Andrzej Cabala
Eivind Gautvik
Trygve Nerland

Using Machine Learning to Detect Fraud and Predict Time Series

Bachelor's project in Computer Engineering
Supervisor: Nils Tesdal
May 2019

Bachelor's project

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Kunnskap for en bedre verden

Andrzej Cabala
Eivind Gautvik
Trygve Nerland

Using Machine Learning to Detect Fraud and Predict Time Series

Bachelor's project in Computer Engineering
Supervisor: Nils Tesdal
May 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Norwegian University of
Science and Technology

Abstract

The rise of machine learning allows for many new applications in data analysis and prediction. Self-learning algorithms have shown themselves to be capable of outperforming human-made systems in several fields, and have been integrated on a large scale in human computer systems.

In this document, we will first examine the possibility of risk-scoring transactions with respect to the data already sent as part of the transactions, using unsupervised machine learning. Then, we will showcase our attempts at creating a model to predict maintenance data time series with Recurrent Neural Networks such as LSTM and GRU. Our goal is to determine whether the methods are applicable to the data sets, and to create proofs of concept, rather than working systems. Relevant machine learning theory will be introduced, and further work suggested.

Sammendrag

Kraftigere maskinvare og bedre maskinlæringsbibloteker gir nye muligheter for databehandling og analyse. Selv-lærende algoritmer har vist at de er i stand til å utkonkurrere menneskeskapte systmer i flere felt, og har blitt viktige deler av mange systemer.

I denne rapporten skal vi se på mulighetene rundt risikovurdering av transaksjoner med henysn på data som allerede er inkludert innad i transaksjonene ved å bruke maskinlæring. Videre skal vi demonstrere våre forsøk på å utvikle en modell til å forutse driftsdata med nevrale nettverk, som LSTM og GRU. Målet vårt er å bestemme om metodene kan anvendes på datasettene, og å lage et gjennomførbarhetsbevis, istedenfor et komplett system. Vi skal beskrive relevant maskinlæringsteori, og foreslå videre arbeid.

Foreword

We acquired our bachelor task from Signicat. The task seemed interesting and challenging enough for us to improve our machine learning knowledge. The goal of the project is not to develop a product, but rather to create a proof of concept, and therefore we had no use for extensive documentation or any specific software development methodology. We started with what some of us already knew of basic machine learning, and went onward from there.

During the initial weeks we had no data available, and we spent our time reading relevant articles and testing algorithms on generated data. We had no knowledge of the systems, and did not know whether or not we could access them, so we had to rely on help from employees to retrieve the data. Our intention was to prepare ourselves for the production data, by making sure we would have functioning machine learning methods ready. The primary problem was that some of our methods were based on supervised learning, which requires a form of indicator to train. Unfortunately the production data had no such, and we had to change our approach to find a solution.

We had to turn to unsupervised machine learning methods in order to have any chance to classify the data. Yet again, the lack of any fraud indicators made it impossible to be certain about the correctness of our results. As a result we had to modify our task to something more achievable. We agreed with the task giver to instead attempt to predict performance metrics using machine learning. The reason why we chose this instead, is because all the necessary data is already available from within Signicat, and the results are therefore verifiable. We quickly acquired the necessary access, although much of the project time had already passed by then. This new task allowed us to create a simple demonstration of how machine learning applies to the time series. This is also the reason the report is split in two parts. The first part details our progress with the original task, while the second describes the task we spent most of the semester on, the time series prediction.

Acknowledgements

Special thanks to Magnus Selvaag for helping us interfacing with Signicat's systems, and suggesting courses of action. Additional thanks to Signicat and Lars Klemetsaune for providing us with computers, a work space and a bachelor thesis task. We also wish to thank Nils Tesdal for guidance and wisdom throughout the project.

Contents

1 Introduction **7**
1.1 Task and Objective . . . 7
1.1.1 Original Task description . . . 7
1.2 Software Base . . . 8

2 Theory Part 1 **9**
2.1 Scaling Data . . . 9
2.1.1 Min-Max Scaler . . . 9
2.1.2 Standard Scaler . . . 10
2.1.3 Robust Scaler . . . 10
2.1.4 Normalization . . . 10
2.2 Clustering . . . 10
2.2.1 Distance Calculation . . . 10
2.2.2 Categories of Clustering Algorithms . . . 10
2.3 Association Rule . . . 12
2.4 Dimension Reduction . . . 13
2.4.1 Principal Component Analysis (PCA) . . . 14
2.4.2 Independent Component Analysis (ICA) . . . 14
2.4.3 t-distributed Stochastic Neighbor Embedding (t-SNE) . . . 14
2.4.4 Isometric Mapping (ISOMAP) . . . 14
2.4.5 Uniform Manifold Approximation and Projection (UMAP) . . . 14
2.5 Outlier Detection . . . 14
2.5.1 Methods for Unsupervised Outlier Detection . . . 15

3 Method Part 1 **16**
3.1 Initial Work . . . 16
3.2 After Receiving Datasets . . . 16
3.3 Strategy . . . 17
3.4 Data . . . 17
3.4.1 Production Authentication data . . . 17
3.4.2 BankID Data . . . 18
3.5 Clustering . . . 18
3.6 Association Rules . . . 18
3.7 Outlier Detection . . . 19
3.8 Choices . . . 19
3.8.1 Scaling . . . 19
3.8.2 Dimension Reduction . . . 19
3.8.3 Outlier Detection . . . 20
3.8.4 Difficulties . . . 21

4 Analysis Part 1 **22**

5 Discussion Part 1 **25**
5.1 Reliability 25
5.2 Result 25

6 Conclusion Part 1 **27**
6.1 Further Work 27

7 Introduction Part 2: Time series **28**
7.1 Thesis Problem 28

8 Theory Part 2 **29**
8.1 Time Series 29
8.1.1 Forecasting 29
8.2 Neural Networks 30
8.2.1 RNN 31
8.2.2 LSTM 31
8.2.3 GRU 32
8.2.4 Optimizers 32
8.3 Regression 33
8.3.1 Linear Regression 33
8.4 Decision Tree 34
8.4.1 Random Forest 34
8.5 SVM 35
8.6 Common Challenges in Machine Learning 35
8.6.1 Datasets 35
8.6.2 Run Time 36
8.6.3 Overfitting and Underfitting 36
8.6.4 Exploding/Vanishing Gradient 36
8.7 Training, Validation and Test Sets 36
8.7.1 Train-Test Split 36

9 Method Part 2 **39**
9.1 Datasets 40
9.1.1 Pre-processing and Scaling 41
9.2 Naive Approach 41
9.3 Feature Selection 41
9.4 Scikit-learn Regression Methods 42
9.5 Model Optimization 42
9.5.1 Bayesian Optimization 42
9.5.2 Early Stopping 43
9.6 Comparability of Results 43

10 Analysis Part 2 **44**
10.1 Talos 44
10.2 Comparison 45
10.3 Final Model 47

11 Discussion Part 2 **49**

12 Conclusion Part 2 **51**
12.1 Further Work 51

List of Figures **52**

List of Tables **54**

Bibliography **55**

A Talos Run 1 **60**

B Talos Run 2 **69**
B.1 Sorted by validation loss 69
B.2 Sorted by test loss 71

Abbreviations

DBSCAN	Density-Based Spatial Clustering of Applications with Noise
ELU	Exponential Linear Unit
GeoIP	Geographical information based on an IP address
GRU	Gated Recurrent Unit
HDBSCAN	Hierarchical DBSCAN
ICA	Independent Component Analysis
ISOMAP	Isometric Mapping
LARS	Least-angle regression
LASSO	least absolute shrinkage and selection operator
LSTM	Long Short-Term Memory
MIMO	Multi-Input Multi-Output
ML	Machine Learning
MSE	Mean Squared Error
NN	Neural Network
PCA	Principal Component Analysis
ReLU	Rectified Linear Unit
RNN	Recurrrent Neural Network
SVM	Support Vector Machine
t-SNE	t-distributed Stochastic Neighbor Embedding
UMAP	Uniform Manifold Approximation and Projection

Chapter 1

Introduction

In our increasingly digitized world where thousands of users access services that handle sensitive information at any time, there are always new threats to our security and privacy. While new security measures are constantly being created and broken, could machine learning provide us with some stability?

Signicat is responsible for maintaining and developing authentication services for many other services which require high security. Our goal is to explore whether machine learning can be used to estimate the chances of authentication fraud within these services, and complement existing methods used to detect suspicious user activity. This would allow Signicat to offer better fraud detection services to their customers, by helping to prevent misuse of their systems.

In this report, we will first describe some relevant theory, and then detail the path we took in order to reach a solution. Finally, we will reflect on the results and their reliability.

1.1 Task and Objective

Our goal is to determine whether machine learning algorithms can be used to supplement and improve the services used to detect fraud and abuse provided by Signicat. Should that be the case, we are to create a proof of concept for a "fraud risk score" to aid those services.

1.1.1 Original Task description

Signicat has for several years delivered solutions for strong authentication and signing in Scandinavia, and is now about to establish solutions for the rest of Europe. In addition to being an ID hub Signicat wants to deliver value added services on top as registry lookups and similar. Signicat collects huge amounts of data about a person's use of electronic ID and signing and wants to use these data in order to avoid fraud and abuse. The solution need to comply with GDPR regulation and the end-user's need for confidentiality protection.

The students should create a proof of concept that covers the following aspects:

- *Find and normalize data from different ID providers*
- *Anonymize data when required*

- *Use AI/ML for finding anomalies in behaviour as input for fraud risk score*
- *Create a fraud risk score based on the findings*

1.2 Software Base

We used the Python programming language[57] version 3.6.7 in this project. The major software libraries are listed below:

Library	Version	Description
Keras [13]	2.2.4-tf	High-level neural networks API. Running on top of Tensorflow.
Tensorflow [3]	1.12.0	Open source machine learning library for research and production.
scikit-learn [43]	0.20.3	Simple and efficient tools for data mining and data analysis.
Pandas [27]	0.24.2	High-performance, easy-to-use data structures and data analysis.
Talos [6]	0.4.9	Hyperparameter Optimization for Keras Models.
Matplotlib [24]	3.0.2	Plotting library for the Python programming language.

Chapter 2

Theory Part 1

Machine learning is a term that encompasses a wide variety of methods, where certain parameters of algorithms are approximated without human intervention [5]. While social media might portray these algorithms as capable of solving any problem, there are still many and great limitations to what one can achieve using these techniques.

We will now describe two types of machine learning; supervised and unsupervised. In order to apply supervised machine learning, large sets of labeled examples are required, from which an algorithm can learn to recognize which patterns lead to which results. By labeled is means that the dataset includes a truth table. [5]

An unsupervised algorithm on the other hand, does not require the examples to be labeled. These types of algorithms focus on recognizing general patterns within data, allowing to group or separate the data according to some formula. The two types of machine learning algorithms have different use cases, and not every problem can be solved by either or both. [5]

2.1 Scaling Data

We will briefly describe the scaling methods considered during his project. Below is a quote from [8, p.19] explaining the importance of data scaling:

> "Normalization (scaling) of data within a uniform range (e.g., 0–1) is essential (i) to prevent larger numbers from overriding smaller ones, and (ii) to prevent premature saturation of hidden nodes, which impedes the learning process. This is especially true when actual input data take large values. There is no one standard procedure for normalizing inputs and outputs."

These scalers can be set to scale each feature of the input data independently, so that every feature will have the same resulting range. In other words, every feature will have the same minimum and maximum value. All the scalers used in this project are from the scikit-learn library.

2.1.1 Min-Max Scaler

With this technique the original data range will be scaled linearly to fit a defined new range, for example $[0, 1]$. The relationship between the scaled data points will be the same as the unscaled data. [42]

2.1.2 Standard Scaler

Standard Scaler, also known as Z-Score normalization, is a non-linear Scaler. The data points are scaled by taking the mean value as the zero value, and then scaling the rest to a Gaussian distribution accordingly, which results in values in the range $[-1, 1]$. [42]

2.1.3 Robust Scaler

While the Standard Scaler can behave in unintentional ways if the dataset contains outliers, the Robust Scaler utilizes more resistant statistics to scale the data well despite the outliers.[48]

2.1.4 Normalization

One easy way to scale the data is to use norms, for instance the l1 or l2 norm. The data is scaled by dividing by the norm, resulting in normalized values. Additional information on normalization can be found at[63].

