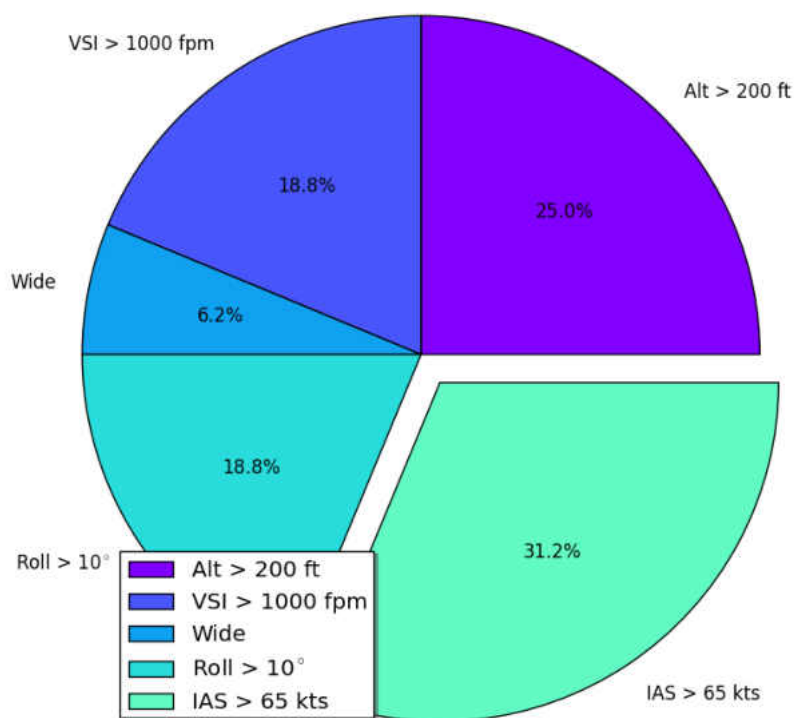


Figure 58: Outliers - Pie chart showing the outlier approaches that were evaluated based on the validation criteria.

Parameter	Max	Min	Mean	Standard Deviation
Altitude	425.0	5.0	153.39	91.82
Airspeed	79.31	56.92	69.50	4.70
Vertical Speed	810.43	-1460.21	-410.65	666.85
Pitch	10.49	-12.34	-0.79	6.12
Roll	22.96	-17.89	1.88	6.77
Eng RPM	2480.40	935.70	1613.84	622.29
Vertical Acceleration	0.37	-0.34	0.019	0.115
Position	310.17	0.31	27.96	50.03

Table 17: Outliers - Summary statistics for the outlier flights, which contains the highest altitude in the dataset; most of these approaches were aborted.

2.3.4 Summary

The evaluation of the above clusters demonstrated that VAEFM identified clusters of correlated approach configurations. The results indicate that for the month of June 2015, the approaches were predominately fast, some of which were beyond 10 knots of UND's mandated 61 knots, and were initiated around 250 feet AGL (which is also higher than expected). The wind component analysis did not indicate strong tailwinds as contributory factors for the excessively fast approaches. However, strong crosswinds or tailwinds are often present in approaches with significant deviations in roll in their descent. Excessive VSI is another issue and the maximum reported sink rate was over 2000 feet per minute.

There were several contributory factors for unstabilized approaches that were present. However, only two clusters were identified as unstable, and there were several outliers revealing fluctuations in aircraft trajectories and missed/aborted approaches. Clusters 1 and 7 contained a high number of aborted approaches, which are indications that some pilots took the necessary precaution, and performed a go-around maneuver; on average this decision occurs at 70 feet AGL.

Table 18 contains a summary of the results obtained from the various algorithms.

Table 18: The performance assessment of the various algorithms.

