university of groningen / faculty of science and engineering

Automatic Timestep Selection for In Situ Visualization of Large Data

MSc Thesis Computing Science

Author
Alpheaus Feltham
S4216768
Supervisors
Prof. Steffen Frey
Prof. Jiri Kosinka

University of Groningen

September 2023

Abstract

This thesis examines in-situ simulation timestep selection and reconstruction, through the use of a predictive autoencoder system. This autoencoder is used to detect various anomalies within a simulated 2D Kármán Vortex Street ensemble in order to select timesteps of interest, as well as to reconstruct timestep data from a given sub-sampling based on the selected timesteps. This autoencoder setup is trained to predict subsequent timesteps in a series. The detection capacity of this autoencoder is tested against a non-predictive autoencoder system. This is done using a number of datasets which are copied from the original ensemble and subsequently modified to contain artificial anomalies of various types. Then the detection and reconstruction ability of both systems are compared against each other. The thesis finds that both autoencoder setups are similar in their detection capacity, but the predictive model performs slightly better in data reconstruction. The detection system is found to be only somewhat effective otherwise, and various possible reasons therefore as well as some potential solutions and alternatives are discussed.

Contents

Acknowledgements

I would like to thank both Dr. Steffen Frey and Jiri Kosinka for their time supervising, and providing feedback for this thesis project. Their advice and support was invaluable and greatly appreciated. I would also like to thank Luka Petrovic for their help in reviewing and editing the language in this document. Finally, I would like to thank the friends and family who have shown their support and provided advice.

1 Introduction

As computational technology and available software has improved over the decades, the scientific community has taken the crucial step of making good use of the resources available to it. This has led to the development of increasingly intricate, complex and extensive large-scale models and simulations for the purposes of study and research. In-depth engineering simulations, expansive climatology simulations and cosmological simulations on a grand scale, among others are all prevalent examples. One of the common features of these simulated models is simply the large amounts of data that is generated as timesteps for the end-user. When dealing with the vast quantities of data that is created as the simulation output, the transfer of massive quantities of timestep data to external storage media can be a significant bottleneck for larger simulations, as the supercomputers and high-performance computing clusters these simulations are run on are not designed to handle this volume of data.

This is not a new problem in the realm of modelling and simulation, and numerous methods are already in general use to mitigate the issue[5]. Depending on the requirements of the research, these methods can vary. If only a hollistic view of the data is needed, an abstraction or subset of the output may be sufficient. But, often the phenomena being studied are specific events occurring within a larger and more complex pattern of resultant data. This means that specific timesteps within the output of a simulation are often much more desirable than large portions of the rest of the data. The objective is to design an algorithm that selects and saves these timesteps as appropriate, while discarding the unselected remainder. This selection of timesteps during the actual processing of a simulation is often referred to as "In-Situ Timestep Selection". An overview of some of the various timestep selection approaches is described by Bruder et al[5]. A more in-depth look is provided in the following Related Works section.

Some more recent works have begun looking into using Neural Networks to assist in the timestep selection process[14], though the system reviewed in the related works section is specifically selecting timesteps with the goal of Spatiotemporal Volume Visualization. Given the specialization of many of these systems, it is of interest to determine if a more generic detection system is possible. Generic anomaly detection is a common use for Autoencoders[9][4], which can be trained to reconstruct data based on a training dataset. This allows the reconstructions to be compared to the original in order to detect discrepancies. Therefore, the goal of this thesis project is to investigate the effectiveness of an autoencoder for in-situ timestep selection and data reconstruction.

This thesis describes a simple autoencoder used as a test case to determine the validity of such a system for timestep selection. This includes the results of a number of experiments run to determine the capacity of this system when it comes to the detection of anomalies and reconstruction of timesteps. The intention of the system presented here is to be trained on a baseline of "non-anomalous" or expected outputs, allowing it to detect instances that do not conform to the trained values. The exact specifications of what is to be considered anomalous (eg. what kind of data is of interest) up to the end user, as they will need to provide a baseline set of "non-anomalous" training data for the autoencoder to train on. For the purposes of determining

anomaly detection, this thesis covers three basic types of artificial anomalous data as described in the following section, as well as an adjusted training set that isolates a more turbulent subset of the flow data to use as anomalous data instead of the artificially created sets.

Chapter 2 will provide an overview of a number of related works, providing a brief synopsis and describing their relation to this thesis. Chapter 3 will go into detail on the methodology used, describing the systems and processes this thesis implements. This includes the structure of the autoencoders, how they were separately trained, and the anomaly detection system for which they are used. Chapter 4 will go over the parameters of the experiment and how they were selected, as well as describe the specifications of the system on which the program was run. It will discuss the data used in the experiments, and how the ensembles were augmented with artificially anomalous datasets. Following this, chapter 5 will briefly present the training and validation losses of the predictive and regular training processes, the baseline system output in a non-anomalous scenario, and the standard output of the system. It will then present the system output for the anomalous datasets, the detection accuracy and the results of the reconstruction experiments. Chapter 6 provides an in-depth analysis of the results presented in chapter 5, as well as the conclusions that can be drawn from them. Finally, Chapter 7 will summarize the conclusions and present prospective work and experiments that could follow.

2 Related Works

This thesis is based on similar works in the field of timestep selection, visualization, and anomaly detection. A number of the papers referenced for this thesis are described in this section.

2.1 Timestep Selection

As described above, the core of this thesis is simulation timestep selection. In order to properly review the various problems and solutions already existing in this field, the following papers were referenced. These papers cover the numerous techniques and methods used by different authors for various types of simulations circumstances. Xin Tong, Teng-Yok Lee and Han-Wei Shen[15] describe a timestep selection algorithm for which the goal is to select a subset K of the total collection of timesteps N which maximizes the information contained by the original data. To do this, they use a method known as "Dynamic Time Warping" to determine the similarity between two given timesteps in separate series. They propose that this method could be used to determine the similarity of a specific timestep to as many of its neighbours as possible so that an optimal subset of timesteps could be found to minimize data loss. S. Frey and T.Ertl[8] provide a solution more specific for volume visualization with an algorithm that makes use of a method known as the "Wasserstein Metric", or "Earth-Movers Distance". This metric essentially calculates the minimum cost of turning one mass distribution into another, and is used by Frey and Ertl to determine the flow of voxels over time within their data. They can then select their timesteps using more standard distance-based metrics in order to get optimal coverage of the volume flow data. Yoshiaki Yamaoka, Kengo Hayashi, Naohisa Sakamoto and Jorji Nonaka[18] propose a system that adaptively adjusts the sampling rate of timesteps during simulation runtime. This is done by calculating the variation between timesteps for a set of subvolumes. The intensity of these timestep divergences can then be used to determine the best timestep sampling rate for different regions of the data. Much of the work in this thesis is based on the timestep selection work demonstrated by S. Frey, T. Ertl and Gleb Tkachev[14]. In their paper they demonstrate a simple Neural Network setup being used to predict events within a temporal data stream based on local data, eg. the N most recent timesteps. The accuracy of these predictions are used for a number of things, including ensemble analysis and timestep selection.

2.2 Anomaly Detection

Based on the above papers, an approach to timestep selection based on anomaly detection through autoencoders was selected. The following papers were referenced for various anomaly detection methods using autoencoders. Similar to the paper by S. Frey, T. Ertl and Gleb Tkachev[14], authors Battikh and Lenskiy[4] describe a system which also uses the comparison between the output of a neural network setup and the expected output to detect anomalies in given data. In this case, they use an Autoencoder setup to reconstruct a given input for comparison. Their variant includes a latent training phase for the Autoencoder that freezes the Encoder and Decoder networks to train only the latent layer of the Autoencoder. The basics of anomaly detection with autoencoders are also described by An and Cho[2], as they describe the

method of using the reconstruction loss to determine the probability of input data being anomalous. Tziolas et al.[16] also describe the use of an autoencoder for anomaly detection, albeit for time-series data in a real-time manufacturing setting. Similar work done by Lyudchik[12] uses deep autoencoders for outlier detection for computer vision datasets. This thesis incorporates autoencoders in a similar manner in its anomaly-detection mechanism, reconstructing the input data and using the loss to determine deviation from the norm.