2.2 Clustering

Clustering is one of the most common types of unsupervised learning, where the idea is to group similar objects into as uniform groups as possible. These groups, also known as "clusters", should then have some underlying properties in common. This is commonly used to create an algorithm to classify objects into categories, so that new unknown objects can be classified into a category based on the clusters found earlier. It is not uncommon for real life datasets to have high dimensional data, and therefore one has to consider dimension reduction in order to visualize or reduce run time.[10]

2.2.1 Distance Calculation

All clustering algorithms are dependent on distance measurement to determine the dissimilarity between two data points. The we used the Euclidean distance because it is easy to understand and was the default distance measure for scikit-learn. The Euclidean distance is the length of a straight line drawn between two points in a space. Another term for this is the L_2 distance.[35]

$$d(p, q) = d(q, p) = \sqrt{\sum_{i=1}^{n}(q_n - p_n)^2} \tag{2.1}$$

2.2.2 Categories of Clustering Algorithms

Distribution Based

Distribution based clustering algorithms employ statistical models such as Gaussian distribution. The clusters are defined by the probability of objects belonging to the same distribution. [64]

Expectation-Maximization (EM) with Gaussian Mixture Models (GMM) is an example of distribution based clustering algorithm. Provided that the data points have a Gaussian distribution, we can use the covariance and mean to find GMM. The clusters are determined by the mean and standard deviation of the parameter, and EM is used to find the most optimal values for each cluster. The method can be described with these steps: We begin by choosing a number of clusters and initial parameters. Data points are then distributed according to computed probabilities for cluster connection. New parameters are calculated based on the probabilities, and the placing of data points and calculation of parameters continues until convergence. [64]

Centroid Based

A centroid based algorithm uses a given number of center points that the clusters will be based around. This center points will get moved around by the algorithm in an attempt to reduce the loss. The loss can be calculated in a number of ways, the most common loss function is to calculate the Euclidean distance between each point and closest centroid. This is repeated until the centroids stop moving. The most common algorithm for centroid based algorithms is K-Means. Every data point has the closest cluster centroid assigned to them. [21]

Connectivity Based

Connectivity based or hierarchical clustering can be seen as a binary tree, where every node is a cluster with two subclusters as children, data points as leaf nodes and the whole collection of data points as a root node.

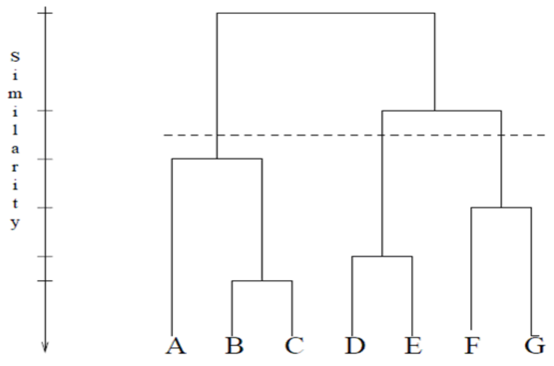


Figure 2.1: *The output from hierarchical clustering is a tree diagram called dendrogram, which shows relationships based on similarity. [60]*

We can divide connectivity based clustering into two types: Agglomerative clustering and divisive clustering. In agglormerative clustering we consider all the data points as individual clusters. Clusters will then be paired based on a proximity calculation (e.g Euclidean distance). The pairing continues until all clusters are merged together into one "root node" cluster. Divisive clustering is rarely used, and its execution is the opposite of agglomerative. It begins with the data points as a single cluster and divides the clusters by two and two until every data point has its own cluster. [12, 64]

Density Based

Density based clustering turns high density areas into clusters, which means that this type of clustering can be used to find noise or outliers. The clusters are then separated from each other by low density areas. There are two parameters that will determine the result from this type of clustering, and these are epsilon and minimum points, which are used to dictate core points. A core point is a data point, which satisfies the requirement of having minimum points inside its epsilon or radius reach. If a data point is connected to a core point, but has less than minimum points required inside its reach circle, we call it a border point. A cluster is shaped of core points and border points connected to each other directly or indirectly through other points. Noise points or outliers are points, which are not connected to any core point and not fulfilling the core point conditions. [17]

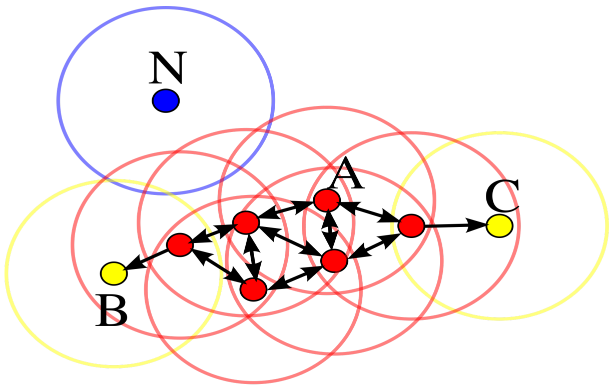


Figure 2.2: *Density based clustering with 3 minimum points. Blue = noise, yellow = border and red = core [61]*

The most common density based clustering algorithm is DBSCAN (Density-Based Spatial Clustering of Applications with Noise), and it uses the parameters mentioned above: minimum points and epsilon. The epsilon should be picked wisely based on the overall distance in the data frame. A large amount of noise indicates a too low epsilon value, and a cluster with the majority of data points indicates otherwise. The minimum points required can be adjusted based on the quantity of data points. The algorithm iterates through the points and decides whether the point is a core point or not. If the point is a independent core point it will become a new cluster, but if the point is connected with a cluster it will be assigned to it. The algorithm is finished when all points are visited. [18]

2.3 Association Rule

Association rule or co-occurrence grouping is a type of unsupervised learning which finds patterns in datasets. The method is often used in market analysis to find which items are often purchased together, but it is also useful in other situations since it finds frequent combinations of features and correlations between them.

Support, confidence and lift are commonly used terms in association rule mining. To describe the terms we have used the words antecedent and consequent. Antecedent is a value or values causing the consequent to appear.

Support is the frequency of an item or items occurring together, and we can not tell whether a rule is a coincidence or not without a high support value.

$$Support(antecedent) = \frac{number\ of\ data\ points\ containing\ antecedent}{number\ of\ data\ points}$$
$$Support(antecedent => consequent) = \frac{number\ of\ data\ points\ containing\ both\ antecedent\ and\ consequent}{number\ of\ data\ points}$$

Confidence is the probability that a consequent occurs with given antecedents. This measure tells us how useful information the association rule provides.

$$Confidence(antecedent => consequent) = \frac{Support(antecedent=>consequent)}{Support(antecedent)}$$

Lift is an indicator of how strong the correlation between the consequent and the antecedents is. A high Confidence value does not necessary mean an interesting association rule, because the consequent can be an common item with a high support value located in almost every data point. To check if the confidence is based on popularity or a strong correlation we could divide the value by the support value of the consequent to get the lift value. A good minimum for the lift would be 1.0. [19, 55, 53]

$$Lift(antecedent => consequent) = \frac{Support(antecedent=>consequent)}{Support(antecedent)*Support(consequent)}$$

The most commonly used association rule algorithm is the Apriori algorithm. The algorithm is composed of the following steps [40]:

1. Set the minimum requirements for metrics such as support.

2. Create a frequency table for every item set with length n, and remove those which do not meet the requirements.

3. Join the frequent item sets from step 2 to create a frequency table for item sets with length n+1, and again remove those which do not meet the requirements.

4. Repeat steps 2 and 3 until no more combinations can be created.

2.4 Dimension Reduction

Datasets often have many features, which can increase the complexity. Dimension reduction is used in order to reduce this complexity, by using various methods to reformat the high-dimensional space into one with lower-dimensional space. The resulting space should then have more relevant features/dimensions, which then allow machine learning algorithms to learn the important connections faster. This can especially help with visualization of the dataset, given that the higher dimensions are not as easy to process for the human mind. A common use case is visualizing the result of a clustering algorithm. The methods have varying degrees of structure preservation, usually either focusing on maintaining the local structure, or the global patterns of the data.[20]

Below are the Dimension reduction methods used in this project:

2.4.1 Principal Component Analysis (PCA)

This method creates "new" features by using orthogonal linear combinations of the original features. The new features will be ordered most relevant to least relevant, using the variance of the features, making it easy to further eliminate more features. [20]

2.4.2 Independent Component Analysis (ICA)

The goal of ICA is to create dimensions which are as independent of each other as possible. ICA is more commonly used for separating mixed signals to retrieve the originals, though it can also be used for dimension reduction. [25]

2.4.3 t-distributed Stochastic Neighbor Embedding (t-SNE)

A method based on statistics. It is sensitive to the local structure of the data, so the global structure becomes less preserved. One of the problems with this method is that it costs large amounts of processing power, and is therefore not well suited for larger datasets. [15]

2.4.4 Isometric Mapping (ISOMAP)

This method consists of several steps. First it searches for neighbours, and then uses an algorithm, commonly Dijkstra's algorithm, to find the shortest paths. Finally, it utilizes partial eigenvalue decomposition. [7]

2.4.5 Uniform Manifold Approximation and Projection (UMAP)

UMAP is a recent method similar to t-SNE, but is noticeably faster and scales more efficiently for high-dimensional data. It also preserves global structure better in most cases. We are using the UMAP implementation in the paper cited here. [39, 38]

2.5 Outlier Detection

Machine learning can be used in many different ways. When our dataset have data points that lie too far away from the rest, it can sometimes be a good idea to remove these values in order to improve the accuracy of our machine learning models. The methods used to achieve this goal can be grouped under the term "outlier detection". While similar to "anomaly detection", the former assume that the training data have the outliers already contained within them, and seeks to locate them. The latter assumes the data is clean of any strange data points, and then evaluates whether new data points fit in with the previous data. [41] The results of these methods might be similar to that of clustering, but the goal is different. These methods try to find which regions have the highest density of data points, and evaluates which points are too far away from these regions. Therefore, we gain two "clusters", or groups; one for the points that are inliers, meaning the ones that were accepted, and the outliers, which were deemed too far away. [31, 4]

2.5.1 Methods for Unsupervised Outlier Detection

The methods described below are the same methods compared in [2].

Robust Covariance

This method assumes the data has a normal distribution, and attempts to draw an ellipse around the data points to decide which points to consider inliers. [45]

One-Class SVM

One-Class SVM, also known as Single-class SVM, is a SVM with only one class, the normal class. Instead of separating two classes with a hyperplane, One-class SVM marks an area in multidimensional space witch where everything within will become a part of the normal class, and everything outside will be an outlier. SVM is further detailed at 8.5 on page 35. As One-class SVM is sensitive to outlying points, it often does not produce good results for detecting outliers. [51]

Isolation Forest

The algorithm selects a feature, and chooses a random value to split at for the feature, to then separate the values by this division. This process is recursive, and therefore the process of isolating one single point is presentable by a tree data structure. Points which are separated quickly on average are considered to be more likely to be outliers, as these points are more susceptible to isolation due to their low numbers and difference from the rest. [36]

Local Outlier Factor

The Local Outlier Factor is a value that shows how divergent a point is from local clusters of points. The main difference here is that the inliers are not gathered in one big area in the center, but instead in smaller clusters surrounded by outliers near those clusters. In other words, this method focuses more on the local divergence, rather than the global density. [37]

Chapter 3

Method Part 1

While working on the task, we were located in Signicat's office, which allowed us to easily contact the employees when we needed help with technical issues. It was also necessary to access internal resources, especially data representations. We were provided with computers, and an area where we could work. In the beginning, we had to wait a few weeks for Signicat to acquire the data we needed.

3.1 Initial Work

As the year begun we wished to start working on the project as soon as possible, and distribute the work evenly throughout semester. Unfortunately, it was difficult to do anything meaningful at the time, as we had to wait until someone would grant us access to the necessary data. This was not something we could acquire ourselves due to security and privacy concerns. As this took an unexpected amount of weeks, we decided that we must get something else than reading theory done in the meantime, so we decided to work on setting up the project, and to create a random self-generated dataset for the time being. We did not know the structure of the dataset we would receive, or what would be in the dataset. Still, we set up some basic machine learning methods, such as supervised learning and clustering. The supervised learning later turned out to be useless, as the data did not have any columns that stated whether a data point is fraudulent. There were some ideas from employees about using the alerts that companies use to determine suspicious activity for this purpose, though it did not lead anywhere. This is also the reason we did not detail the supervised learning further here.

3.2 After Receiving Datasets

After a few weeks, we were given some simple anonymous transaction data. This data is further described later in this document. Initially, we aimed to cluster and classify the data, using appropriate algorithms combined with dimension reduction. The idea was to observe whether any interesting patterns would emerge within the clusters. The clustering is not particularly interesting here. While we could not find fraud in this dataset, we hoped that would change. Alas, our inquiries were in vain, as we were later notified this data would not be retrievable. As this did not lead anywhere, we searched for other solutions, while considering modifying the task to something more appropriate. We discovered outlier/anomaly detection and saw that this would be fitting for the task, as it is an unsupervised method, it does not require labels to train. Implementing this allowed us to at least determine how divergent points are. We then experimented with a few setups with different methods, dimension reduction and scaling.

3.3 Strategy

Classification is rather impossible to accomplish, as having labels for the training data is essential to train a classifier, and we have no way of knowing which data points are fraudulent.

Initially we tried clustering the data, to see how they are distributed. This allows us to get a early idea about what patterns are prevalent throughout the dataset. Naturally as we change datasets these patterns would change, so running clustering is still a viable option for visualization. This left us with the option of outlier detection. The idea is to see whether any data points stand out from the others, and which might have a higher risk of being fraudulent.