Algorithm	Distance Metric / Measure	Serial / Parallel Processing	Summary
K-Means	Euclidean	Serial	Average case analysis; identified k user-defined clusters. The distance metric did not produce intuitive cluster formations.
	Mahalanobis		
DBSCAN	Euclidean	Serial	Average case analysis; identified one cluster of high density. The distance metric influenced the cluster shape and the number of outliers.
	Mahalanobis		
DBSCAN	Euclidean	Serial	Detailed analysis; very fast runtime. Finding appropriate values for the hyperparameters was challenging. Poor quality clusters; most of the data was identified as outliers.
		Parallel	Encountered the same challenges as its serial counterpart. However, parallel DBSCAN's workload became imbalanced after 8 processors.
	DTW	Serial	Average runtime: 33 hours, poorly separated clusters.
		Parallel	Worst runtime: 34 hours (2 processors), best runtime 25 hours (16 processors); the performance gained was sluggish. There was a bottleneck after 16 processors. Hyperparameters produced poorly separated clusters.
SOM	Euclidean	Serial	Initial rate of false-positive flights were very high. Retraining improved the results.
	DTW & Avg. Rate of Change		Created an atypicality score using DTW warp cost and the average rate of change [87].
	DTW		Average runtime: 39 hours; clusters improved. DTW occasionally produced ineffective alignments. SOM's topological preservation was not upheld.
		Parallel	Faster computation time than DBSCAN, however it did not scale beyond 16 processors. Best runtime: 4 hours (16 processors), worst runtime: 27 hours (2 processors).
VAEFM	Reconstruction-based	Parallel	Scalable; Identified 12 clusters, several outliers with fluctuating trajectories and unstabilized approaches. High density regions can be problematic. Best runtime: < 5 minutes (256 processors), worst runtime: 35 hours (8 processors).

CHAPTER VII

CONCLUSION & FUTURE WORK

This dissertation entailed the analysis of time series data which presents many challenges due to its high dimensional nature. Therefore, the accumulation of temporal data makes its beyond the scope of efficient human analytical capabilities – hence the need for data mining techniques. However, the fundamental issue is representing temporal data in a manner where algorithms can detect patterns, extract useful features and establish correlations using appropriate similarity measures.

This research presented a framework for the analysis of time series data and a comparative analysis of serial and parallel implementations of various unsupervised machine learning algorithms were performed. The results indicated that the centroid-based and density-based algorithms' ability to identify clusters was affected by the high dimensional nature of the data. Additionally, their sequential nature made parallel implementations challenging, and the workload became imbalanced when the number of processors increased. The neural network-based clustering algorithm, i.e. the Self Organizing Map (SOM), identified correlations in the data, however it also suffered from scalability issues and potentially overfitting the data.

A new algorithm was developed, called the Variational Autoencoder Feature Map (VAEFM), which was designed to improve the scalability of SOM, enhance its training algorithm and minimize overfitting the data. VAEFM's design addresses two essential issues for effectively and efficiently analyzing time series – data representation and similarity assessment [6, 7]. The VAEFM achieved an average improvement in computation time of 70%, which demonstrate its scalability for high performance computing systems; and its parallel granularity

is determined by the volume of data.

The VAEFM analyzed 2582 flights that transpired at the Grand Forks International Airport (GFK) in June 2015. It identified 12 clusters, two of which contained stabilized approaches. The remaining results comprised of unstabilized approaches, and other contributory factors such as steep changes in roll, excessively high altitudes and/or fast airspeed, and rapid descent rates. Outlier approaches with unstable flight trajectories and missed/aborted approaches were also detected. The analysis of the wind components revealed that tailwinds of 5 knots or greater was not a contributory factor for excessively fast approaches. However, strong crosswinds or tailwinds were often present in approaches with unsafe low-level maneuvers.

The technique successfully identified clusters and outliers, some of which would have a lower detection rate using current flight safety analysis techniques. Furthermore, it identified anomalies with limited human intervention which is crucial for flight training institutions that conduct thousands of approaches each month.

1 Future Work

A limitation in the current methodology is the occurrence of densely populated neurons, which could potentially obscure anomalies due to the volume of data. Consequently, optimization methods can be applied to encourage the training algorithm to distribute densely populated data to neighboring neurons.

The focus of this research was on the identification of atypical descent patterns, however future efforts can extend the methodology to analyze: 1) other phases of flight, e.g. take-off, and 2) aircraft maintenance issues.

APPENDIX

Appendix A
Pseudocode Algorithms
1 K-Means

Algorithm 1: K-Means Algorithm

Input : k, x_1, x_2, \dots, x_n \triangleright k - number of clusters, x - training set data
Output: k clusters

- 1 Initialize k random cluster centers (centroids) $\mu_1, \mu_2, \dots, \mu_k$
- 2 **repeat**
- 3 For each data point x_i , assign it to the nearest centroid by minimizing h_j

$$h_j = \sum_{j=1}^k \sum_{i=1}^n \|x_i^j - \mu_j\|^2$$
- 4 **for** $k = 1 \rightarrow K$ **do**
- 5 $\mu_k =$ the mean of all data in cluster k
- 6 **end**
- 7 **until** *convergence criteria*