2.3 Visualization Through Dimension Reduction

Autoencoders are often used as data visualization tools, in which they perform dimension reduction on given data. This is frequently done through the extraction or manipulation of the feature vector, as can be seen in the paper by Battey, Coffing and Kern[3]. They describe using a variational autoencoder architecture to better cluster datapoints through training the system to make better distinctions between data points in latent space. Similarly, Wang and Gu [17] use a deep variational autoencoder for dimension reduction and clustering purposes to visualize single-cell RNA sequence data. While none of these techniques are explicitly used in this thesis, their use of the feature vector was referenced and used as inspiration for the feature-vector based detection described later.

2.4 Predictive Models

In order to be able to reconstruct missing data from a linear series of timesteps that describe a changing simulation over time, predictive modelling is required to determine subsequent timesteps as accurately as possible. Without any kind of predictive modelling, the system should not be capable of adapting to the changing timesteps. The following recent papers have begun exploring the use of autoencoders in prediction models or as prediction models themselves. Jin et al.[10] describe the use of a long-and-short term memory based autoencoder architecture in their prediction model for a variety of atmospheric conditions, including air quality, humidity and temperature. Lu, Hsu and Huang[11] also demonstrate a hybrid prediction model incorporating an autoencoder and a gated recurrent unit in order to predict the remaining useful lifespan of factory equipment.

3 Methodology

In order to solve the problem of anomaly detection for the purposes of timestep selection, as well as the reconstruction of unselected timesteps, this thesis proposes the use of an autoencoder system. The idea being that autoencoders are already used for anomaly detection[14][4][2][16][12], and the method by which they achieve this is through encoding and subsequently reconstructing a given image. While a standard autoencoder would function for anomaly detection, it should not function nearly as well for data reconstruction, as it is a system that simply reconstructs a given input. Because of this, it should not be able to recreate any changes that should happen from one timestep to the next.

The structure used for the autoencoder was modified from a basic autoencoder setup. Figure 1 shows a diagram of the autoencoders' structure, in this case, three 2D convolution layers with a decreasing number of filters for each layer in the encoder (16, 8 and 4), and an increasing number of filters for the decoder (4, 8 and 16). The kernel size for each 2D convolution layer is 3, and each has a stride of 2. A Tanh activation function was used for each layer. This structure allows for basic anomaly detection via reconstruction and the extraction of a feature vector before encoding again without taking too long to train or run in-situ. A few other structures were tested during development, primarily adjusting the number of layers and neurons, but this setup was found to detect anomalous timesteps well enough without excessively extending training time.

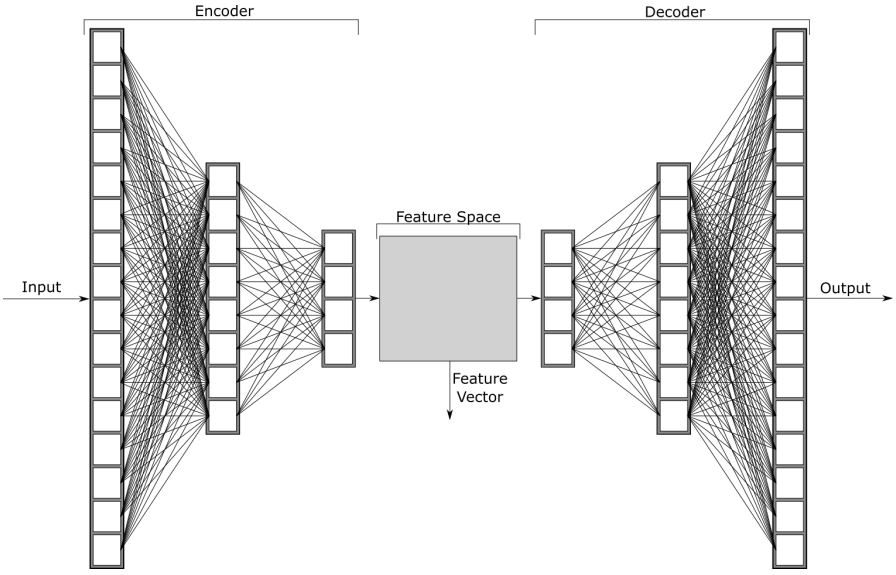


Figure 1: Visual representation of the autoencoder structure.

This autoencoder structure was then used as the anomaly detection system that acted as the

core of the timestep selector. Figure 2 shows the process of the anomaly detection algorithm, in which the autoencoder first encodes a given timestep into a feature vector. The feature vector is set aside and, if this is not the first timestep, compared to the previous feature that was set aside using a mean average error (MAE) loss function. The feature vector is then de-coded, and the reconstructed image is then compared to the actual subsequent timestep, once again using an MAE loss function. If the reconstruction loss and feature vector loss are both above their respective thresholds, that timestep is saved.

The reconstruction loss represents the deviation of the generated timestep from the actual timestep, and therefore the actual timesteps' deviation from the expected value. The threshold value here determines the algorithms' sensitivity to anomalies. The lower the threshold value, the more likely the algorithm should be to save a timestep based on unexpected changes in the data, and vice-versa. The feature vector loss allows for the detection of sudden major change from one timestep to the next. The threshold value here should adjust the algorithms' sensitivity to change in the data, the lower the value the smaller the change between timesteps needed to trigger this threshold. These threshold values should be adjusted during setup to best match the sensitivity desired by the end user.

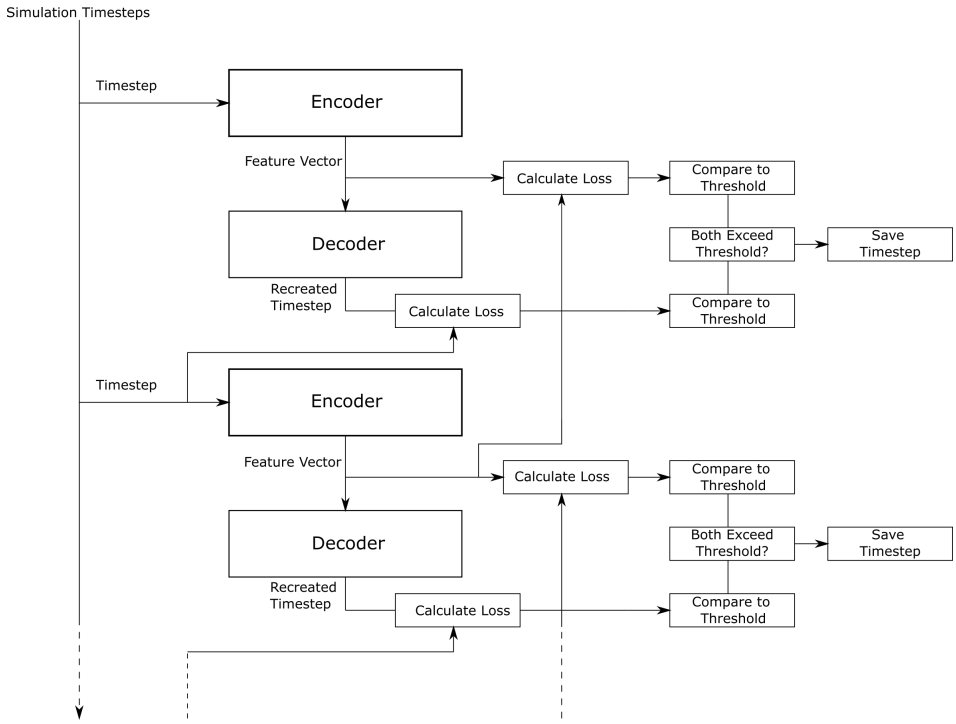
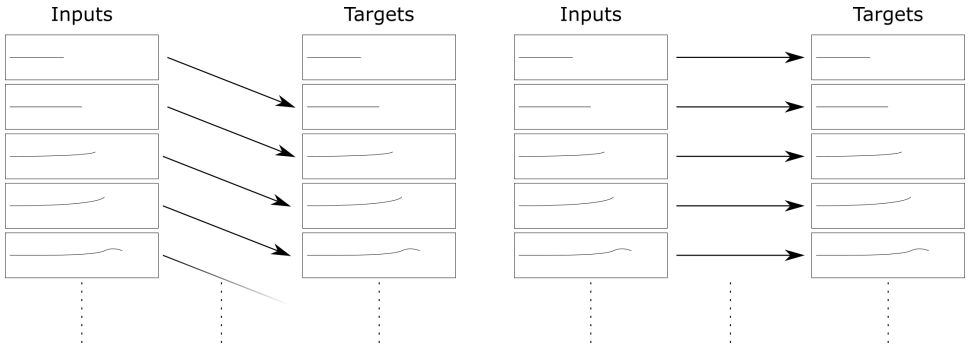


Figure 2: Visual representation of the anomaly detection process.