3.4 Data

When we started to work on the project, we had no access to data. Therefore we first made a program to generate example data for the machine learning algorithms, so that we could set up our project more easily before we got access to real data.

3.4.1 Production Authentication data

Later in the project we received access to production data containing the following variables:

- User-Agent - browser, operating system, language settings
- GeoIP data - longitude, latitude, continent, country, region and city, timezone
- Metadata - Timestamp, Duration, Session ID, Transaction ID, Service, State, Method
- IP address based identifier.

The IP addresses were each hashed and converted to an unique index before we gained access to them. This is because the anonymity of the users has to be maintained. In addition several fields were duplicates or unique identifiers, which means they can be removed from the dataset to improve the learning rates and reduce memory usage. Features with very high correlation are also not very valuable, as they contain very similar information, and should therefore be reduced before training. Some operations included multiple transactions and where combined based on session ID. We should expect fraudulent transactions to be quite rare compared to legitimate transactions, and must therefore use large amounts of data in order for them to be represented. The dataset contained 115130 transactions. In some examples we use a reduced dataset due to run time and to demonstrate various concepts, though our results use the complete set.

3.4.2 BankID Data

We have evaluated the possibility of using BankID fraud risk indicators as a “truth-table” for supervised learning and as a metric for unsupervised learning. A fraud risk indicator is a flag that has been triggered for a transaction that could indicate suspicious activity. The idea was to use these as a starting point to train algorithms with supervised learning, and then determine whether such an approach results in better accuracy than the tests in use today. We where later informed that Signicat does not have access to the data and it would be unrealistic to acquire any such. The information is stored by the customers, and gathering it would require co-operating with them, as well as more time than we can afford during this project.

3.5 Clustering

We have experimented with different clustering algorithms, among them KMeans, Birch, DBSCAN, MeanShift, AgglomerativeClustering. Since Birch and Meanshift were not mentioned earlier, it can be of importance to say that Birch is a hierarchical clustering type, and Meanshift is centroid based. In addition we also used HDBSCAN, which is a hybrid between hierarchical clustering and DBSCAN. The idea was to experiment and look for small clusters and outliers which are isolated from the rest.

3.6 Association Rules

Association rule is an unsupervised method, which works well with datasets containing structured discrete data [34]. Since the majority of the transaction data contains discrete values, we decided that an application of this method could be helpful in our problem. We have used association rules to find relationships between features in the datasets, giving us a better overview of the correlations between them. These can be used to find good candidates for feature reduction. This method can also used in combination with clustering, because it gives information about typical cluster characteristics. One can then get an general understanding of which features' values separate the clusters from each other.

The Apriori algorithm is good at finding common patterns with high occurrence, which does not fit well with our task. Since we expect the fraudulent points to be extremely rare, it would be difficult to find anomalies by using Apriori. This is due to the association rules receiving low support when attempting to cover fraudulent data points. However, we thought of two ways this algorithm could provide us with useful information. Both solutions utilize a combination of outlier detection and Apriori. The first solution is to label inliers using outlier detection, and to use Apriori to find rules with this label as a consequent. We can then get information about what defines a inlier. The other solution is to separate outliers from inliers and use the algorithm on the data labeled as outliers to get insight in common outlier patterns.

3.7 Outlier Detection

In our case, outlier detection can help us find which data points in our datasets are the most unlike the others. In other words, we can get a good idea of which points are suspicious. Many of the methods within scikit-learn use a value known as "contamination" to estimate how much of the data should be outliers. As we have no statistics available to estimate this, we can use an incrementing value for the contamination to see at which value the points become selected as outliers. To visualize this, we convert the contamination value to colour values, as shown in figure 3.1. This way we get a "score" for each point, using any of the methods described earlier except One-Class SVM. Note that Local Outlier Factor has a limit for the contamination value of 0.5, while the others can go up to 1. This means that the distribution of the score will inevitably be different for this method.

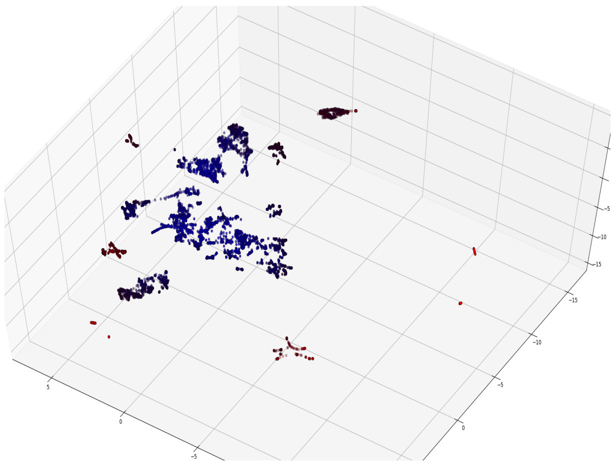


Figure 3.1: *Color values represent contamination values. Blue points are closer to the norm, while red indicates high divergence from the rest. The dark-coloured points are somewhere in between.*

3.8 Choices

We had to choose between the methods available for dimension reduction, outlier detection and scaling functions.

3.8.1 Scaling

We chose to use the robust scaler for our project as it is not sensitive to outliers, leaving other outliers easier to detect after scaling. Other scalers, such as min-max scaler or standard scaler, are much more sensitive to outliers because of their linear nature, and therefore less useful in our case.

3.8.2 Dimension Reduction

While applying all the methods mentioned in theory above, we focused on UMAP and PCA for the unsupervised learning. PCA was useful in the beginning when testing out how other functions affect the result because it is quite fast, and it allows us to see to some degree which features are important for the result (see Figure 4.1). For the most important tests we used UMAP, because of the way it preserves both the local and global structure. The results from UMAP were also the most visually intuitive, which is important when it comes to illustrations and understanding. ISOMAP causes high memory usage, and t-SNE is much too slow for this practical application. On a dataset with over 115 000 elements, the run time is above 3 hours. Therefore we do not include ISOMAP and t-SNE as options in this case.

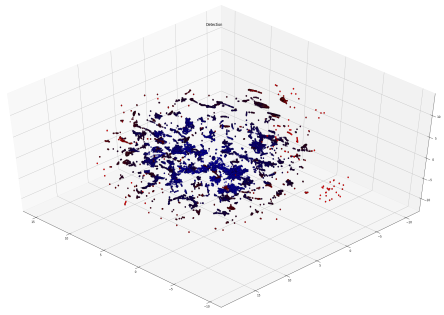


Figure 3.2: *dataset reduced with UMAP. Points more or less gather in clusters.*

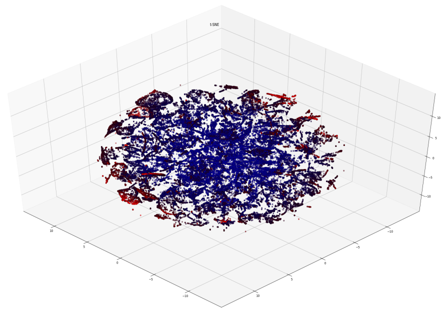


Figure 3.3: *The same dataset reduced with t-SNE, here the data points lose much more of the global structure, resulting in a less clear cloud of points.*

3.8.3 Outlier Detection

As outlier detection is best suited to the situation, we have considered various methods of detecting outliers. We focused on the methods illustrated in [2]. It turns out that the results from Robust Covariance and Isolation Forest are very similar, and therefore we focus on only Isolation Forest from this point. Although UMAP gives a slightly different distribution every time, both the results are quite similar. Note that applying the methods can produce slightly different results every time, and sometimes the same distribution is merely rotated while maintaining the same structure. While we had no numerical metric to measure how good any given result is, visually observing the result is a good approximation when checking whether dense clusters of points count as inliers. See figure 3.4 and 3.5.

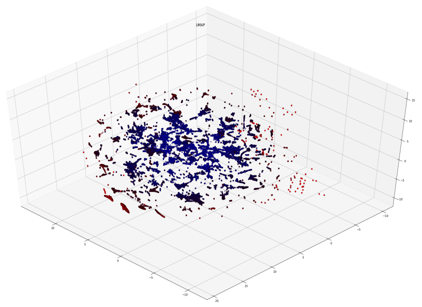


Figure 3.4: *Robust Covariance*

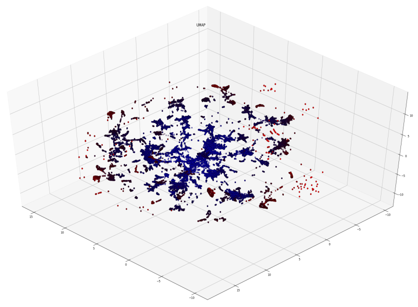


Figure 3.5: *Isolation Forest*

The One-Class SVM implementation in scikit-learn does not use a contamination value, as it is more suited for novelty detection, which is not what we are dealing with, and therefore we do not consider it a meaningful option. Still, figure 3.6 shows an example of how it divides the data. The division here is much more chaotic, though that is because the divergence here is not a colour gradient.

Local Outlier Factor gives a completely different result, although is not necessarily better or worse. As described in the theory section, it detects outliers on a local scale, and therefore focuses outliers near clusters of points. As seen in figure 3.7 this approach looks more chaotic, though it catches more singular separated points, likely at the expense of many false positives. It is also less probable for whole clusters to be considered anomalous.

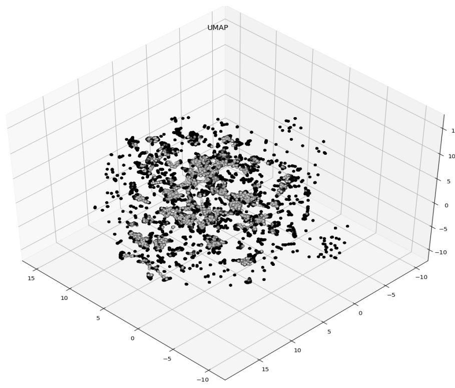


Figure 3.6: *One-Class SVM, all the black points are considered outliers.*

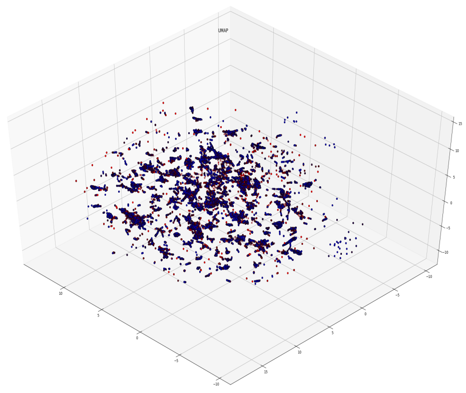


Figure 3.7: *Local Outlier Factor*

3.8.4 Difficulties

Our initial expectation was that we would gain access to data with some kind of “truth table” as to whether a transaction is fraudulent or not. This is a prerequisite for any kind of supervised learning, and is necessary to verify the accuracy of any findings. Later it turned out this data would be unrealistic to acquire, and therefore we had to make do with the data available from within Signicat. This meant that our only options were various kinds of unsupervised learning, where we attempted to detect divergence from what is “normal”. Therefore we had no way of verifying that the divergent transactions are in any way fraudulent. Considering the rarity of fraudulent transactions in the real world, the best outcome here was a score of how divergent a point is. The most likely cause for the initial lack of needed data, was poor preparation before we begun. We might also have saved some time had we not been waiting for fraud risk indicator data, and likely concluded this part of the thesis sooner.

Chapter 4

Analysis Part 1

One of the first issues that come as a result of using outlier detection on this dataset, is that the majority of the transactions are originating from within Norway. This leads data points originating from other countries, especially countries far away from Norway, to quickly become labeled as outliers, see figure 4.1 for an example.

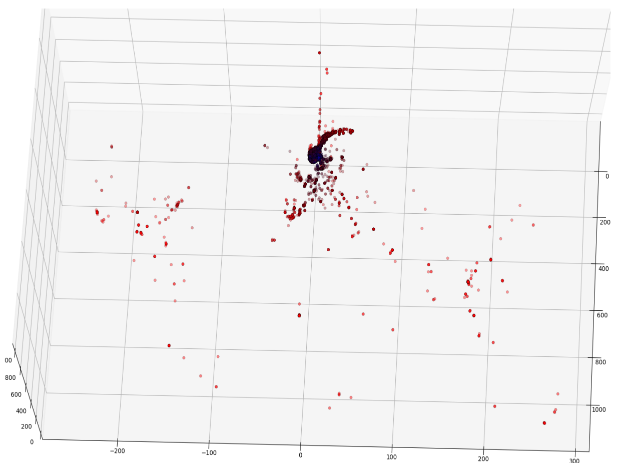


Figure 4.1: *3D visualization of the dataset after dimension reduction with PCA. There is a distinct pattern here that implies geographical location plays an important role in deciding how divergent a point is. Keep in mind that most of the transactions originate from within Norway.*

While this is not as apparent with the other dimension reduction methods, it is still raises a concern whether the geographical location should be included. Is the geographical distance from Oslo, Norway a good measure for divergence? In addition, some customers have different configurations and use cases for the services provided by Signicat. This results in transactions from customers who use the service with an old configuration to be classified as outliers. In a more extreme case, one single customer accounted for 93% of the 5% most divergent points, although this was not prevalent through all configurations of dimension reduction and scaling. Worth noting here is that the customer accounted for approximately $\frac{1}{6}$ of the total data points in this set.