2 Density Based Spatial Clustering of Applications with Noise (DBSCAN)

Algorithm 2: The DBSCAN Algorithm

```
Input : Data, Eps, MinPts
1 ClusterId := 0
2 for  $i = 1 \rightarrow Data.size$  do
3   | Point :=  $data_i$ 
4   | if Point is UNCLASSIFIED then
5     | EXPANDCLUSTER(DATA, POINT, CLUSTERID)
6     | ClusterId := ClusterId + 1
7   | end
8 end
9 Procedure ExpandCluster(Data, Point, cID)
10  | result := regionQuery(Point, Eps)
11  | if result.size < MinPts then
12    | Point = NOISE
13  | else
14    | for  $j = 1 \rightarrow result.size$  do
15      | Mark  $result_j$  as CLASSIFIED
16      | neighbors := regionQuery( $result_j$ , Eps)
17      | if neighbors.size  $\geq$  MinPts then
18        | Merge neighbors with result
19      | end
20    | Assign result to cID
21  | end
22 end
```

3 Kohonen's Self Organizing Map (SOM)

Algorithm 3: The Serial Self Organizing Map Algorithm

- 1: Initialize SOM
- 2: Normalize input vectors **for** $t \leftarrow 1 \rightarrow N_{epochs}$ **do**
 - for** $i \leftarrow 1$ to M_{data} **do**
 - $I \leftarrow data(i)$
 - Present I to each neuron to find the BMU(I), using eq. 10

$$D = \sqrt{\sum_{i=1}^n (I_i - W_i)^2} \quad (10)$$

$$BMU = \operatorname{argmin} \|D\|$$

update the weights of the BMU and the neighboring neurons, using eq. 11:

$$W_{t+1} = W_t + \Theta_t * \alpha_t * (I_i - W_i) \quad (11)$$

$$\triangleright \alpha_t \text{ eq. } 12$$

$$\triangleright \Theta_t \text{ eq. } 13$$

$t \leftarrow t + 1$

$$\alpha_t = \alpha_0 * \exp(-t/N) \quad (12)$$

$$\Theta_t = \exp\left(-\frac{dist^2}{2 * \sigma_t^2}\right) \quad (13)$$

$$\sigma_t = SOM_{size} * \exp(-t/\lambda)$$

$$\lambda = N/SOM_{size}$$

In the DTW variant of the SOM, equation 10 is replaced with Dynamic Time Warping (DTW) (see equation 14) to identify: 1) the distance between each Neuron's weight vector and the time series data. The DTW heuristic calculates a similarity score between the neuron's weight vectors and the input data, called the warp cost.

$$D = \sqrt{\sum_{i=1}^n DTW_{cost}(I_i - W_i)^2} \quad (14)$$

$$BMU = \operatorname{argmin} \|D\|$$

4 Parallel SOM

Algorithm 4: Async SOM Algorithm

```

1 Initialize MPI variables
2 Initialize SOM for all processors
3 Assign training data to the master process
4 if process is MASTER then
5   ┌ MASTERPROCESSING()
6 else
7   ┌ WORKERPROCESSING()
8 MPI.Finalize()
9 Procedure masterProcessing()
10  ┌ for  $t \leftarrow 1 \rightarrow N_{epochs}$  do
11    ┌ for  $i \leftarrow 1 \rightarrow M_{data}$  do
12      ┌  $I \leftarrow data(i)$ 
13      ┌ MPI_SEND( $I$ , workers)
14      ┌ MPI_RECEIVE(weight_matrix, workers)
15      ┌ Accumulate weight_matrix
16      ┌ update master's SOM weights using eq 15
17      ┌  $updated_{SOM} \leftarrow master_{SOM}$ 
18      ┌ MPI_SEND( $updated_{SOM}$ , workers)
19      ┌ MPI_SEND( $I$ , workers)
20 Procedure workerProcessing()
21  ┌  $bmu \leftarrow calculateBMU(data(I))$ 
22  ┌ for  $i \leftarrow 1 \rightarrow N_{somLength}$  do
23    ┌ for  $j \leftarrow 1 \rightarrow M_{somWidth}$  do
24      ┌  $nf \leftarrow calculate\ neighborhood(bmu)$  eg. 13
25      ┌  $weight\_matrix_{ij} \leftarrow eq.\ 16$ 
26  ┌ MPI_SEND(weight_matrix, master)

```

$$W_{i+1} = \frac{\sum_{i=0}^n masterWeights(i) + weight_matrix(i)}{\sum_{i=0}^n weight_matrix(i) * numData} \quad (15)$$

$$W = worker_weight + \frac{\sum_{i=0}^n nf * \sqrt{\|I - W\|^2}}{\sum_{i=0}^n nf} \quad (16)$$