In order to solve the reconstruction issue, the training for the autoencoder was altered. This was primarily done through the training process, in this case through altering the training targets of the autoencoder. While standard procedure would be to provide the autoencoder with the initial input again as the training target for the autoencoder (as represented in figure 3b), in this case the input for the subsequent timestep would be used as the target for each individual input timestep (as represented in figure 3a). This should allow the autoencoder system to learn to predict upcoming timesteps. Should the predictions then prove accurate enough, these predictions should be able to be used as inputs for subsequent predictions in order to reconstruct any unselected timesteps.

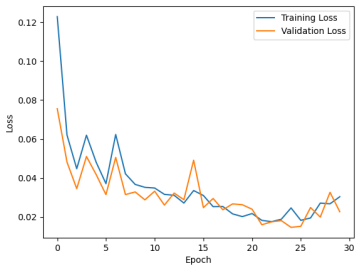
In order to provide a proper baseline for comparison, a second autoencoder was trained. This time using the more standard method of using the inputs as reconstruction targets during training. Outside of this difference, there were no structural changes between the two autoencoders. Figure 3 demonstrates the differences in the training of the two autoencoder variants, showing how the predictive autoencoder was provided future timesteps as targets while the regular autoencoder was given the same timestep for both the input and the target.



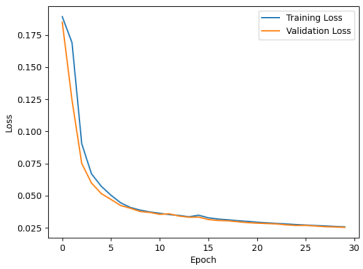
(a) Predictive (b) Regular

Figure 3: The difference in training targets between the regular and predictive autoencoders.

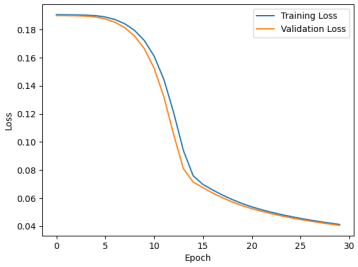
In both cases, the autoencoders were trained for 30 epochs, using a learning rate of 0.001. During development, other learning rates were experimented with, but it was found that the standard learning rate behaved the best. Some examples of the training loss over time at these learning rates can be seen below in figure 4.



(a) Learning rate of 0.01



(b) Learning rate of 0.001



(c) Learning rate of 0.0001

Figure 4: Training and Validation loss over time with various learning rates.

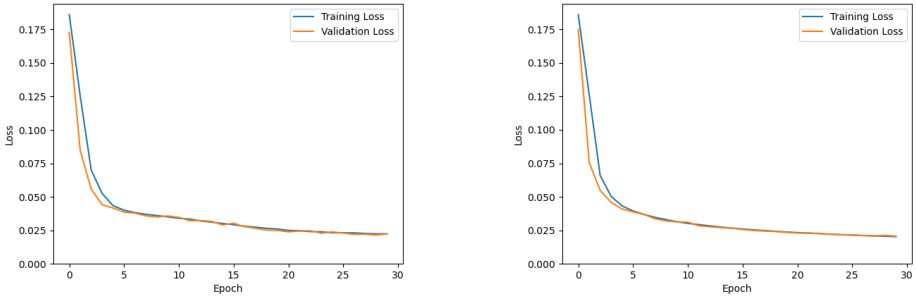
An ablation test was also performed, removing various layers from the model and measuring the performance of each structure. The results of this test can be seen in table 1, where the reconstruction loss of non-anomalous timesteps for different structures are shown. The table shows which layer from the full model is either the only one present, or the only one missing, and what the average reconstruction loss was for that structure. The layers are identified by their filter size, eg. 4, 8 or 16. The average reconstruction losses should be compared to the baseline average reconstruction loss of 0.0209 for the fully-intact system. The table shows that different layers had various effects on performance, and that indeed all 3 layers were required for the best results.

	N = 4	N = 8	N = 16
Model has layer with filters of size N	0.0294	0.0269	0.0234
Model is missing layer with filters of size N	0.0250	0.0244	0.0213

Table 1: Ablation Test Results: Average reconstruction loss of a non-anomalous timestep vs. missing model component

The final training and validation losses of both types of autoencoder were plotted and can be

seen in figures 5 below. This was also done for the autoencoder trained only on laminar datasets, the results of which can be seen in figure 6. Training times varied only slightly between the regular and predictive autoencoders, and the 30 epochs of training took about 3300 seconds on average to complete on the standard data ensemble.



(a) Predictive training and validation loss

(b) Regular training and validation loss

Figure 5: Both regular and predictive autoencoders have similar training and validation losses.

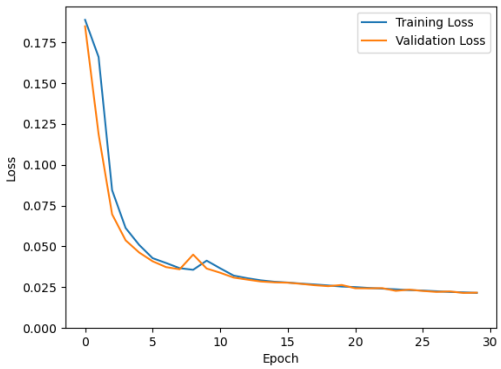


Figure 6: Training and Validation Loss during training of predictive autoencoder trained only on laminar data.

The methods described above were implemented in python 3. The autoencoder model was created using Keras via the Tensorflow library [6][1], and the dataset and ensemble setup and splitting for testing and training was implemented using sklearn [13]. The anomaly detection algorithm itself and rest of the framework code was written by hand[7].

4 Experiment

The following experiments were run on the Rijksuniversiteit Groningen's Hábrók supercomputer machine learning clusters. Hábrók has a number of nodes for machine learning, each of which has 64 cores, 512 GB of memory, and for Nvidia A100 Graphics cards with 40 GB of memory each.

A number of experiments were run to determine the capability of the system. For the first set, both variants of the autoencoder were trained on the ensemble described above, excluding the artificial anomalous datasets. The ensemble was split into a 90/10 train/test split. Once trained on the training data, a random selection datasets was pulled from the non-anomalous data pool, and a smaller subset was pulled from the anomalous pool. Then, for each set of timesteps in each chosen dataset, the autoencoder was run on each timestep in order to represent an active simulation. This was repeated for each type of artificial anomalous data, and the training setup was used for all variations of the experiment, with the only changes being the datasets used.