Since we initially attempted clustering the data, we will demonstrate one result of the clustering here.

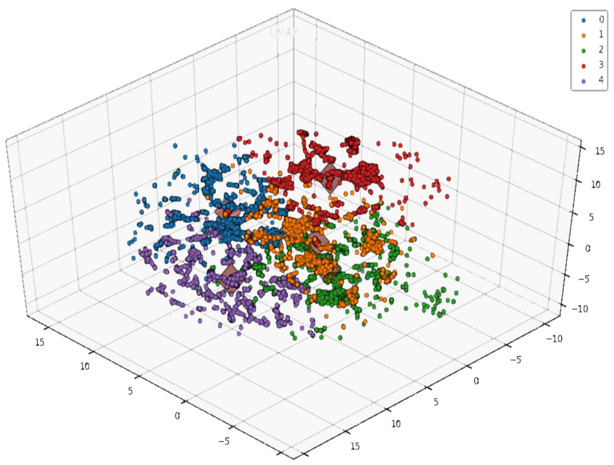
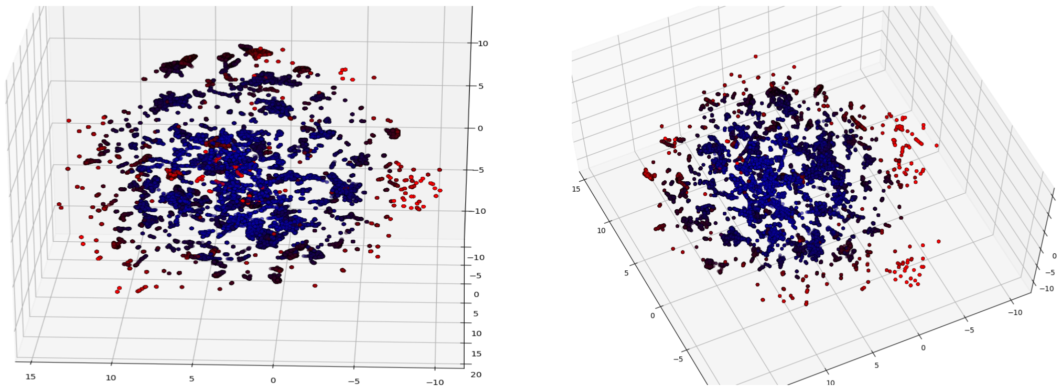


Figure 4.2: *3D visualization of the dataset after dimension reduction with PCA. There is a distinct pattern here that implies geographical location plays an important role in deciding how divergent a point is. Keep in mind that most of the transactions originate from within Norway.*

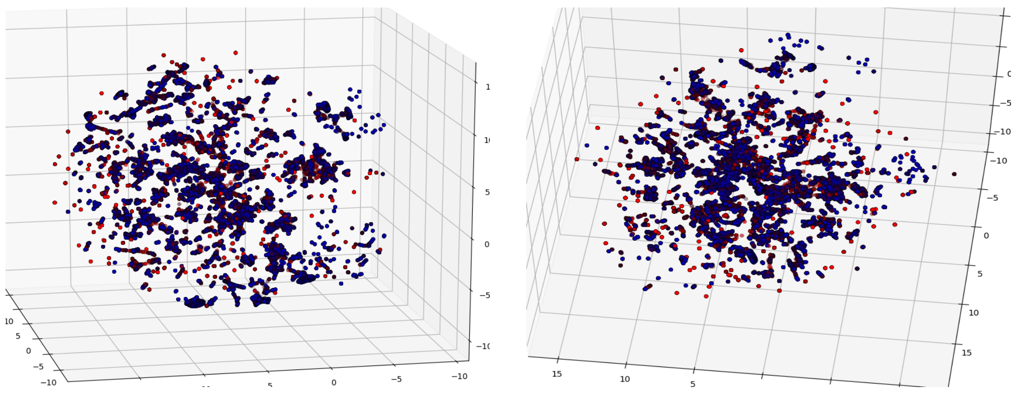
In figure 4.3 we can observe how the dataset looks like after accounting for how likely the data points are to be outliers. Here we can observe that the majority of points are grouped together in clusters near the center, surrounded by smaller clusters nearby. This might be a symptom of UMAP preserving some of the global structure. Further out from the center there are a few clusters, which have a quite dark colour, because they are far out from the norm, but are still in close proximity to other points. Those of the points with the strongest red gradient are visibly the furthest away from the dense areas, which is what we expected. Some of the isolated points seem to be blue or dark, while they could be good candidates for outliers.

The result in figure 4.4 is consistent with the behaviour described in the examples in the method section. Here we can see that more of the singular points between the clusters get a stronger red colour, indicating they have a high divergence. On the other hand, the groups of point near the outer edges of the point cloud get lesser values of divergence, despite being far away from the big clusters.



(a) View 1 (b) View 2

Figure 4.3: *The result is achieved using UMAP, Robust Scaler and Isolation Forest. There are two different views, because the three-dimensional structure is hard to discern when viewing it from only one perspective.*



(a) View 1 (b) View 2

Figure 4.4: *Same as figure 4.3, though it uses Local Outlier Factor instead of Isolation Forest. The variance in the point distribution can be attributed to UMAP giving different results after each run.*

By utilizing the results from both clustering and outlier detection we observed that outliers are distributed across multiple clusters as shown in table 4.1. The chance of finding a "pure fraud" cluster is minimal.

Outlier score	Outlier	Cluster
0.02	Outlier	4
0.01	Outlier	3
0.02	Outlier	1
0.05	Outlier	4
0.05	Outlier	2
0.02	Outlier	4
0.05	Outlier	4
0.05	Outlier	4
0.03	Outlier	2
0.02	Outlier	1
0.03	Outlier	0
0.05	Outlier	0
0.01	Outlier	1
0.02	Outlier	4
0.01	Outlier	0
0.02	Outlier	4
0.02	Outlier	4
0.04	Outlier	3
0.03	Outlier	2

Table 4.1: *Outliers detected by Isolation Forest and which clusters the outliers belonged to, using KMeans with 5 clusters.*

Chapter 5

Discussion Part 1

5.1 Reliability

The lack of any way to determine which points are fraudulent makes the task of predicting whether a new transaction is fraudulent impossible, though by using outlier detection, we can at least determine that something is out of the ordinary. The definition of "ordinary" here is of course left for the machine learning algorithms to define, and with this dataset it often ends up being based on the geographical location or user agent. We can approximate a fraud risk score by using the divergence score mentioned before, and therefore complete the task as it was specified. The reliability of the score can not be verified without testing it on transactions that are certainly fraudulent.

The use of geographical location in the anomaly detection warrants consideration. In theory, if most of the transactions originating from within Norway, any transaction outside Norway will be much more divergent, as the algorithm will consider being in Norway the norm. A supervised learning method should not relay so heavily on such factors, as the algorithm could learn that geographical location is not the cause of divergence from the labels. Since we are forced to rely on unsupervised methods, filtering it out manually might increase the accuracy.

5.2 Result

As for the result displayed in the Analysis chapter, if the dimension reduction is to be trusted, the Isolation Forest does well when it comes to the clusters close to the center. Applying Local Outlier Factor instead can catch more of the points between the clusters of points, while seemingly also being stricter as to what counts as outliers. Which of these is more desirable depends on the use case, though for this report we can not conclude that one of them gives better accuracy, because as mentioned before, we have no metric to use for this purpose.

In addition, clusters and outliers were mostly determined by the user agent and GeoIP-location. This made decision trees and association rules not as useful as we hoped for. Decision trees could find the relevant features which outliers and clusters were based on, however the trees ceased to provide a good visual overview due to the high variance in the user agent features. This also made it difficult to find interesting association rules, as the variety in user agents led to low support values. It is difficult to provide association rules covering geographical location since some of the data is continuous, and association rules algorithms are not suited for those. If we had to work further on this, we could perhaps prevent the high variance by grouping similar user agents and using discrete intervals for the continuous values.

Another issue when using outlier detection, is that it is difficult to measure the accuracy of the model, and to conclude whether it does well or poorly. We have no concrete examples of what should or should not be considered divergent, making it impossible to verify whether the model is reliable.

In addition, the results of doing outlier detection on the entire dataset at once might not necessarily be useful, although it is a first step. Since the use cases vary between customers, creating one common model will not highlight the normal state for each customer. Some customers might become outliers when nothing out of the ordinary is happening. Therefore, creating separate models for each customer should give more accurate representations.

Clustering also can potentially have some use cases, as transactions can have other properties that could be interesting to group by. This could then be used further to identify in which group new transactions belong to quickly. Still, this was not the task we were supposed to solve, so we did not pursue this further.

Chapter 6

Conclusion Part 1

We have examined ways in which machine learning, specifically unsupervised learning, can be applied for detecting authentication fraud. Unfortunately due to lack of appropriate data, no conclusions can be drawn as to the existence or non-existence of fraud in the dataset. The only information our models can provide is a level of divergence any one point has from the rest, creating a value that fits the task description to some degree. A sudden divergence in the activity of a single user or company could imply something is out of the ordinary. This could supplement existing methods of detecting suspicious activity, though it can not be used as a basis to determine whether any one transaction is fraudulent. Should the appropriate data surface in the future, a more reliable model would be possible.

If this data would become available, one could disprove the validity of our approach should most of the fraudulent points end up with within the dense clusters, or support it should they most of them be far from the center. One could also verify which of the methods is the most accurate, or whether the combination we suggested in the discussion chapter is a viable option. In the end, our approach can at the very least provide a starting point for further work on the subject.

6.1 Further Work

This can be further developed to follow a single individual or company over time, which could potentially show sudden divergence from the normal behaviour of that person or company. Such a alteration would potentially allow for much more accurate risk scoring, as the variance for a single individual or company should on average be much lower. This would however require access to datasets which contain personal information, and privacy concerns would have to be addressed beforehand.

The most important improvement would be to acquire more relevant data, even from customers if possible. The datasets we used have quite limited information, and are insufficient to give the desired results. A possible improvement to our results would be to combine both Isolation Forest and Local Outlier Factor, in order to catch both the isolated points closer to the center, and the ones in smaller clusters further out. This could be done by taking the average of the two, or to weight them according to which points are desirable to catch.

There are also many more options to analyze the data which does not involve machine learning that could be attempted. Those would make it possible to draw conclusions about whether or not machine learning is necessary for this problem in the first place.

Chapter 7

Introduction Part 2: Time series

Whenever an issue occurs, it is already too late to prevent it from happening. At this point, the only thing one can do is try to minimize the damage and attempt to find the cause. The occurrence of errors, or operation anomalies, can in some cases cause severe disruptions to the delivery of service for companies working with information technologies. If such errors could in any degree be predicted and mitigated early, the problems caused could be avoided, saving costs and working hours. Can this be achieved using modern machine learning techniques? With the increasing popularity of machine learning, new applications of these algorithms are discovered continuously.

The objective of this project is to examine the potential of predicting Signicat's server performance metrics. Can we accurately predict the resource utilization or network traffic in the future? In order to limit the scope of the project, we will focus on predicting one single variable in the system. We selected CPU activity because it should reflect the activity in Signicat's systems well. This CPU activity variable represents the average percentage of free CPU time for the system in the last minute.

From this point, we will present some of the most relevant theory necessary for this project, and show how we applied it to predict a variable from Signicat's systems. Furthermore, we will discuss the reliability of the models, and consider which factors might affect the result. Finally, we conclude the report with suggestions for future improvements.

7.1 Thesis Problem

Can a form of Recurrent Neural Networks provide accurate predictions for Signicat's service performance metrics, and do the predictions provide useful information?

Chapter 8

Theory Part 2

8.1 Time Series

Time series are data gathered from a period of time. This data can take various forms, for example weather, sensor measurements or stock prices. It is common to separate the time series in two categories, univariate and multivariate. Multivariate time series are as the name implies, time series which contain multiple variables for the same points in time. Univariate time series on the other hand, only contains one variable per point in time. [59]

8.1.1 Forecasting

There are different approaches one can take to predict groups of future values for a time series. For this report, we will focus on the two methods described in the following sections.