5 Variational Autoencoder Feature Map

Algorithm 5: Asynchronous Variational Autoencoder Feature Map Algorithm

Input : s, x, y ▷ s - SOM size, x - training set, y - test set

- 1 Initialize MPI variables
- 2 Initialize $SOM \leftarrow w, h$
- 3 **if** *process is master* **then**
- 4 \lfloor MASTERPROCESSING()
- 5 **else**
- 6 \lfloor WORKERPROCESSING()
- 7 **Procedure** *masterProcessing()*
- 8 **for** $i = 1 \rightarrow num_{data}$ **do**
- 9 **for** $s = 1 \rightarrow comm_{size}$ **do**
- 10 \lfloor MPISEND($x_i, worker_s$)
- 11 \lfloor $i \leftarrow i + 1$
- 12 **for** $s = 1 \rightarrow comm_{size}$ **do**
- 13 \lfloor MPIRECEIVE($weights, worker_s$)
- 14 Select Best Matching Unit
- 15 Identify BMU's neighbors
- 16 \lfloor Update SOM Weights (eq 17)
- 17 Divide test set into n equal partitions
- 18 **for** $s = 1 \rightarrow comm_{size}$ **do**
- 19 \lfloor MPISEND($[SOM, data_{partition}], worker_s$)
- 20 \lfloor MPIRECEIVE(flight Mappings, $worker_s$)
- 21 Merge mappings from all workers
- 22 **Procedure** *workerProcessing()*
- 23 Initialize d variational autoencoders (VAE) ▷ d - duration of flight
- 24 **for** $i = 0 \rightarrow d$ **do**
- 25 \lfloor Train $VAE_i \leftarrow x_i$
- 26 \lfloor MPISEND($VAE_{weights}$, master)

$$\begin{aligned}
Weights &= Weights_{old} + (\alpha * BMU_{weights} * \Theta_i) \\
&\triangleright \alpha - \text{learning rate} \\
\Theta_i &= \exp\left(-\frac{dist^2}{2 * region_i^2}\right) \\
\sigma &= SOM_{size}/2 \\
\lambda &= max_{iterations}/\sigma \\
region_i &= \sigma * \exp\left(-1 * (cur_{Iteration}/\lambda)\right)
\end{aligned} \tag{17}$$

BIBLIOGRAPHY

- [1] M. Ester, H. Peter Kriegel, J. S. Xu, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise.” AAAI Press, 1996, pp. 226–231.
- [2] M. Müller, “Dynamic time warping,” *Information retrieval for music and motion*, pp. 69–84, 2007.
- [3] P. Esling and C. Agon, “Time-series data mining,” *ACM Comput. Surv.*, vol. 45, no. 1, pp. 12:1–12:34, Dec. 2012. [Online]. Available: <http://doi.acm.org/10.1145/2379776.2379788>
- [4] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh, “Searching and mining trillions of time series subsequences under dynamic time warping,” in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’12. New York, NY, USA: ACM, 2012, pp. 262–270. [Online]. Available: <http://doi.acm.org/10.1145/2339530.2339576>
- [5] Q. Yang and X. Wu, “10 challenging problems in data mining research,” *International Journal of Information Technology & Decision Making*, vol. 5, no. 04, pp. 597–604, 2006.
- [6] T.-c. Fu, “A review on time series data mining,” *Engineering Applications of Artificial Intelligence*, vol. 24, no. 1, pp. 164–181, 2011.
- [7] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, “Querying and mining of time series data: experimental comparison of representations and distance measures,” *Proceedings of the VLDB Endowment*, vol. 1, no. 2, pp. 1542–1552, 2008.
- [8] M. Längkvist and A. Loutfi, “Not all signals are created equal: Dynamic objective auto-encoder for multivariate data,” in *NIPS workshop on Deep Learning and Unsupervised Feature Learning*, 2012.
- [9] V. Guralnik and J. Srivastava, “Event detection from time series data,” in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1999, pp. 33–42.
- [10] T. R. Chidester, “Understanding normal and atypical operations through analysis of flight data,” in *Proceedings of the 12th International Symposium on Aviation Psychology, Dayton, OH*. Citeseer, 2003, pp. 239–242.
- [11] S. Mahendran *et al.*, “Enhancing flight data monitoring and analysis can increase flight safety,” *Journal of Aeronautics & Aerospace Engineering*, vol. 2015, 2015.
- [12] L. Li, “Anomaly detection in airline routine operations using flight data recorder data,” Ph.D. dissertation, Massachusetts Institute of Technology, 2013.