All of the experiments run here have the threshold pre-set to values specific to the kinds of artificial anomalies being detected. These thresholds can be seen in table 2 below, and were chosen based on histograms compiled on test runs of the training data, an example of which can be seen in figure 7. The feature vector loss and reconstruction loss of each timestep was then plotted, as well as the specific timesteps and resultant autoencoder timestep reconstructions. Additional information was also collected, including the presence of anomalies within the timestep and whether or not the anomalies were detected. These were both also included in the plots alongside the feature vector and reconstruction losses to provide context. The accuracy of the system's detection was also recorded, calculating the true positive and negative rates, and then averaging these values over the course of all timesteps in a dataset, and over each dataset in an ensemble.

Anomaly Type	Reconstruction Loss Threshold	Feature Vector Loss Threshold
Blank	0.1	0.2
Blob	0.03	0.03
Chunk	0.046	0.047
Turbulent	0.03	0.05

Table 2: Threshold Values for Reconstruction and Feature Vector Losses

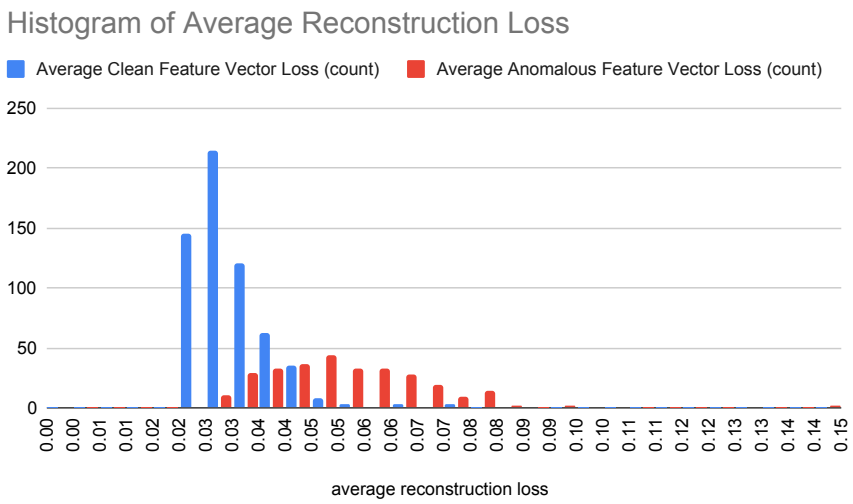


Figure 7: Histogram of Average Feature Vector Loss for non-anomalous (clean) timesteps vs. anomalous timesteps.

Finally, a brief additional set of experiments was run to determine the system's ability to reconstruct missing output data from a selection of saved timesteps. For this, the autoencoders were trained on the full data ensemble, and the primary experiment was run for both types of autoencoders. This was done twice, without any anomalous data, artificial or otherwise. The first experiment run included periodic saving of the simulation output every 10 timesteps, whereas the second saved only the first. The system was then run again in both cases, but instead of using the input dataset, each timestep was provided the output of the previous timestep as the input. The only exception being in cases where the simulation output for that timestep was saved, in which case the saved simulation output was used instead. For both experiments, the average loss of each reconstructed timestep as compared to the expected simulation output across all datasets was then collected, for both autoencoder types. In *all* cases, the first 10 simulation timesteps were saved, so as to effectively skip them for the duration of the experiment. This is due to the unpredictability of the first 10 or so timesteps for which the simulation output is still stabilizing.

The data used was from a 2D Kármán Vortex Street ensemble, depicting the flow of fluid around an obstruction with varying parameters. Parameters changed in the data include flow rate, Reynolds number, and obstruction size. Each run contains 54 timesteps. Generally each pixel contained a value between 0 and 1, and represented the flow rate at that specific pixel. An example of some renderings of this timestep data can be seen in figure 8 below.

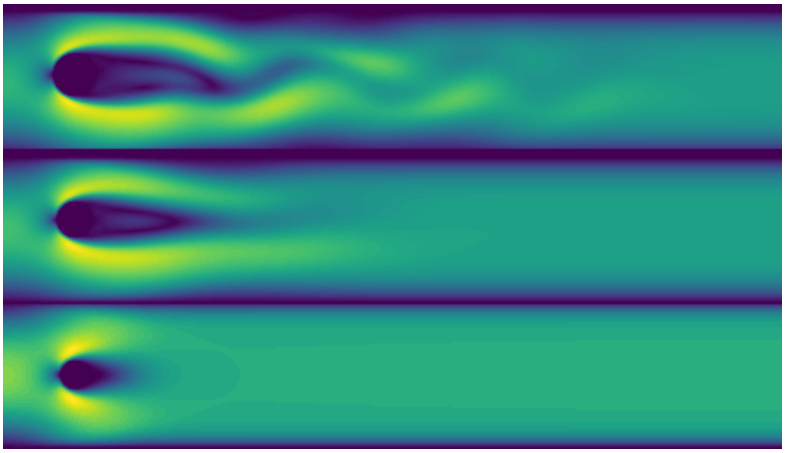
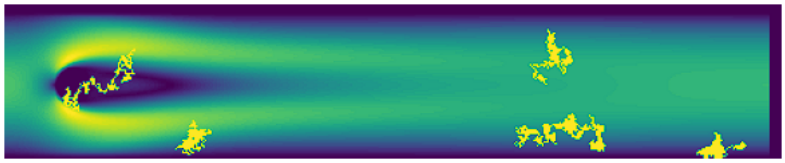
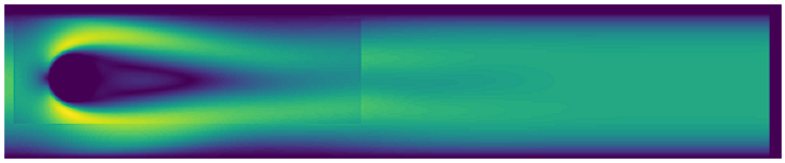


Figure 8: A collection of renderings from different timesteps within the 2D Kármán Vortex Street ensemble

(a) Blank-style anomaly



(b) Blob-style anomaly



(c) Chunk-style anomaly

Figure 9: Examples of the 3 artificial anomalies introduced into the datasets.

To represent anomalous data, a few different methods were used in order to determine the efficacy of the autoencoders at dealing with various kinds of anomalies. In each case , the data was augmented with cloned datasets in which starting at a random timestep, a random number of subsequent timesteps (between 5 and 10) was rendered anomalous through one of three methods. For the first of the artificial anomalies, each selected timestep was "blanked" (each pixel set to a value of 1) in order to produce a noticeably anomalous set of timesteps with a massive divergence from the baseline. For instance, a "blanked" timestep can be seen in figure 9a below. The second means was to select a random pixel somewhere in the timestep, change its value to 1, and then select a random neighbour and repeat the process a pre-determined number of times (in the case of these experiments, 400). Through this method, a blob of pixels is set to 1, and this process was repeated up to 6 times for each anomalous timestep so that 6 such blobs would appear in different shapes, and locations on each anomalous timestep. An example of such an anomalous timestep can be seen in figure 9b below. The final method aimed to produce an anomaly that better matched the style of the data itself. In this case, for each cloned dataset, a second donor dataset was chosen, and a subset (or "chunk") of each timestep matching an anomalous timestep in the cloned dataset was copied from the donor timestep to the clone one. This resulted in timesteps such as the one that can be seen in figure 9c. For the sake of simplicity, these anomalies will be referred to as "blank"-style, "blob"-style and "chunk"-style anomalies respectively. These "Anomalous" timesteps were not used during training, but were introduced during testing to determine the systems' ability to detect anomalies while simultaneously not returning back-to-back timesteps of anomalous outputs.

An additional data ensemble was made that separated the turbulent datasets from the more laminar ones. This final ensemble was not provided with any artificially anomalous data, but treated the turbulent datasets as anomalous instead.

5 Results

5.1 Baseline Results

The recorded data was then plotted, with a sample shown in figure 10 displaying the results at the end of a series of timesteps for the predictive system. Figure 11 then shows the results at the end of a series of timesteps for the same dataset but using a regular autoencoder. These graphs display the reconstruction and feature vector loss over time, with the dotted lines representing the color-corresponding thresholds, and the vertical lines representing points at which anomalous data was detected. In certain cases the outputs of datasets containing anomalies (see figures 12, 13 and 14), there are also portions of the data that contain a raised line. This represents the period of time in which the timesteps are anomalous. Additionally, periodic or otherwise notable timesteps were saved and included in the figures, in line with where they would occur in the time series.