Recursive Strategies

The recursive strategy attempts to predict one point ahead in time from the known values, by using the predictions produced as a basis to predict further points. This is visualized in figure 8.1. The disadvantages of this strategy is that the error from a previous step affects every step afterwards. Since no prediction is perfect and has a certain loss, basing the next prediction on the previous one will in essence mean that the new prediction is based on partially inaccurate information. This again leads to the loss of that next prediction being greater, as it is composed of both the loss of its predecessors and its own prediction error. This means the errors will rapidly accumulate over time. [9]

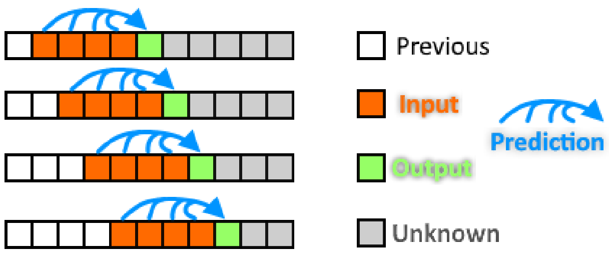


Figure 8.1: *A recursive prediction strategy.*

Multi-Input Multi-Output (MIMO) strategy

The MIMO strategy is based on the idea of stochastic dependency. Unlike Recursive strategies, a MIMO strategy returns all future values in one single step. This means there will be no accumulation of errors, though because the model will be used to predict all future values at once, it could end up being less flexible and result in worse predictions overall. [9]

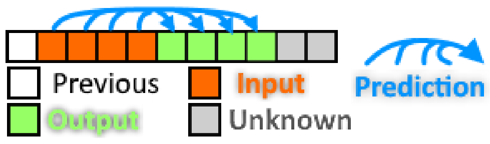


Figure 8.2: *A MIMO based prediction strategy.*

8.2 Neural Networks

Neural Networks can be described as structures comprised of many neurons connected together. A neuron is a simple processing element that takes one or more inputs and gives an output. The network starts with an input layer and ends with an output layer, between these there are one or more hidden layers connecting with connections between them. For a visual representation of a NN see figure 8.3. [8]

There are several types of NNs, each made to tackle different tasks. We will focus on the type of NNs often used for time series prediction.

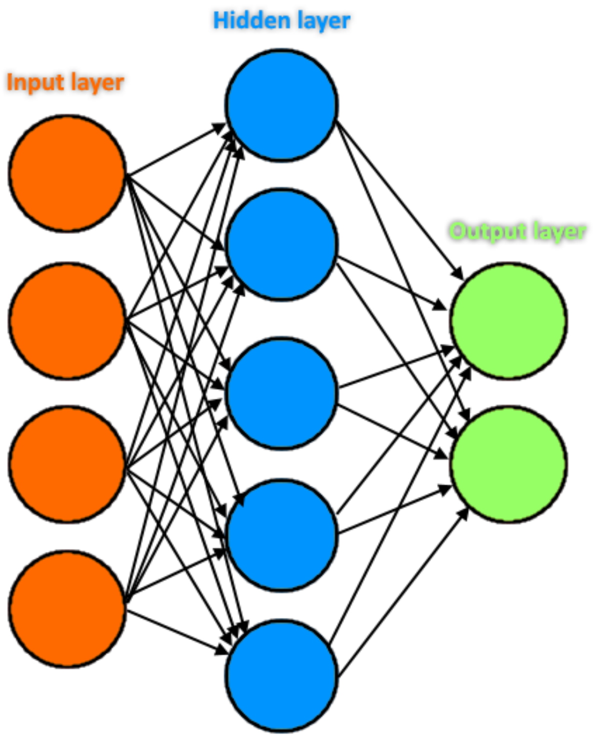


Figure 8.3: *A neural network with one hidden layer.*

8.2.1 RNN

Some of the most relevant NN to time series are Recurrent Neural Networks (RNN). These networks are distinguished by looping the previous output from the unit and adding it to the new input. This leads recurrent networks to have a form of short term memory, where previous decisions impact future ones. This memory fades away quickly as new predictions are made. The use of both past and present information makes the order of the samples relevant. Its architecture is quite simple compared the other architectures described below. [16, 23, 50]

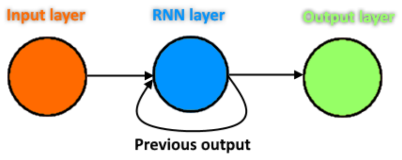


Figure 8.4: *A RNN network.*

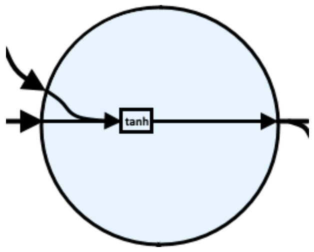


Figure 8.5: *A RNN cell.*

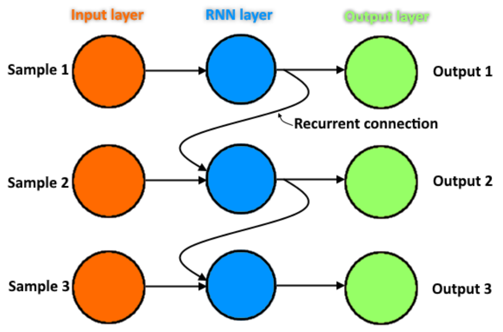


Figure 8.6: *An unfolded RNN network predicting on three samples.*

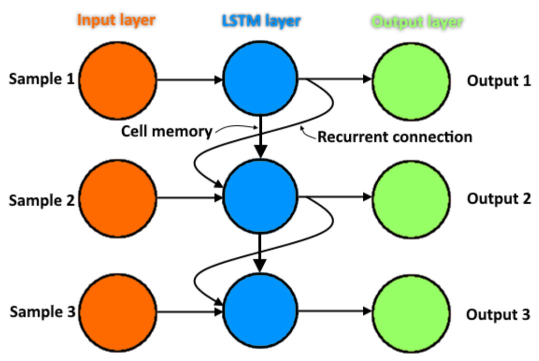
For more detailed theory on RNN types not described here, and RNNs in general, see [50].

BRNN

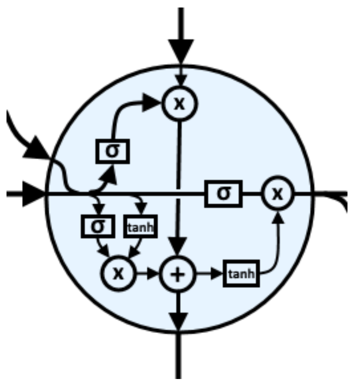
While the standard RNN only forms recurrent connections in one direction, it can also be modified to form connections both ways. Using a Bidirectional RNN(BRNN), the neural network will learn connections not only in the positive time direction, but also the negative. Other RNN types can also be made bidirectional. [52]

8.2.2 LSTM

LSTM stands for “Long Short Term Memory”, and is designed to let RNN retain their memory for longer and make more complex predictions based on both short term and long term trends. For usual RNNs, this either takes a extremely long time or does not work at all. LSTM aims to remedy the issues of exploding and vanishing gradients that are common in RNNs, by keeping the back-propagating error more constant. To achieve this, it uses gates and memory cells. The memory changes infrequently, allowing the learning to remain more stable than simple RNNs. [23]



(a) An unfolded LSTM network predicting on three samples.



(b) LSTM cell

Figure 8.7: *LSTM*

8.2.3 GRU

The Gated Recurrent Unit has some similarities with LSTM, though it uses an update and a delete gate to adjust the learning inside the unit, rather than keeping additional memory. The update gates control the rate of updates to the functions, while the delete gate resets the unit, allowing it to restart, and continue without considering all that it has seen earlier. Using these update and delete gates, the GRU can store information for longer periods of time, and counter the vanishing gradient problem. [50, 28]

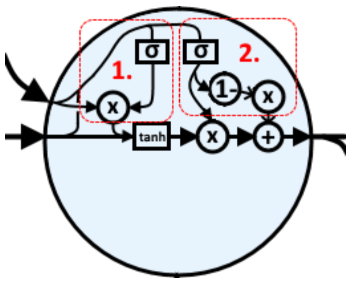


Figure 8.8: *A GRU cell.* ***1.*** *is the reset gate.* ***2.*** *is the update gate. The structure around the cell is the same as in figure 8.6.*

In some cases GRU performs better than LSTM and learns faster in terms of CPU time [28, 14]. Though it is not always the case as mentioned by [14, p. 7]:

> ”These results clearly indicate the advantages of the gating units over the more traditional recurrent units. Convergence is often faster, and the final solutions tend to be better. However, our results are not conclusive in comparing the LSTM and the GRU, which suggests that the choice of the type of gated recurrent unit may depend heavily on the dataset and corresponding task.”

8.2.4 Optimizers

The algorithms used to adjust the weights/parameters of NNs during training are called optimizers. An optimizers goal is to reduce the NN model’s loss and doing it the most optimal way. Different optimizers are designed to tackle different challenges. The choice of optimizer can be what makes a model succeed or fail. [49]

A very common algorithm for adjusting the weights toward their optimal values is known as Gradient Descent. It uses mathematical gradients in order to navigate the weights towards the most optimal value. The length of the steps taken using the gradient is known as the "learning rate". Other optimizers are often based on gradient descent. [49]

All optimizer algorithms used in this project are based on gradient descent. These are the ones we use:

- Adam
- Nadam
- RMSprop

Loss Functions

A loss function calculates how far the prediction deviates from the true value. Different loss functions have different properties and will affect how the model learns. Mean Square Error (MSE) is the most popular loss function. Small error values have little effect on the loss, and larger error values cause the loss to increase more drastically. Mean Absolute Error (MAE) is just the absolute value of the error value. Root Mean Square Error (RMSE) is the square root of MSE, which causes the loss to increase more gradually. All loss functions described above always result in a positive value regardless of the input.

8.3 Regression

Regression predicts continuous numeric values by expressing a relationship between the dependent feature and the predictors, and there are many different models for expressing this relationship.

8.3.1 Linear Regression

Linear regression finds the relationship between two continuous variables and expresses it as an equation with in the form $y = a + bx$, where x is a explanatory feature used to predict the dependent y. The equation is the best line to fit x and y observations from the data points. One explanatory feature is rarely enough to make a precise prediction, so we can take advantage of more features using multiple linear regression. A model with multiple predictors has the form $y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ...\beta_p x_{ip} + \varepsilon_i$ for $i = 1, 2, ...n$. A common method for estimating the unknown values is to use least square error to minimize the vertical deviations between the best fitting line and the data points. For details see [33, 32].

To avoid overfitting in our linear regression model, one can add a regularization term to the least square approach using Lasso, Ridge or Elastic net. These regularization terms has a lambda parameter. If λ is too low we will get a model very similar to linear regression, but if the parameter is to high however, the chance of underfitting will increase. Choosing a optimal lambda can prevent both underfitting and overfitting. [58]

Lasso

Lasso or L1-regularization adds the absolute value of β coefficients to the least square calculation. [58]
$Min \sum_{i=1}^{n}(y_i - \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ...\beta_p x_{ip})^2 + \lambda \sum_{j=1}^{p} |\beta_j|$

Ridge

Ridge or L2-regularization uses squares of β coefficients instead. [58]
$Min \sum_{i=1}^{n}(y_i - \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ...\beta_p x_{ip})^2 + \lambda \sum_{j=1}^{p} \beta_j^2$

Elastic Net

Elastic net uses both L1 regularization and L2 regularization. [58]

Least Angle Regression

Least Angle Regression or LARS is a fast step-wise algorithm used to fit linear regression models. To include the same solutions as linear regression, the algorithm only require steps equivalent to the p predictors. All β coefficients are initially set to zero, and it then finds the most correlated explanatory value by finding the least angle between the predictors and dependent. It translates in the direction of the most correlated explanatory value until a new one has the same correlation and takes over. The process continues until all the correlated features have been incorporated in the model. If the number of the correlated features equals p, we will get same model as with linear regression. [26]

8.4 Decision Tree

Decision trees are a supervised method used to both classification and regression problems. The method is non-parametric, which means that it is not dependent on numerical data. The trees are structured by approximating sine curves with if-then-else decisions, and the complexity of these rules is equivalent to the depth of the tree. The complexity is correlated to how fit the model is, and to prevent overfitting or underfitting, we can adjust the minimum samples required for each sample and the max depth of the tree. Trees are useful to create an easily interpretable visualization of a problem, which does not require any data processing such as normalization, dummy variables and blank data removal beforehand. It is also a suitable method for finding which features are relevant for predicting the dependent feature. [44]

8.4.1 Random Forest

Random forest is as the name indicates, a forest containing decision trees. Each tree in the ensemble is created by generated random vectors. One way to generate the random vectors is to use random selection with replacement on the training set. The vectors are independent from each other and have the same length as the training set. Both the training data x and the random vector Θ_k is used to assembly the k-th tree. When the number of k trees are created, the most popular model becomes the chosen one. Random forest is another great method for preventing overfitting problems in addition to depth and leaf node adjustments. [11]

8.5 SVM

Support vector machines can be used as a classifier and a regression method. SVMs are inherently binary classifiers. It finds a function for predicting a target feature, which focuses on minimizing the error between the output from the input points and the target. For linear SVM we will get a function with the structure $f(x) = wx + b$, where w is a weight vector, x is a vector containing predictors and b a constant. Figure 8.9 is a simple linear model for visualizing how SVM works. The blue and the green patterns are separated from each other by a hyperplane, which is constructed by the marginal line $wx - b = 0$ and the two other lines on each side. The three points with black border lying on the edge of the hyperplane are support vectors. The distance between the marginal line and the hyperplane edges is the margin $\frac{1}{||w||}$, and the total margin is $\frac{2}{||w||}$. For a optimized marginal line, the algorithm seeks to maximize the margin, and to do that it can minimize $||w||$. Instead of finding the minimum of the term $||w||$, it finds the minimum of $\frac{1}{2}||w||^2$, because this term is more convenient for more complex mathematics later on. For more details about the mathematics, and nonlinear kernel functions such as poly, sigmoid and RBF(Radial basis function) see [65, 46].