- [13] L. Rokach and O. Maimon, *Data Mining with Decision Trees: Theory and Applications*, ser. Series in Machine Perception and Artificial Intelligence. World Scientific, 2008. [Online]. Available: <http://books.google.com/books?id=G1KIIR78OxkC>
- [14] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.
- [15] T. N. Phyu, “Survey of Classification Techniques in Data Mining,” *Proceedings of the International MultiConference of Engineers and Computer Scientists 2009 IMECS*, vol. I, 2009.
- [16] CAST/ICAO, “Phase of flight definitions and usage notes,” 04/2013. [Online]. Available: <http://www.intlaviationstandards.org/Documents/PhaseofFlightDefinitions.pdf>
- [17] (FSF) Flight Safety Foundation, “FSF ALAR Briefing Note 7.1, Stabilized approach,” 2009. [Online]. Available: <http://www.skybrary.aero/bookshelf/books/864.pdf>
- [18] Airbus - Flight Operations Briefing Notes, “Aircraft Energy Management during Approach,” 2005. [Online]. Available: http://www.airbus.com/fileadmin/media_gallery/files/safety_library_items/AirbusSafetyLib.-FLT_OPS-APPR-SEQ03.pdf
- [19] AOPA, “What is General Aviation,” 2009. [Online]. Available: http://www.aopa.org/info/what_ga.pdf
- [20] W. Allen, D. Blond, A. Gellman, G. A. M. Association, N. A. of State Aviation Officials (U.S.), and I. MergeGlobal, *General Aviation’s Contribution to the U.S. Economy*. General Aviation Manufacturers Association, 2009.
- [21] B. Elias, *Securing General Aviation*. DIANE Publishing Company, 2009.
- [22] K. Shetty, “Current and Historical Trends in General Aviation in the United States,” MIT International Center for Air Transportation (ICAT) Department of Aeronautics & Astronautics Massachusetts Institute of Technology, Tech. Rep., Aug. 2012. [Online]. Available: <http://dspace.mit.edu/handle/1721.1/72392>
- [23] NTSB, “General aviation safety.” [Online]. Available: <http://www.nts.gov/safety/mwl-2.html>
- [24] NTSB, “Annual Aviation Statistics for 2012 Released: General aviation accidents continue upward trend, no fatalities on US airlines or commuters.” [Online]. Available: <http://www.nts.gov/news/2013/130806b.html>

- [25] S. K. Lau, “General Aviation Flight Data Monitoring Fly with Intelligence – Best Practices to Improve the Safety and Efficiency of Flight Operations,” CAPACG, LLC., Tech. Rep., 2007. [Online]. Available: http://www.ihst.org/portals/54/AttachmentH_GeneralAviationFlightDataMonitoring.pdf
- [26] AOPA, “General Aviation Information and Statistics,” 2011. [Online]. Available: <http://www.aopa.org/whatsnew/stats/safety.html>
- [27] D. R. Hunter and U. States., *Risk perception and risk tolerance in aircraft pilots [electronic resource] / David R. Hunter*. U.S. Dept. of Transportation, Federal Aviation Administration, Office of Aerospace Medicine Washington, DC, 2002.
- [28] D. R. Hunter, “Risk perception among general aviation pilots,” *The International Journal of Aviation Psychology*, vol. 16, no. 2, pp. 135–144, Apr 2006. [Online]. Available: http://dx.doi.org/10.1207/s15327108ijap1602_1
- [29] J. Reason, *Human Error*. Cambridge [England] ; New York : Cambridge University Press, 1990. xv, 302 p., 1990.
- [30] D. A. Wiegmann, A. Boquet, C. Detwiler, K. Holcomb, T. Faaborg, D. A. Wiegmann, P. D, P. D, A. Boquet, P. D, C. Detwiler, K. Holcomb, and T. Faaborg, “Human error and general aviation accidents: A comprehensive, fine-grained analysis using hfacs,” in *DOT/FAA/AM-05/24*, 2005.
- [31] J. Macqueen, “Some methods for classification and analysis of multivariate observations,” in *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281–297.
- [32] G. Fung, “A comprehensive overview of basic clustering algorithms,” 2001.
- [33] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg, “Top 10 algorithms in data mining,” *Knowl. Inf. Syst.*, vol. 14, no. 1, pp. 1–37, Dec. 2007. [Online]. Available: <http://dx.doi.org/10.1007/s10115-007-0114-2>
- [34] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, “An efficient k-means clustering algorithm: Analysis and implementation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 881–892, Jul. 2002. [Online]. Available: <http://dx.doi.org/10.1109/TPAMI.2002.1017616>
- [35] K. G. Derpanis, “K-Means Clustering,” no. 1995, pp. 1–2, 2006.
- [36] R. Xu and Ii, “Survey of clustering algorithms,” vol. 16, no. 3, pp. 645–678, May 2005.