The baseline system output for both autoencoder types show the expected reconstruction and feature vector losses which in both cases hover around 0.025 and near 0 respectively, especially as the simulation progresses. This was the standard across most datasets, only increasing slightly in cases of higher turbulence in the data. This is to be expected given the lack of any meaningful anomalies.

Regardless of autoencoder type, the processing time for each timestep averaged about 0.0111 seconds. The average runtime of an entire dataset was about 1.5620 seconds, though this was within the bounds of the experiment, eg. feeding existing data through the system, and does not accurately represent runtimes for an actual simulation scenario for the same number of timesteps.

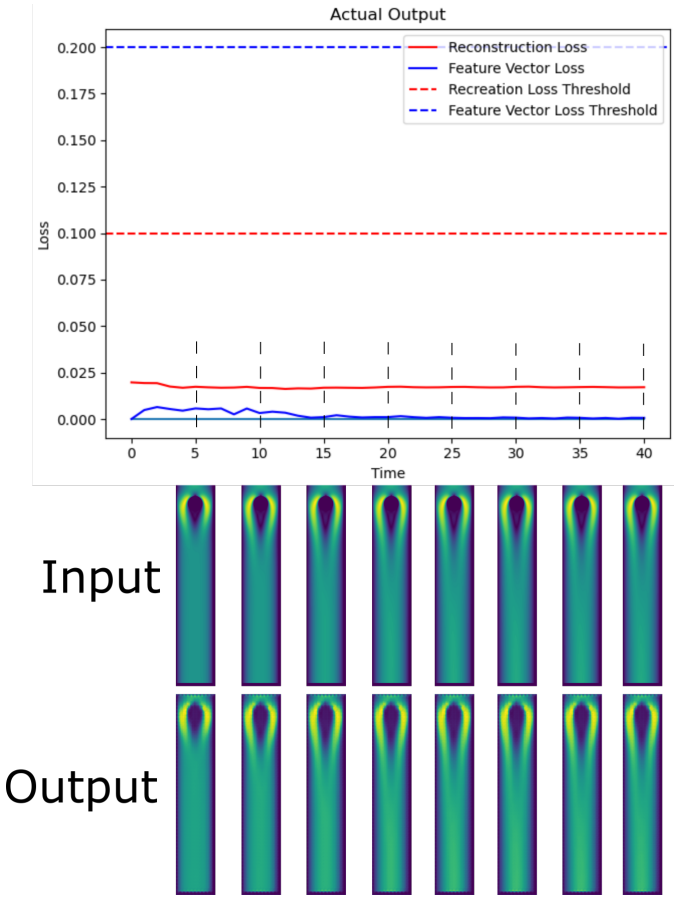


Figure 10: Total system output from a collection of timesteps with no anomalous data using a predictive autoencoder

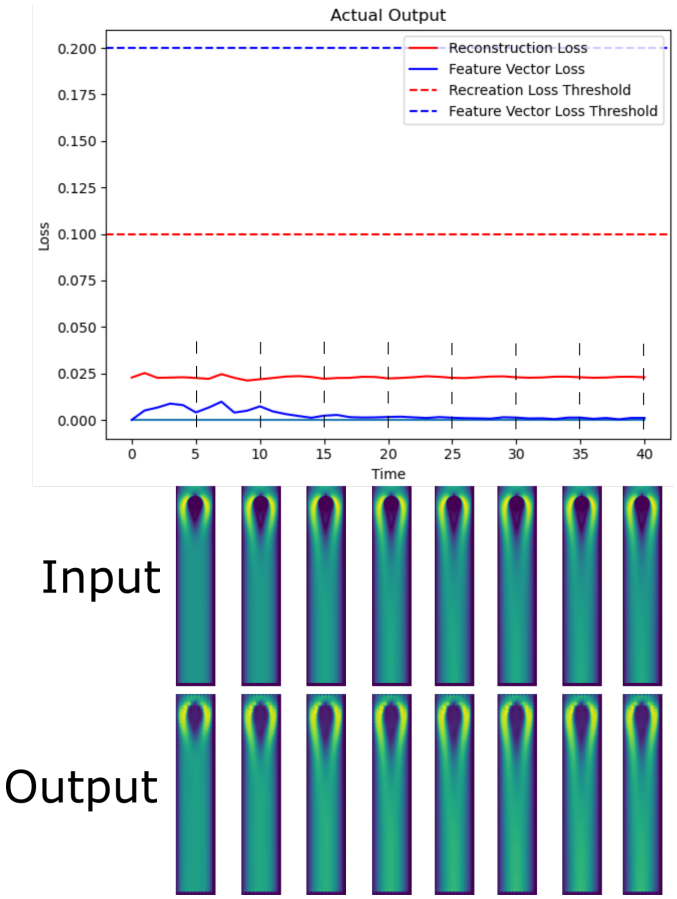


Figure 11: Total system output from a collection of timesteps with no anomalous data using a standard autoencoder

5.2 Anomalous Results

The resultant figures show that each of the artificially anomalous dataset types have very different interactions with the feature vector and reconstruction losses of this system. The blanked anomalies (figure 12), being the most obvious, have the greatest jump in reconstruction loss across the entire anomalous period, and large spikes of feature vector loss at the start and end of the segment of anomalous timesteps. Not only is the change in feature vector and reconstruction loss is evident in the plot, but the resultant outputs as well. Figure 12 clearly shows the output nearly inverting the given input entirely. This massive change is the source of the huge spike in both the feature vector and reconstruction loss.

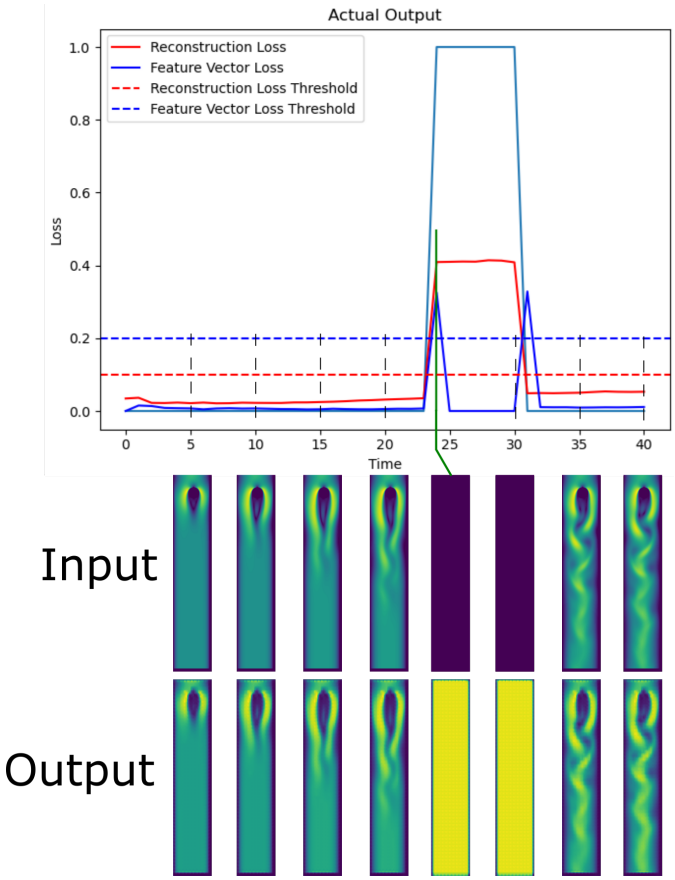


Figure 12: Inputs and outputs of predictive autoencoder encountering "blanked"-style anomalous data.