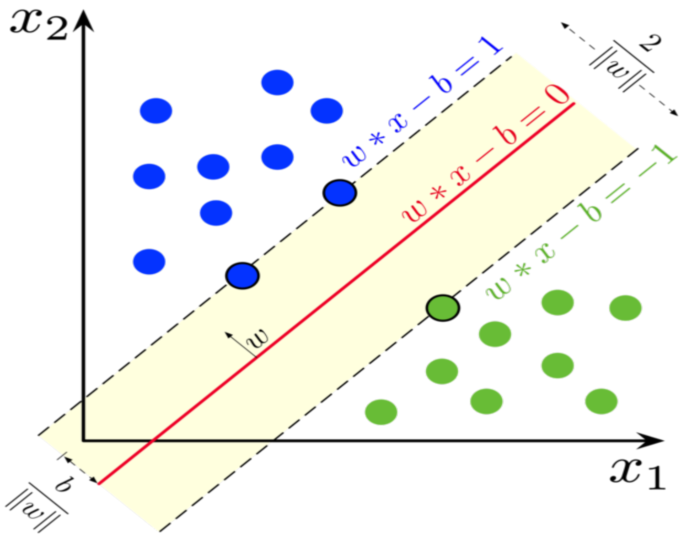


Figure 8.9: *Hyperplane and margin for a SVM-model trained on two predictors [62]*

8.6 Common Challenges in Machine Learning

8.6.1 Datasets

Often the hardest challenge in machine learning is acquiring a relevant and adequate dataset.

8.6.2 Run Time

Training some of the machine learning algorithms can require tremendous amounts of time and computational power. The complexity of the algorithm or model can become so great, that training will days or weeks to complete. The required time can be reduced using techniques such as feature reduction or early stopping [47].

8.6.3 Overfitting and Underfitting

When fitting a model to a dataset, it is possible that the model becomes very good at predicting the values that are within the training set, but poor at predicting new values such as those found in validation set or test sets. This phenomenon is usually called *overfitting* [47, 56]. One method of countering overfitting is mentioned in 9.5.2 on page 43. *Underfitting* on the other hand, is the phenomenon that occurs when the model is too simple, causing it to be unable to learn the patterns in the dataset. [56]

8.6.4 Exploding/Vanishing Gradient

When error gradients grow during training, they can become too large, leading in many multiplications with values greater than 1. This results in the gradient growing too quickly, making the learning less stable. This is commonly referred to as *exploding gradients*. The opposite case, *vanishing gradients*, is when the multiplication happens with a very small value close to 0. Repeated multiplication will cause the gradient to quickly shrink, eventually becoming so close to the lowest possible represented value that is practically nonexistent. Both of these are issues for RNNs, because the output is reused as input, retaining the values like described in 8.2.1 on page 31. [22]

8.7 Training, Validation and Test Sets

Training data is used to train the model, by optimizing the weights and biases. Test data has the purpose of allowing to test the accuracy of the model, and it must be separate from the training data in order to be a reliable test. Should the model train on the test data, the results could be biased due to overfitting.

8.7.1 Train-Test Split

This is a simple and intuitive split method. One selects a percentage of the data to be used for testing, and another for the validation data, then take the corresponding percentage from the end of the dataset. This is illustrated on figure 8.10. It is usable for time series because the order of points is retained, which is necessary as the temporal displacement of points in a time series can affect the result. The advantage is the easy implementation, and only one model needs to be created and trained. One of the drawback of only using points from the very end of the dataset for testing, is that the tests might not be representative of the entire dataset. An example of this is a strong trend in the train set, but the opposite trend is in the test set. [29, p. 223]

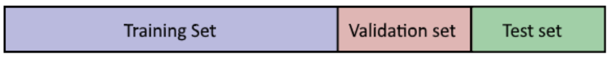


Figure 8.10: *Splitting of a dataset into training, validation and testing sets.*

Random Split

Splitting randomly involves taking a certain percentage of the data is taken from random indexes in the dataset for use in the testing set. This is illustrated on figure 8.11. Random split has the advantages of having varied training, validation and test sets [29]. Splitting this way is not well suited for use with time series because data leakage can occur if precautions are not taken. In addition, the temporal continuity is broken, which can be an issue for RNNs, see 8.2.1 on page 31.

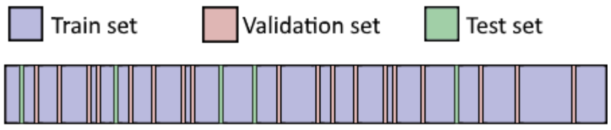


Figure 8.11: *Random split of a dataset into a train, validation and test set.*

Deterministic Split

Deterministic splitting involves taking a certain percentage of the data from indexes at specific intervals in the dataset for use in the testing set. This is illustrated in figure 8.12. [29, p. 223] explains why deterministic split should not be used: "A deterministic method, such as selecting every nth observation as a testing fact, is also not recommended since it can result in cycles in the sampled data due solely to the sampling technique employed."

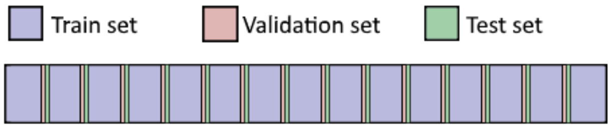


Figure 8.12: *Deterministic split of a dataset into a train, validation and test set.*

Walk-Forward Validation

In walk-forward validation the time series is divided in several parts. The parts are then used in the training set, the test set or not at all. When moving to the next step a new model is trained, causing this method to be more computationally expensive. The advantage is that larger numbers of tests can be performed, providing more accurate results on the performance of the model. We will take a look closer at the two variants of walk validation. [29]

The first variant is Expanding Window. The dataset is divided in segments. Initially only the first segment is used for training, and the second for testing. In the further steps the test segment becomes a part of the training segment, while the one after it becomes the new test segment. This proceeds until the last segment becomes the test segment. The advantage here is that both test and training sets are well represented, and little potential training data is lost to testing.

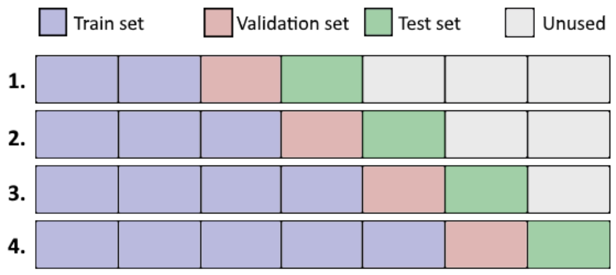


Figure 8.13: *Expanding Window increases the amount of data used for training used each step. A square represents a "part". A segment is all the squares of the same colour in the figure.*

The other variant is Sliding Window, also known as Moving Window, where the difference is that the size of the training set is kept constant throughout the steps. This allows the older parts of the data to be "forgotten", which is useful if only the newest points are of greater relevance. [29]

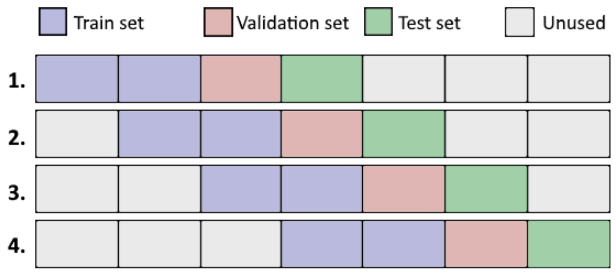


Figure 8.14: *Sliding Window keeps the amount of training data constant while moving the training window. A square represents a "part". A segment is the squares of the same colour in the figure.*

Chapter 9

Method Part 2

Since we had no prior experience on the subject of time series prediction, the first step was to identify which approaches could potentially achieve this task. One of the first methods that appeared common for this kind of problem was using recurrent neural networks[29]. LSTM seemed especially interesting, because we suspected some of the correlations between columns might be stretched out over longer periods of time.

We began this part of our project the moment we gained access to performance metrics from Signicat's Prometheus service. With help from one of the employees here we quickly set up a shell script to download the various performance metrics. We worked further upon this script to allow us to download larger numbers of data points than what Prometheus allows per request, because of this we could use data from a much larger time span.

While we initially decided to use LSTM, we also set up some simpler regression models to check whether they do better in this case. If the neural network model can not defeat these models, then the simpler approach might as well be used. For all our models we used MSE as the loss metric, for more about loss see 8.2.4 on page 33.

Towards the end of the project we discovered that one can use hyperparameter optimization libraries to better test different hyperparameters for the model, so we applied one to find the optimal configuration. We chose to use Talos, which is further described at 9.5.1 on page 42. This also gave us the opportunity to easily compare different model configurations, such as using different layers or other number of layers, to make sure that our choice of using LSTM was justified. It was at this point discovered that GRU does better on this dataset.

Due to our lack of initial knowledge on the subject, we missed a better approach to the problem in the beginning. Considering that in a practical application one would always expect to see new points in the correct temporal sequence, it would be plausible to train in a similar way, better utilizing the memory properties of RNNs. By the end of the project we attempted to correct the data separation by applying walk-forward validation. Doing so allowed us to train the model better, and utilize the data more efficiently. This led us to run Talos again with a few alterations, to make sure we got a good model.

9.1 Datasets

We had access to the last few months of performance metrics, which are acquired through Signicat's Prometheus service. Older values are not accessible through Prometheus, so we could not use a dataset better distributed throughout the year. The metrics were discovered through Grafana and the alerts used in Prometheus, with the assumption that these alerts and graphs show and use information that is of interest or importance to the system. It was difficult to gather much knowledge about the variables and various aliases used in these systems, as there was no direct documentation, and few people know all the details of the setup. An alias is just an abbreviated name for a host. The time distance between points can be specified, resulting in time series data with a certain resolution.

Processing it to an useful format only requires it to be unpacked from the JSON object, resulting in a array of values. Unfortunately, Prometheus enforces a limit of sending out maximum 11 000 data points per request. To circumvent this limit, we used a Shell script which divides the time period into small enough intervals, and then sends multiple requests. The script concatenates the JSON objects and saves them to a file. This happens for every variable. The dataset is then loaded in Python and assembled as a Pandas data frame, ready to be used for machine learning. In total this data frame contains about 50 columns with different metrics. If a metric column is missing one or more data points in the interval, the column is dropped. All Prometheus metrics are composed of continuous numerical values.

Our scripts permit to select the variables with emphasis on one specific alias or as a regular expression for them. We attempted to filter out some of the aliases such as proxy-services, after being informed that these should not be relevant. In addition, when using regular expressions to filter aliases the variables become averages for those aliases. This greatly simplified the acquisition of data points, and gives a more global scale graph, at the possible expense of accuracy. Not all aliases are represented in all of the variables, so selecting one specific alias also reduces the amount of viable columns. Some of the variables, such as swap space, are represented as the rate of change they have rather than the total value. The idea is to make the variance more visible to the neural networks. In such cases, a change from 1000.0 to 1000.1 would otherwise appear as a very small change even after the data is scaled, and considering most of the changes could manifest themselves as such, the variable would be almost constant. Those of the variables that were already represented as percentages, for instance the amount of free CPU, were kept as such.

Our target variable for the purposes of this report, the percentage of free CPU time, is a general metric for the computing load on the system. The exact value we use is a sum of the relevant aliases divided by the amount of aliases used, and therefore an average per alias. In order to compare the various methods in a time efficient manner, we used a small dataset with the time interval beginning at April 15th at midnight and ending at 1st May at midnight. When we select our final model, we intend to use more data.

Some of the variables we chose practically did not influence the end results, as they almost always were missing data, and our policy was to disregard the columns where that was the case. Especially columns regarding amounts of HTTP statuses, such as 404s or 500s, were always lacking data, and therefore not used.

9.1.1 Pre-processing and Scaling

There are many options for scaling the data. First of all, the result of applying default scikit-learn normalization was that all the points changed their value to 0, so we did not try it further. The cause might have been incorrect settings. Other alternatives are the Robust Scaler, Standard Scaler or Min-Max Scaler. The Standard Scaler can not guarantee balanced feature scales in the presence of outliers [1]. We used the Min-Max Scaler to scale the variables to the range $[0, 1]$.

9.2 Naive Approach

In order to get an idea of how well the models perform, we compare them to a simple naive prediction. There are a few naive approaches we could use, for example selecting random values from the input or taking a weighted random number from the input space's distribution. Our naive method of choice here is simply taking an average of the entire input and use that as the future predictions. The output is in other words a straight line parallel to the x-axis.