- [37] B. Amidan and T. Ferryman, “Atypical event and typical pattern detection within complex systems,” in *2005 IEEE Aerospace Conference*. IEEE, 2005, pp. 3620–3631. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1559667>
- [38] Amidan, B.G. and Ferryman, T.A., “APMS SVD Methodology and Implementation,” Pacific Northwest National Laboratory, Tech. Rep., 2000.
- [39] M. Gariel, A. Srivastava, and E. Feron, “Trajectory clustering and an application to airspace monitoring,” *Intelligent Transportation Systems, IEEE Transactions on*, vol. 12, no. 4, pp. 1511–1524, 2011.
- [40] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, “Density-based clustering in spatial databases: The algorithm gbscan and its applications,” *Data Min. Knowl. Discov.*, vol. 2, no. 2, pp. 169–194, Jun. 1998. [Online]. Available: <http://dx.doi.org/10.1023/A:1009745219419>
- [41] K. Mumtaz, M. Studies, and T. Nadu, “An Analysis on Density Based Clustering of Multi Dimensional Spatial Data,” *Indian Journal of Computer Science and Engineering*, vol. 1, no. 1, pp. 8–12.
- [42] L. Li, M. Gariel, R. J. Hansman, and R. Palacios, “Anomaly detection in onboard-recorded flight data using cluster analysis,” in *2011 IEEE/AIAA 30th Digital Avionics Systems Conference*. IEEE, Oct. 2011, pp. 4A4–1–4A4–11. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6096068>
- [43] M. Coppola and M. Vanneschi, “High-performance data mining with skeleton-based structured parallel programming,” *Parallel Computing*, vol. 28, no. 5, pp. 793–813, 2002.
- [44] D. Arlia and M. Coppola, “Experiments in parallel clustering with dbscan,” in *European Conference on Parallel Processing*. Springer, 2001, pp. 326–331.
- [45] M. Chen, X. Gao, and H. Li, “Parallel dbscan with priority r-tree,” in *Information Management and Engineering (ICIME), 2010 The 2nd IEEE International Conference on*, April 2010, pp. 508–511.
- [46] S. Brecheisen, H.-P. Kriegel, and M. Pfeifle, “Parallel density-based clustering of complex objects,” in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2006, pp. 179–188.
- [47] Y. X. Fu, W. Z. Zhao, and H. F. Ma, “Research on parallel dbscan algorithm design based on mapreduce,” in *Advanced Materials Research*, vol. 301. Trans Tech Publ, 2011, pp. 1133–1138.
- [48] Y. Guo and R. Grossman, “A fast parallel clustering algorithm for large spatial databases, high performance data mining,” 2002.
- [49] A. Zhou, S. Zhou, J. Cao, Y. Fan, and Y. Hu, “Approaches for scaling dbscan algorithm to large spatial databases,” *Journal of computer science and technology*, vol. 15, no. 6, pp. 509–526, 2000.

- [50] A. C. A. Neto, T. L. C. da Silva, V. A. E. de Farias, J. A. F. Macêdo, and J. de Castro Machado, "G2p: A partitioning approach for processing dbscan with mapreduce," in *International Symposium on Web and Wireless Geographical Information Systems*. Springer, 2015, pp. 191–202.
- [51] M. M. A. Patwary, D. Palsetia, A. Agrawal, W.-k. Liao, F. Manne, and A. Choudhary, "A new scalable parallel dbscan algorithm using the disjoint-set data structure," in *High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference for*. IEEE, 2012, pp. 1–11.
- [52] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998.
- [53] J. A. Freeman and D. M. Skapura, *Neural networks: algorithms, applications, and programming techniques*. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 1991.
- [54] P. Sydenham and R. Thorn, *Handbook of measuring system design*, ser. Handbook of Measuring System Design. Wiley, 2005, no. v. 2.
- [55] P. Stefanović and O. Kurasova, "Visual analysis of self-organizing maps," *Nonlinear Analysis: Modelling and Control*, vol. 16, no. 4, pp. 488–504, 2011.
- [56] Y. Singh and A. S. Chauhan, "Neural networks in data mining," *Journal of Theoretical and Applied Information Technology*, vol. 5, no. 6, pp. 37–42, 2009.
- [57] M. Långkvist, L. Karlsson, and A. Loutfi, "A review of unsupervised feature learning and deep learning for time-series modeling," *Pattern Recognition Letters*, vol. 42, pp. 11–24, 2014.
- [58] T. Kohonen, "Neurocomputing: foundations of research," J. A. Anderson and E. Rosenfeld, Eds. Cambridge, MA, USA: MIT Press, 1988, ch. Self-organized formation of topologically correct feature maps, pp. 509–521. [Online]. Available: <http://dl.acm.org/citation.cfm?id=65669.104428>
- [59] Kohonen, Teuvo, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.
- [60] G. Cabanes and Y. Bennani, "Learning the Number of Clusters in Self Organizing Map," in *Self-Organizing Maps*, G. K. Matsopoulos, Ed., 2010.
- [61] J. Vesanto, E. Alhoniemi, and S. Member, "Clustering of the Self-Organizing Map," *IEEE TRANSACTIONS ON NEURAL NETWORKS*, vol. 11, no. 3, pp. 586–600, 2000.
- [62] E. Turban, R. Sharda, D. Delen, D. King, and J. E. Aronson, "Neural Networks for Data Mining," in *Business Intelligence: A Managerial Approach*, 2nd ed. Prentice Hall, Inc., 2010, ch. 6.