The blob-style (figure 13) anomalies provide distinct enough increases in both the feature vector and reconstruction loss, but are relatively noisy throughout. This is because each timestep in the anomalous period has a different set of randomly-generated blobs, as can be seen in the autoencoder input/output portion of the figure. These anomalies are still detectable with the system, but not as cleanly as the blanked dataset, though this is to be expected given the nature of the anomalous data. Were these blob-style anomalies to be more consistent from timestep to timestep, it is likely that the change in the feature vector loss would be a lot less noisy.

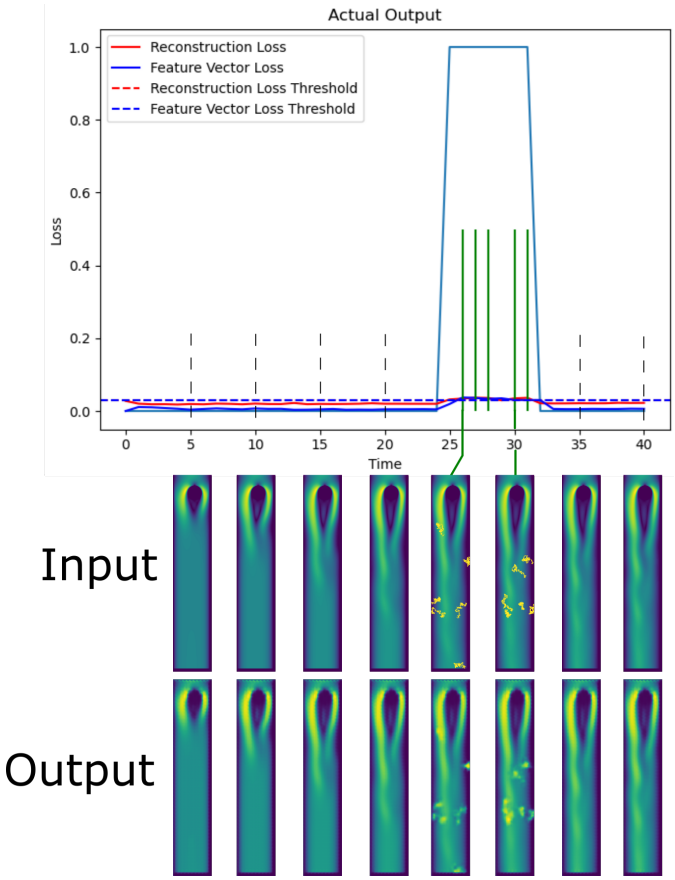


Figure 13: Inputs and outputs of predictive autoencoder encountering "blob"-style anomalous data.

The more "chunk"-style anomalies (figure 14) are detected by reconstruction loss, though due to the much more subtle changes from the regular timestep to the anomalous ones, they are hardly noticed at all by the feature vector loss. This is pretty clearly demonstrated in the inputs and outputs of the autoencoder, where the differences can go unnoticed by a passive observer. In figure 14 only timestep 30's input and output is anomalous, and at a glance it appears to fit in with the rest, only being given away by the lack of turbulence in the upper portions of the image. In this case, change in the reconstruction loss is all that can really be used to attempt to detect the anomalous timesteps.

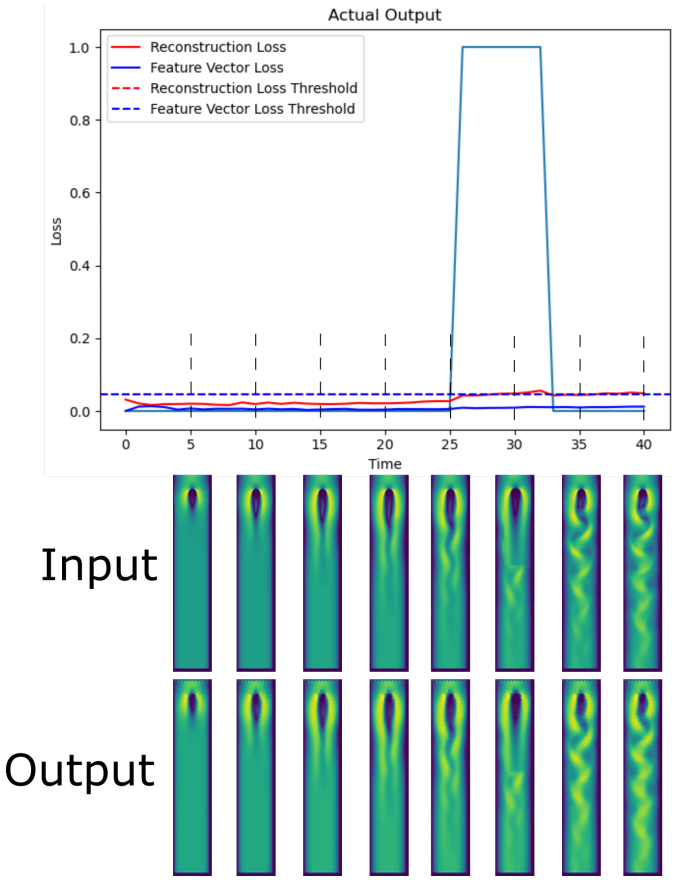


Figure 14: Inputs and outputs of predictive autoencoder encountering "chunk"-style anomalous data.

Figure 15 shows that when trained on laminar data the turbulent data does stand out, displaying a steadily increasing reconstruction loss that certainly becomes more obvious over time. This shows the expected result of the training data having a heavy impact on the actual application. But, demonstrates that anomaly detection is possible when training with only baseline data. That said, the gradual nature of the appearance of turbulence in this dataset is a stark contrast to the sudden appearance of the other types of artificial anomalies discussed earlier. This causes the feature vector loss to remain relatively consistent across most timesteps, as no major change is happening from one to the next. The inputs and outputs show the degrading reconstruction loss pretty effectively, while also demonstrating that no major change in the feature vector loss is occurring for the system to detect.

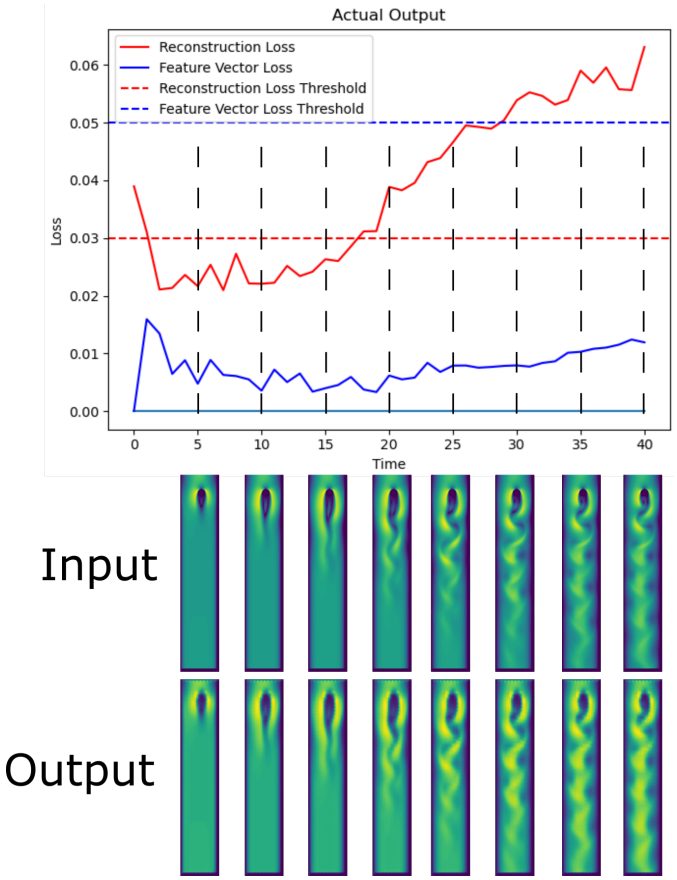


Figure 15: Inputs and outputs of a predictive autoencoder trained on non-turbulent data being presented with turbulence

Table 3 shows that both types of autoencoders had similar true positive and negative detection rates for the reconstruction loss portion of the detector, with both autoencoders easily able to detect the more extreme "Blanked" anomalous timesteps, but then having their accuracy drop off somewhat when detecting the more subtle anomalies. The true positive and negative rates for the turbulent data ensemble are missing from this table due to the setup of the turbulent data. The turbulent dataset timesteps tended towards turbulent or non-turbulent immediately, meaning there was no identifying labels marking a subset of a dataset as anomalous and such labels would have had to be added manually. Additionally, as can be seen in figure 15, no timesteps were generally detected as anomalous due to the low amount of change in the feature vector loss. Because of this, the true positive and true negative rates were not calculated for the turbulent data ensemble, as the results would not be useful.