As an example, when given the input array $[[2, 0, 1, 9]]$ and the output size is set to 3 values, the output would be $[[3.0, 3.0, 3.0]]$.

While this method is an extremely simple approach, it actually yields a fairly good accuracy, due to the data often swinging back and forth in a sine-like pattern. A more complex naive approach we tried, where the line is no longer parallel to the x-axis, used the derivative of the line drawn between the average values of the first and second half of the input points. This was in an attempt to make the "prediction" more in line with where the data is headed, and this version did not do much differently than the simple approach. Either way, a good model should be able to defeat any such methods on the testing set.

An advantage of using this method as a comparison over using something that generates random values, is that it outputs the same value after every run. Since several of the methods were applied separately, often in parallel, it also helps us ensure that the data was divided equally in all the applications. Since this is such an important part of comparing the methods, having this value be constant ensures we filter out any mistakes.

Calculating the loss from this naive method and comparing it to the current model see whether it is better. Furthermore, using the difference $loss_{naive} - loss_{model}$ we can compare different methods by how well they do versus the naive method, for instance by sorting them by this difference. The greater the difference value, the better the model is. If the difference is 0 or less, we know the model is bad as it does equally well as or worse than the naive method.

9.3 Feature Selection

Our datasets contains about 50 features, which can lead to a long and memory intensive training process. Especially if we use many points backwards to predict ahead. If we use 10 points, every x sample will contain 500 values. To prevent this we can use feature selection to find the most correlated predictors for the target. We have used ExtraTreesRegressor from scikit-learn to reduce the number of features, as they gives us information about relevant features for the target feature. Some target features only require 4-7 features, which means shorter run times, potentially better learning, and the option to use more data points without depleting the memory. As shown in table 9.1 the loss difference before and after feature selection is quite low, as not all features are correlated to the target in the first place. The variance in the selection of features is caused by the implementation randomly initializing the decision trees each time.

Method	With feature selection	Without feature selection
Random forest	0.0393903121708898	0.03891878954942943
Lasso	0.0664701189850755	0.6647011462052438
Elastic net	0.10126800923282353	0.10126800747470582
Lars	0.07512170871355144	0.07512170871354992
Ridge	0.06221560252076057	0.0592391682566869

Table 9.1: *This table shows the MSE for CPU prediction after running the scikit-learn methods with and without feature selection. 30 points are used to predict 10 points, and it uses Min-Max Scaler from -3 to 3.*

9.4 Scikit-learn Regression Methods

The methods we imported from scikit-learn were linear regression, Lasso, Ridge, Elastic net, Lars and Random forest regression. We tested the scikit-learn methods against each other using MSE and plotting them together. The scikit-learn models were also tested on how well they adapted to new data in form of using expanding windows. Recursive prediction strategy was also tested, however the output from the models quickly converged to a straight line, so we decided to focus on using m points from the past to predict n points in the future.

We did not use support vector machines in our comparison, because scikit-learn's implementation of SVM did not handle the same y input as the other methods we used. For predicting multiple points ahead we used 2D arrays of y values to train our models, and the implementation of SVM requires 1D arrays.

9.5 Model Optimization

To get the optimal model, much experimentation and testing is required. There is no universal best model, it will vary based on the dataset and what you are attempting to accomplish. We used the two techniques below to improve the model.

9.5.1 Bayesian Optimization

We used the Bayesian Optimization library Talos to find the best hyperparameters for the NN model. Before the optimization process begun, we needed to provide options for the hyperparameters, which include number of layers, units per layer, activation functions and optimizers. This is useful for testing many alternative configurations of the NN efficiently. The downside of testing all the possible combinations is that it takes a very long time. [54]

In order to run it within reasonable time, we reduced the dataset when running Talos as explained in 9.1 on page 40.

9.5.2 Early Stopping

We used early stopping to reduce overfitting of the model during the training. Early stopping monitors validation loss and stops the training when it stops decreasing. Early stopping can also reduce the time required to train a NN, because the training can end sooner [47]. We also restore the weights of the model from epoch with the lowest validation loss, in order to get the best performing model.

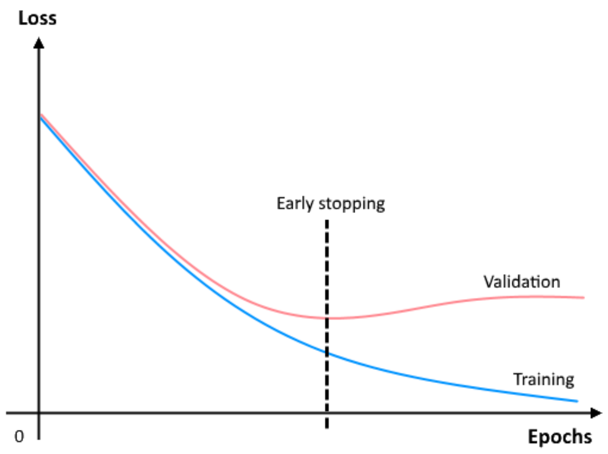


Figure 9.1: *An example of early stopping, the vertical dotted line indicates where the training is to be stopped.*

9.6 Comparability of Results

In order for our results to be comparable across different methods, we ensured that we used the same data, data processing and that it is scaled equally for the testing, validation and training sets. This is important for gathering reliable data, so that we can draw conclusions as to what works and what does not. One difference between the methods is that the scikit-learn methods does not utilize validation sets.

Chapter 10

Analysis Part 2

All the results here use CPU as the target feature, and utilize feature selection. This leads to only the columns "Porter", "Node Load 1", "JVM Memory", "Audit BaseEvent", "JMS Consumer Messages" and previous CPU values being used. Note that this selection varies for each execution, though the selections often have similarities. The number of columns used for this variable varies between 3 and 7, see 9.3 at page 41 for additional information. The full list of columns in use before feature selection is available in the program code, which might have restricted access.

10.1 Talos

See appendix A for the results of the Bayesian optimization using Talos, before we altered our data separation. The top result is a network with one bidirectional GRU layer with the ReLU activation, 60 units and the Nadam optimizer.

This, along with time constraints due to the long run time of this optimization, led us to exclude the normal RNN from the next run. RNN was only represented once in the top 20 results in appendix A, and has also often been found inferior in other similar projects [30]. Furthermore, the tanh and sigmoid activation functions were excluded, both because of poor performance and time constraints. From this point, we compared only LSTM and GRU, which yielded the results in appendix B. Note that this version had early stopping implemented in order to reduce training time and overfitting, as well as an optional additional dense layer at the end.

In appendix B we appended the evaluation/test loss to the table. The first table, sorted on validation loss, shows that a model with the ELU activation, Adam optimizer, 1 layer bidirectional GRU with 60 units, with an additional dense layer at the end is on the top (run 123). The validation loss for this model is approximately 0.0005589, while the test loss is about 0.3772278. The second entry's (run 124) validation loss is only about $1.6 * 10^{-6}$ greater, which is a small difference. In contrast, this value has a significantly lower test loss, at about 0.3240174. Notice that this table has higher validation loss values than the previous table, which is due to the different data separation resulting in more tests being done.

If we take a look at the test loss, we can observe that the lowest validation loss does not correspond with the lowest test loss. Sorting by the test loss instead (second table, appendix B) gives a new top model with the ReLU activation, 3 GRU layers with 60 units, 0.2 dropout and the RMSprop optimizer (run 30). This model has an test loss of about 0.0653591, which is much lower than the model with the lowest validation loss. In fact, that model has the highest test loss. This does puts some of our reasoning behind not including RNN in this run in question, as we did not append the test loss to that result, though the time constraints still apply. In order to minimize the loss on the test set, we will use this model to represent NNs in the comparisons.

10.2 Comparison

Now we will present the end result of running all the scikit-learn algorithms we use in the project, the selected NN and the naive method on the same dataset. We also included the worst of the NN models here for comparison. All the methods were trained and made predictions on the test set 20 times. Then the average MSE for each method over 20 test rounds were calculated, the result are available in figure 10.1. These values were used to compare each of the methods against each other. Unfortunately, two of the simpler methods, lasso and elastic net, failed as a consequence of scaling the data to the interval $[0, 1]$ and where not included. These require further examination.

Method	MSE
Random forest	0.001137607845037226
Linear regression	0.0017986074770151577
Lars	0.0020867141309309583
Ridge	0.0016549403786762721
Naive	0.004172812167342442
GRU-RMSprop	0.0009579661211377005
GRU-Adam	0.0010289894909705287

Table 10.1: *Comparison of the average MSE for each method over 20 runs. 30 previous points are used to predict 10 points ahead, and it uses Min-Max Scaler from 0 to 1. Features used: CPU, Porter, Node Load 1, JVM Memory, Audit BaseEvent and JMS Consumer Messages*

As you can see on figure 10.2, the non-NN models result in predictions of mixed accuracy. However, the accuracy of the non-NN models is lower than the NNs. Furthermore, we can observe that the graphs on figure **??** of both the NNs are very similar, despite using both the worst and best results from Talos. As the NN chosen by sorting the Talos results by test loss did the best, we select it as our final model. Of the simple models Random Forest (figure 10.3) did the best, with a result not very far behind the NNs.

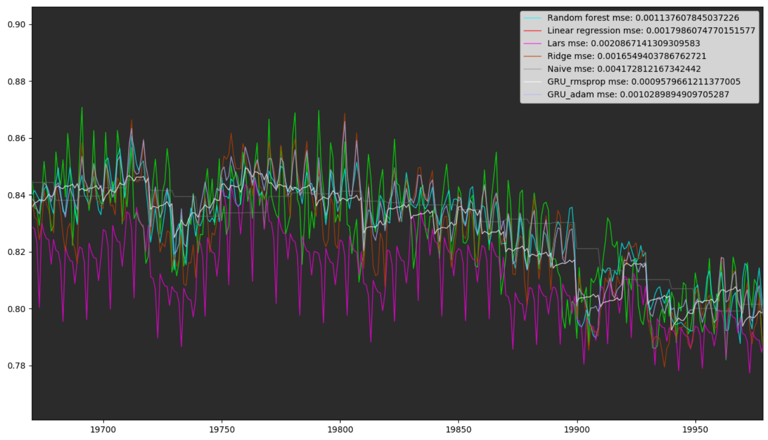


Figure 10.1: *A visual comparison of all methods' predictions averaged over 20 runs. The true values are visualized by the green line. Based on the same data as figure 10.1.*

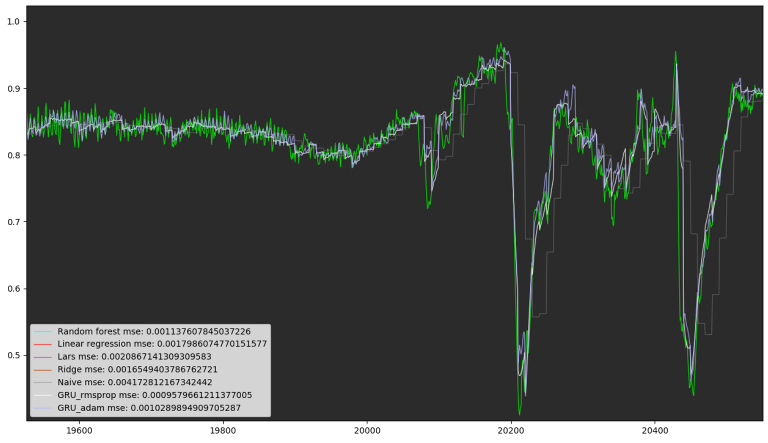


Figure 10.2: *A visual comparison of both NNs' and naive method predictions averaged over 20 runs. The true values is included here in order to clearly display the prediction intervals. The true values visualized by the green coloured line.*

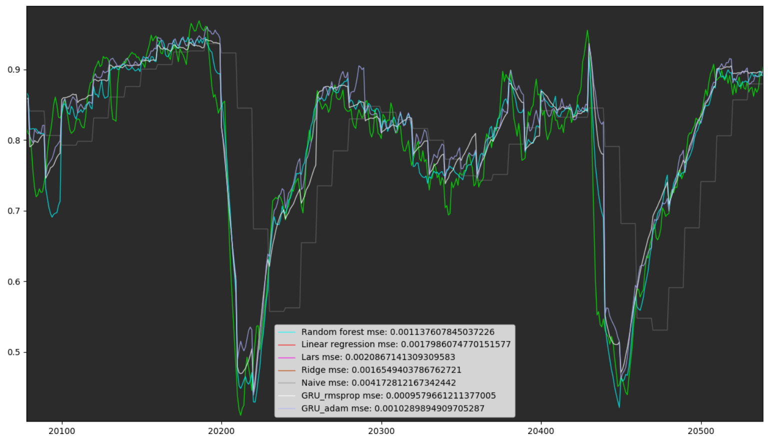


Figure 10.3: *A visual comparison of both NNs', Random Forest and naive method predictions averaged over 20 runs. The naive method is included here in order to clearly display the prediction intervals. The true values visualized by the green coloured line.*

On the graphs on figure 10.2 it seems as if the NNs make simpler predictions than the simple methods. This is due to the final predictions being a average of many and more varying predictions. This affects the simple methods less, as their results are more similar between runs. Note that on the above figures, each tenth prediction series was graphed, in order to ensure a good overview. Overall, none of the models appear to reliably "predict" significant divergence in the time series, despite having a quite low loss in total.