- [63] A. Ultsch, *Self-organizing neural networks for visualisation and classification*. Springer, 1993.
- [64] R. D. Lawrence, G. S. Almasi, and H. E. Rushmeier, “A scalable parallel algorithm for self-organizing maps with applications to sparse data mining problems,” *Data Mining and Knowledge Discovery*, vol. 3, no. 2, pp. 171–195, 1999.
- [65] C. Garcia, M. Prieto, and A. Pascual-Montano, “A speculative parallel algorithm for self-organizing maps,” *Proceedings of Parallel Computing 2005 (ParCo 2005)*, pp. 615–622, 2005.
- [66] A. Rauber, P. Tomsich, and D. Merkl, “parsom: A parallel implementation of the self-organizing map exploiting cache effects: making the som fit for interactive high-performance data analysis,” in *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, vol. 6. IEEE, 2000, pp. 177–182.
- [67] P. Ozdzyński, A. Lin, M. Liljeholm, and J. Beatty, “A parallel general implementation of kohonen’s self-organizing map algorithm: performance and scalability,” *Neurocomputing*, vol. 44, pp. 567–571, 2002.
- [68] B. Silva and N. Marques, “A hybrid parallel som algorithm for large maps in data-mining,” *New Trends in Artificial Intelligence*, 2007.
- [69] Y. Bengio, L. Yao, G. Alain, and P. Vincent, “Generalized denoising auto-encoders as generative models,” in *Advances in Neural Information Processing Systems*, 2013, pp. 899–907.
- [70] S. Takaki and J. Yamagishi, “Constructing a deep neural network based spectral model for statistical speech synthesis,” in *Recent Advances in Nonlinear Speech Processing*. Springer, 2016, pp. 117–125.
- [71] C. Häusler, A. Susemihl, M. P. Nawrot, and M. Opper, “Temporal autoencoding improves generative models of time series,” *arXiv preprint arXiv:1309.3103*, 2013.
- [72] G. Alain and Y. Bengio, “What regularized auto-encoders learn from the data-generating distribution,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3563–3593, 2014.
- [73] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *arXiv preprint arXiv:1312.6114*, 2013.
- [74] D. J. Rezende, S. Mohamed, and D. Wierstra, “Stochastic backpropagation and approximate inference in deep generative models,” *arXiv preprint arXiv:1401.4082*, 2014.
- [75] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, “Generating sentences from a continuous space,” *arXiv preprint arXiv:1511.06349*, 2015.