	Regular		Predictive	
Glitch Type	True Positive Rate	True Negative Rate	True Positive Rate	True Negative Rate
Blank	1.0	1.0	1.0	1.0
Blob	0.7106	0.8512	0.7829	0.8476
Chunk	0.7308	0.9473	0.6015	0.9710

Table 3: Detection true positive and true negative rates for reconstruction loss of artificial anomalous data.

5.3 Reconstruction Results

The reconstruction loss rates collected from the reconstruction tests was averaged across the entire ensemble then plotted, for both the predictive-style of autoencoder as well as the regular style. This was also done for both a case in which the timesteps were saved periodically, as well as a case in which no periodic saving occurred. The graphs plotted can be seen in figures 16 and 17.

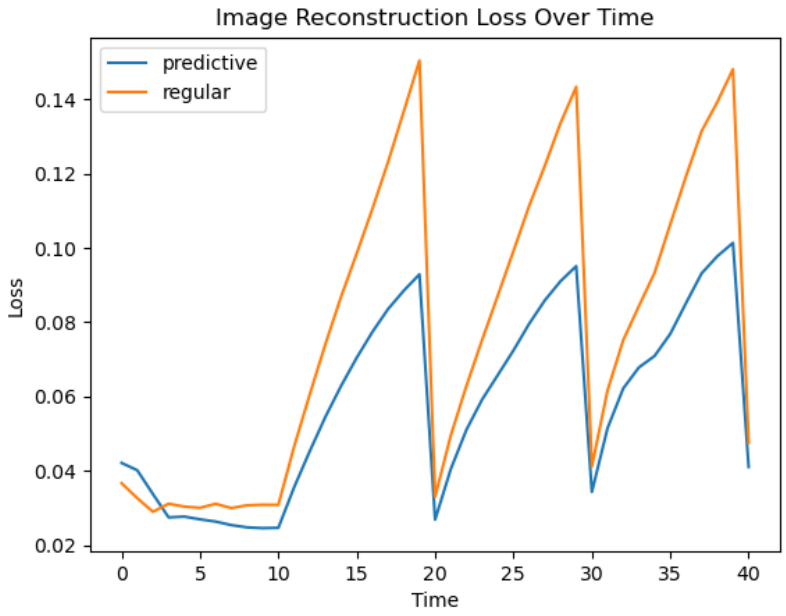
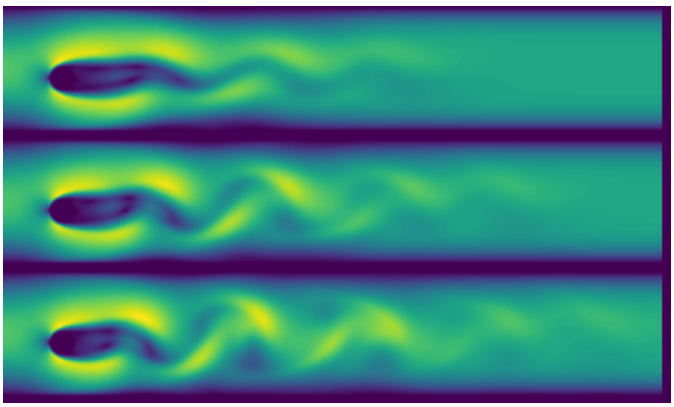


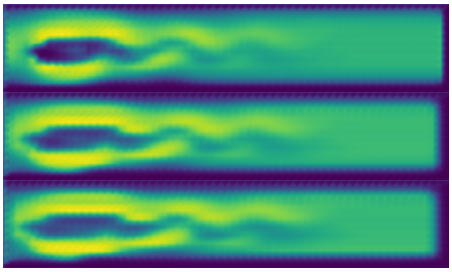
Figure 16: Average loss per timestep of reconstructed data, compared between predictive and regular autoencoders, using periodic timestep saving.

Figure 17: Average loss per timestep of reconstructed data, compared between predictive and regular autoencoders, without periodic timestep saving.

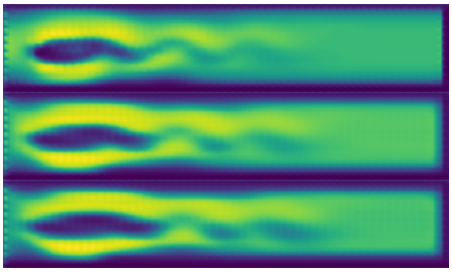
Comparing these plots, it is evident that the predictive autoencoder does a better job of reconstructing missing data. While the two autoencoder types occasionally overlap and perform at similar rates in figure 16, this occurs almost entirely near points where a timestep was saved periodically. Outside of these expected overlaps, the predictive autoencoder consistently outperforms the regular autoencoder in timestep reconstruction. An example of reconstructed data taken from just after a timestep was periodically saved, 5 timesteps after saving, and from right before the next periodic saving can be seen in figure 18c for the predictive autoencoder, and in figure 18b for the regular autoencoder. The original input timesteps for each can be seen in figure 18a.



(a) Initial system input timesteps



(b) Reconstructions generated by a regular autoencoder



(c) Reconstructions generated by a predictive autoencoder

Figure 18: Example of the degradation of reconstructed data over time compared to the original timesteps, for both predictive and regular autoencoders

While the timesteps obviously degrade the more times the are reconstructed from previous reconstructions, it is still evident as to what these images are. However, even though the original timestep is still recognizable, it is especially evident that the regular autoencoder does not really adapt to any changes. It is obvious that it mostly returns the same images over and over, slowly fading out as it degrades. The predictive autoencoder fares better, producing some actual changes, but not any that match the original timesteps too closely, though they maintain a much sharper quality than the regular autoencoder.

6 Discussion

The first thing to note is that there is not too much difference between the predictive and regular autoencoders when it comes to anomaly detection. While the predictive autoencoder style results in outputs with more extreme values, they do not actually differ too much from one another, with preliminary comparisons showing loss differentials of about 0.001 on average. This is shown in the detection rate of the "Blanked"-style anomalies of both autoencoder types, with both having a 100% true positive and true negative detection rate. This is to be expected, as the anomaly type exemplifies the characteristics that are sought by the detection algorithm. The blank timestep acts both as a massive discrepancy between the predicted timestep and the actual timestep, as well as a very significant change in the features that are present within the timestep. This means that both the change in feature vector loss and spike reconstruction loss are both easily detected by the system.

The predictive autoencoder demonstrates a mild improvement in detecting "Blob"-style anomalies, this likely being a result of the predictive autoencoder's tendency to blur hard edges slightly more than its regular counterpart. This results in the blobs blending slightly more into the background data, especially near areas where the data changes rapidly from high to low values. The predictive autoencoder thus generates timesteps slightly closer to the expected non-anomalous baseline in these scenarios instead of the anomalous timesteps that actually appear, allowing for slightly better detection accuracy of said anomalies.

The regular autoencoder shows improved detection capacity for the "chunk"-style of anomaly, as the hard edges of the displaced chunk of another timestep acting as the anomalous portion of the timestep in question are maintained far better by the predictive autoencoder than the regular one. This means the predictive autoencoder does a better job of reconstructing the anomalous data than the regular one, thereby worsening its anomaly detection capacity in comparison.