- [76] S. Kullback and R. A. Leibler, “On information and sufficiency,” *Ann. Math. Statist.*, vol. 22, no. 1, pp. 79–86, 03 1951. [Online]. Available: <http://dx.doi.org/10.1214/aoms/1177729694>
- [77] C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther, “How to train deep variational autoencoders and probabilistic ladder networks,” *arXiv preprint arXiv:1602.02282*, 2016.
- [78] P. Danielsson, “Euclidean distance mapping,” *Computer Graphics and Image Processing*, vol. 14, no. 3, pp. 227–248, Nov. 1980. [Online]. Available: [http://dx.doi.org/10.1016/0146-664X\(80\)90054-4](http://dx.doi.org/10.1016/0146-664X(80)90054-4)
- [79] P. C. Mahalanobis, “On the generalised distance in statistics,” in *Proceedings National Institute of Science, India*, vol. 2, no. 1, Apr. 1936, pp. 49–55. [Online]. Available: <http://ir.isical.ac.in/dspace/handle/1/1268>
- [80] D. J. Berndt and J. Clifford, “Using dynamic time warping to find patterns in time series.” in *KDD workshop*, vol. 10, no. 16. Seattle, WA, 1994, pp. 359–370.
- [81] E. Keogh and S. Kasetty, “On the need for time series data mining benchmarks: A survey and empirical demonstration,” *Data Min. Knowl. Discov.*, vol. 7, no. 4, pp. 349–371, Oct. 2003. [Online]. Available: <http://dx.doi.org/10.1023/A:1024988512476>
- [82] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 8, pp. 1798–1828, 2013.
- [83] S. Parshutin and G. Kuleshova, “Time warping techniques in clustering time series,” in *Proceedings of 14th International Conference on Soft Computing MENDEL*, 2008, pp. 175–180.
- [84] E. Romano and G. Scepi, “Integrating time alignment and self-organizing maps for classifying curves,” *Electronic Proceedings of Knowledge Extraction and Modeling*, 2006.
- [85] P. Somervuo and T. Kohonen, “Self-organizing maps and learning vector quantization for feature sequences,” *Neural Processing Letters*, vol. 10, no. 2, pp. 151–159, 1999.
- [86] E. J. Keogh and M. J. Pazzani, “Derivative dynamic time warping.” in *SDM*, vol. 1. SIAM, 2001, pp. 5–7.
- [87] S. A. Clachar, “Identifying and analyzing atypical flights by using supervised and unsupervised approaches,” *Transportation Research Record: Journal of the Transportation Research Board*, no. 2471, pp. 10–18, 2015.
- [88] E. Keogh and C. A. Ratanamahatana, “Exact indexing of dynamic time warping,” *Knowledge and information systems*, vol. 7, no. 3, pp. 358–386, 2005.

- [89] E. Smart and D. Brown, “A two-phase method of detecting abnormalities in aircraft flight data and ranking their impact on individual flights,” *Intelligent Transportation Systems, IEEE Transactions on*, vol. 13, no. 3, pp. 1253–1265, 2012.
- [90] E. Smart, D. Brown, and J. Denman, “Combining multiple classifiers to quantitatively rank the impact of abnormalities in flight data,” *Applied Soft Computing*, 2012.
- [91] C. Jesse, H. Liu, E. Smart, and D. Brown, “Analysing flight data using clustering methods,” in *Knowledge-Based Intelligent Information and Engineering Systems*, ser. Lecture Notes in Computer Science, I. Lovrek, R. Howlett, and L. Jain, Eds. Springer Berlin Heidelberg, 2008, vol. 5177, pp. 733–740.
- [92] B. Efron, “Bootstrap methods: another look at the jackknife,” in *Breakthroughs in Statistics*. Springer, 1992, pp. 569–593.
- [93] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham, and T. S. Woodall, “Open MPI: Goals, concept, and design of a next generation MPI implementation,” in *Proceedings, 11th European PVM/MPI Users’ Group Meeting*, Budapest, Hungary, September 2004, pp. 97–104.
- [94] L. Dalcín, R. Paz, and M. Storti, “Mpi for python,” *Journal of Parallel and Distributed Computing*, vol. 65, no. 9, pp. 1108 – 1115, 2005. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0743731505000560>
- [95] L. Dalcín, R. Paz, M. Storti, and J. D’Elía, “Mpi for python: Performance improvements and mpi-2 extensions,” *J. Parallel Distrib. Comput.*, vol. 68, no. 5, pp. 655–662, May 2008. [Online]. Available: <http://dx.doi.org/10.1016/j.jpdc.2007.09.005>
- [96] L. D. Dalcin, R. R. Paz, P. A. Kler, and A. Cosimo, “Parallel distributed computing using python,” *Advances in Water Resources*, vol. 34, no. 9, pp. 1124 – 1139, 2011, new Computational Methods and Software Tools. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0309170811000777>
- [97] P. J. Rousseeuw, “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis,” *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.
- [98] J. Lampinen and T. Kostiainen, “Overtraining and model selection with the self-organizing map,” in *Neural Networks, 1999. IJCNN’99. International Joint Conference on*, vol. 3. IEEE, 1999, pp. 1911–1915.
- [99] S. Sinha, T. Singh, V. Singh, and A. Verma, “Epoch determination for neural network by self-organized map (som),” *Computational Geosciences*, vol. 14, no. 1, pp. 199–206, 2010.

- [100] L. Nadalski, “Automated surface observing system (asos) user’s guide,” *Natl. Oceanic and Atmos. Admin., Silver Spring, Md.* (Available at <http://www.nws.noaa.gov/asos/pdfs/aum-toc.pdf>), 1998.
- [101] F. M. H. No, “Surface weather observations and reports,” 2005.