Outside of the differences between the autoencoder types, figures 12, 13, 14 and 15 demonstrate that the system can be used to find various errors and anomalous timesteps, and only a baseline set of generic outputs needs to be used for training. However, if a specific type of anomaly needs to be detected, then this should be known ahead of time so that the system can be appropriately tuned to detect it. These tests also assume that the timesteps progress in a predictable manner, and as such, though untested, it is likely that the anomaly detection and reconstruction capacity of the system could be limited by less linearly progressive inputs.

The experiments done have also highlighted that there is no major difference in the anomaly detection ability of the predictive and regular style of autoencoders, with both showing similar feature vector and reconstruction loss values. There is a difference between the two when it comes to the reconstruction of missing data, though the difference is less substantial than would be relevant to use in a real setting. In the reconstruction of lost timesteps, as shown in figures 16 and 17, the predictive autoencoder eventually outperforms the regular autoencoder where there are no periodically saved simulation timesteps. In the presence of periodic saving, the predictive autoencoder still functions somewhat better. Although both autoencoders do perform similarly

enough thanks to the periodic saves acting as resets for the reconstruction process.

These results demonstrate that the system is capable on a basic level, as it displays a capacity to detect anomalies and recreate timesteps to a reasonable degree. The similarity between the detection accuracy of the predictive and regular autoencoders indicates that, at least for the anomaly types used, there is little enough difference in anomaly detection capacity. On the other hand, for general timestep reconstruction, the predictive autoencoder does prove to outperform the regular version. This disparity in performance between the two modes of operation, anomaly detection and data reconstruction, imply that it may be advantageous to use separate systems for each, allowing specialized systems to perform anomaly detection and data reconstruction, as any specialized system for reconstruction does not have to worry about any potential bottlenecking constraints imposed by working in-situ with the processing of the simulation that a timestep selection system would.

7 Summary and Conclusion

The results demonstrated above show the restrictions and limitations of the autoencoder system, predictive or not. If set up as described in this set of experiments, the system does not provide a generalist solution to all anomalous data detection, though it can be used to detect deviations from the expected outputs. While not entirely necessary, some foreknowledge of the kinds of errors and anomalies that might be encountered by the system can improve detection accuracy. This is so that detection thresholds can be fine-tuned to provide the best anomaly detection rates. While this foreknowledge can be useful for fine-tuning and best results, it does not help with other, more systematic anomalies such as non-sequential behaviour. These will not be detected by this kind of system without adjustments to the algorithm itself.

The experiments also demonstrate that the predictive and regular autoencoder models behave fairly similarly in terms of detection. Though there is a small amount of variance in each method's success rate for each kind of anomaly, the difference is small enough in most cases as to not be too significant. The more noticeable difference between the two autoencoder variants lies within the system's ability to reconstruct unsaved data from given saved timesteps, in which the predictive autoencoder fares somewhat better. Ultimately, neither performs too well in the long-term. The regular autoencoder simply reproduces the original input timestep over and over, slowly degrading with each reconstruction attempt. Even the predictive autoencoder does not provide a convincing reconstruction of the original data, though the image produced is sharper than its regular counterpart. On less turbulent data, this is much less noticeable, with the autoencoders being able to reproduce the lengthening of the flow without too much trouble. The simplicity of the algorithm being used for reconstruction is a likely cause, given it is the same structure as that used for the anomaly detection rather than its own specialized structure.

Considering what has been demonstrated in this thesis, the autoencoder system as presented did not produce effective enough results for immediate use. As such, future work should look into increasing the complexity of the autoencoder architecture, something that was not done originally due to the long training and runtimes involved. Alternatively, it may be worth investigating other ML frameworks altogether, as a comparison between different kinds of models could provide insights into the efficacy of other kinds of models for this task. It would likely also be useful to investigate developing separate networks for anomaly detection and missing data reconstruction, allowing a simple system to work as the anomaly detector so as to not bottleneck any simulation it's attached to. Meanwhile a more complex system works to recreate missing data when it is required.

As the experiments presented in this thesis were performed with only the given narrow set of artificial anomalies and a specific set of linearly-progressing 2D data, it would be prudent to investigate the system's function with other forms of data and anomalies. A greater variety of simulations could be used as sources for data for future investigations, and this should not be restricted to 2D sources. Other variations of anomalies are also important, some simple examples of which that were not used for these experiments include simple blurring, shifting or scaling, or finding simulations that include known anomalous data. In addition , future investigations

should experiment with non-linear time simulations, eg. those that do not progress in a linear manner. This would include things that occur in non-repeating patterns, periodically reverse, or perhaps are cyclic in nature.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Jinwon An and Sungzoon Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special lecture on IE*, 2(1):1–18, 2015.

[3] C J Battey, Gabrielle C Coffing, and Andrew D Kern. Visualizing population structure with variational autoencoders. *G3 Genes—Genomes—Genetics*, 11(1):jkaa036, 01 2021.

[4] Muhammad S Battikh and Artem A Lenskiy. Latent-insensitive autoencoders for anomaly detection. *Mathematics*, 10(1):112, 2021.

[5] Valentin Bruder, Matthew Larsen, Thomas Ertl, Hank Childs, and Steffen Frey. A hybrid in situ approach for cost efficient image database generation. *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[6] TensorFlow Developers. Tensorflow, August 2023. Specific TensorFlow versions can be found in the "Versions" list on the right side of this page.¡br¿See the full list of authors ¡a href="htt ps://github.com/tensorflow/tensorflow/graphs/contr ibutors"¿on GitHub¡/a¿.

[7] Alpheaus Feltham. Project source code, 2023. [Online]. https://github.com/Emraldis/Thesis$_{P}roject_{C}ode$.

[8] Steffen Frey and Thomas Ertl. Flow-based temporal selection for interactive volume visualization. In *Computer Graphics Forum*, volume 36, pages 153–165. Wiley Online Library, 2017.

[9] Chao Huang, Zehua Yang, Jie Wen, Yong Xu, Qiuping Jiang, Jian Yang, and Yaowei Wang. Self-supervision-augmented deep autoencoder for unsupervised visual anomaly detection. *IEEE Transactions on Cybernetics*, 52(12):13834–13847, 2021.

[10] Xue-Bo Jin, Wen-Tao Gong, Jian-Lei Kong, Yu-Ting Bai, and Ting-Li Su. Pfvae: A planar flow-based variational auto-encoder prediction model for time series data. *Mathematics*, 10(4), 2022.

[11] Yi-Wei Lu, Chia-Yu Hsu, and Kuang-Chieh Huang. An autoencoder gated recurrent unit for remaining useful life prediction. *Processes*, 8(9), 2020.

[12] Olga Lyudchik. Outlier detection using autoencoders. 2016.

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

[14] Gleb Tkachev, Steffen Frey, and Thomas Ertl. Local prediction models for spatiotemporal volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 27(7):3091–3108, 2019.

[15] Xin Tong, Teng-Yok Lee, and Han-Wei Shen. Salient time steps selection from large scale time-varying data sets with dynamic time warping. In *IEEE symposium on large data analysis and visualization (LDAV)*, pages 49–56. IEEE, 2012.

[16] Theodoros Tziolas, Konstantinos Papageorgiou, Theodosios Theodosiou, Elpiniki Papageorgiou, Theofilos Mastos, and Angelos Papadopoulos. Autoencoders for anomaly detection in an industrial multivariate time series dataset. *Engineering Proceedings*, 18(1), 2022.

[17] Dongfang Wang and Jin Gu. Vasc: Dimension reduction and visualization of single-cell rna-seq data by deep variational autoencoder. *Genomics, Proteomics and Bioinformatics*, 16(5):320–331, 2018. Bioinformatics Commons (II).

[18] Yoshiaki Yamaoka, Kengo Hayashi, Naohisa Sakamoto, and Jorji Nonaka. In situ adaptive timestep control and visualization based on the spatio-temporal variations of the simulation results. In *Proceedings of the Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization*, pages 12–16, 2019.