10.3 Final Model

Running the model once on a more extensive data set, results in a overall testing loss of about $2.68 * 10^{-4}$. The naive method achieved a loss of about $6.21 * 10^{-4}$ for the same dataset, which is a difference of $3.53 * 10^{-4}$. Figure 10.4 shows the loss graph for the model, which indicates that the model might not be complex or flexible enough to learn all the connections. Furthermore, on figure 10.5 are some example predictions on the CPU target variable.

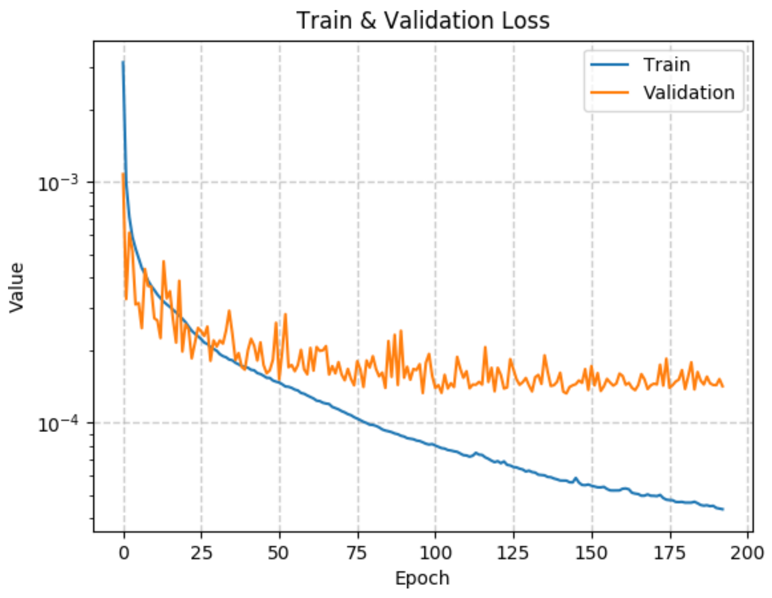
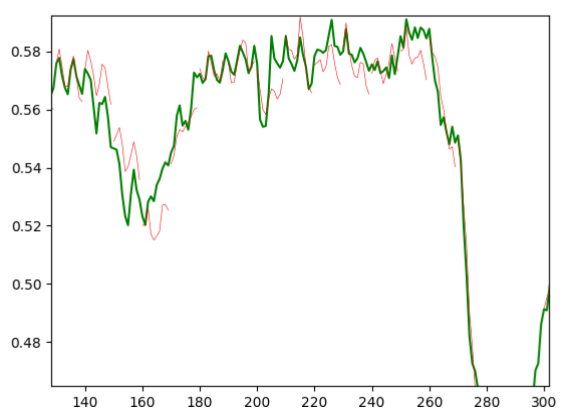
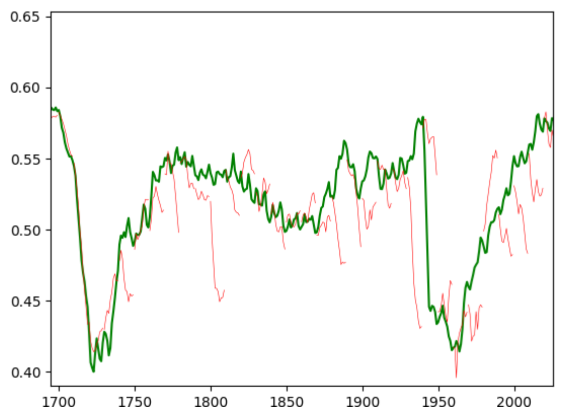


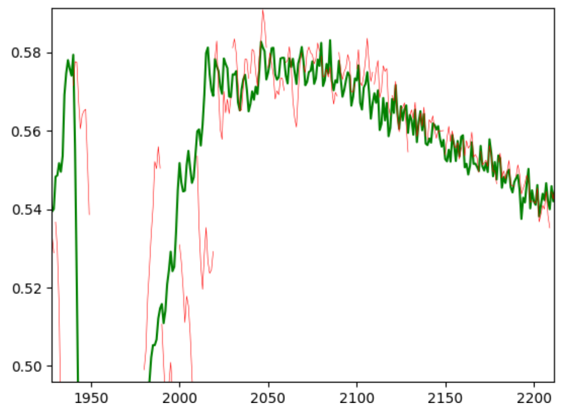
Figure 10.4: *A graph showing the learning of the model, illustrated by validation and training loss.*



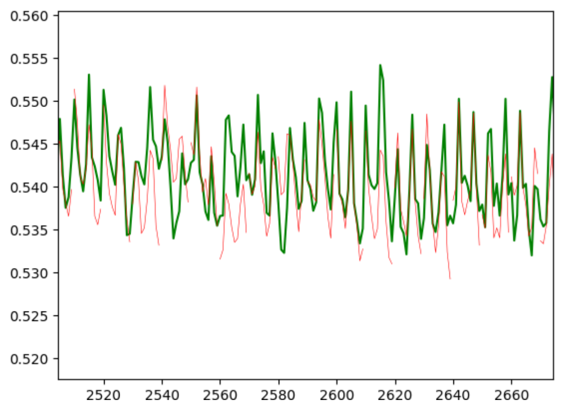
(a) *The model is not very far away from the actual values.*



(b) *In this section of the test set, the model makes many mistakes as to the progress of the true values.*



(c) *The predictions are missing the true values in the area with high change, though it does better on the area to the right.*



(d) *This is a relatively flat area of the test set. The model appears to recognize the rapidly swinging pattern.*

Figure 10.5: *Final model predictions on a test set. The green graph represents the true values, while the predictions are red.*

Chapter 11

Discussion Part 2

One of the most obvious considerations is the time resolution of the data. The speed at which variables can change within a computer system is much greater than more commonly forecast variables. For instance the weather, pressure and humidity outside rarely swing violently over such short periods of time. Therefore, selecting a low resolution might increase the chances of the model discovering correlation between the time series. Prometheus does not retrieve performance metrics often enough for us to use a resolution higher than 1 minute, so this would have to be measured by other means if it where to be used. On the other hand, the lower the resolution is, the harder it could be to predict values further in the future. In our implementation, the time between data points is always constant.

In the case of recursive predictions, a high resolution would cause each prediction step to affect a shorter span of time. This would require more recursive predictions to reach a reasonable amount of time, with the loss growing quickly for each recursion step. For non-recursive predictions this is much less of an issue, as the NN can be set to predict a specific number of points ahead in time.

In addition, the size of the dataset grows proportionally with the increase in resolution. It was a major limitation for us in the beginning of the project, because of memory limitations. We solved this partially by implementing feature selection, reducing the amount of features used in predicting the target to about 3-7. In addition, lower resolution might emphasize the long term changes, rather than the quick short term changes.

Our selection of features from the Prometheus service might not contain all the necessary information to accurately predict sudden changes in the time series. In addition, when considering that the data is often cyclical on the scale of days and weeks, there is a risk that the algorithm learns to repeat the cyclical pattern, rather than predict any interesting changes in the system behaviour.

The use of the optimization library Talos allowed us to test many possible combinations of hyperparameters in an orderly fashion, resulting in a good network for the task. Still, it is not certain that this is the best model, because we only had enough time to test small networks, and the amount of epochs was limited to 400 for the models. In addition, we did not try adding convolution and many other NN types that are compatible.

The table gained from using Talos allows us to see which NN setups work well, and which do not. There is one big flaw to our use of Talos, which is every model combination is only run once instead of 20-30. This means that the results are not reliable, as to get an accurate result with NNs one has to train them several times. This could explain why the worst NN did almost as good as the best, despite them having a completely different evaluation when run through Talos. Due to time constraints we did not address this issue properly, though our result should be enough for a proof of concept.

Since we only trained the models on one dataset, we can not be sure as to the validity of the results. Optimally, we should have used a larger dataset with Walk-Forward validation and multiple runs per window for a more reliable result. Additionally, testing with only CPU as the target feature is not enough in order to make conclusions about how well the other variables can be predicted.

Considering the significantly lower loss for the NN model, we can conclude that it is better suited for these time series than the simple methods. Still, seeing as the model often fails to predict radical changes in the 10-point intervals, it is difficult to trust that the model would do well in those cases. In order for the information to be useful, it would have to be reliable over time and make few mistakes. Though as seen in the analysis chapter, the model is on the right path, as some of the predictions appear to follow the true values quite well.

Chapter 12

Conclusion Part 2

We have compared a few variations of RNN along with several simpler scikit-learn methods in forecasting performance metrics of Signicat's services with CPU as the target feature.

Comparing the methods against each other revealed that the RNN model chosen by using Talos did in fact get the lowest loss, though the worst RNN model was not very far behind. Therefore we can conclude that RNNs, specifically GRUs, are a good choice for the time series we examined, though Random Forest also shows promise. On the other hand, the predictions do not appear to predict the anomalies in the time series, and those are likely the most interesting cases. Therefore, we can not say for sure whether the predictions are useful at this point, though the model is a good starting point for further work, and we conclude that RNNs can be applied to this problem.

12.1 Further Work

Our example was quite artificial, as the metrics were an average of several different aliases. A more realistic and accurate application would involve creating a separate model for every alias, and for every variable one should wish to predict within the alias.

In order to further improve the model, one could try more complicated neural network structures than what we had time to try, as some correlations can be out of reach for our simple models. Also, as mentioned before, acquiring data with higher resolution might allow for more detailed predictions. The rate at which changes occur in a computer system is much higher than the 1 minute interval between our data points.

In addition, there are more machine learning methods that could be applied to the problem. These could be tested, by using the code we already made with few alterations. In this report we only focused on a selected few, in order to make a proof of concept. Have we had more time we would implement the three other forecasting methods described in [9], namely Direct, DirRec and DIRMO, so that we could compare their performance to those we already implemented.

A natural continuation of this work would be to create models for other variables, and to explore whether the same type of network works well with them. It should be a simple matter of running the same code with slightly different parameters, along with large amounts of patience for the NN training, as some of the programs can take days to run.

List of Figures

2.1 *The output from hierarchical clustering is a tree diagram called dendrogram, which shows relationships based on similarity. [60]* 11
2.2 *Density based clustering with 3 minimum points. Blue = noise, yellow = border and red = core [61]* 12

3.1 *Color values represent contamination values. Blue points are closer to the norm, while red indicates high divergence from the rest. The dark-coloured points are somewhere in between.* 19
3.2 *dataset reduced with UMAP. Points more or less gather in clusters.* . 20
3.3 *The same dataset reduced with t-SNE, here the data points lose much more of the global structure, resulting in a less clear cloud of points.* . 20
3.4 *Robust Covariance* 20
3.5 *Isolation Forest* 20
3.6 *One-Class SVM, all the black points are considered outliers.* 21
3.7 *Local Outlier Factor* 21

4.1 *3D visualization of the dataset after dimension reduction with PCA. There is a distinct pattern here that implies geographical location plays an important role in deciding how divergent a point is. Keep in mind that most of the transactions originate from within Norway.* 22
4.2 *3D visualization of the dataset after dimension reduction with PCA. There is a distinct pattern here that implies geographical location plays an important role in deciding how divergent a point is. Keep in mind that most of the transactions originate from within Norway.* 23
4.3 *The result is achieved using UMAP, Robust Scaler and Isolation Forest. There are two different views, because the three-dimensional structure is hard to discern when viewing it from only one perspective.* 23
4.4 *Same as figure 4.3, though it uses Local Outlier Factor instead of Isolation Forest. The variance in the point distribution can be attributed to UMAP giving different results after each run.* 24

8.1 *A recursive prediction strategy.* 29
8.2 *A MIMO based prediction strategy.* 30
8.3 *A neural network with one hidden layer.* 30
8.4 *A RNN network.* 31
8.5 *A RNN cell.* 31
8.6 *An unfolded RNN network predicting on three samples.* 31
8.7 *LSTM* 32
8.8 *A GRU cell.* ***1.*** *is the reset gate.* ***2.*** *is the update gate. The structure around the cell is the same as in figure 8.6.* 32

8.9 *Hyperplane and margin for a SVM-model trained on two predictors [62]* 35
8.10 *Splitting of a dataset into training, validation and testing sets.* 37
8.11 *Random split of a dataset into a train, validation and test set.* 37
8.12 *Deterministic split of a dataset into a train, validation and test set.* . 37
8.13 *Expanding Window increases the amount of data used for training used each step. A square represents a "part". A segment is all the squares of the same colour in the figure.* 38
8.14 *Sliding Window keeps the amount of training data constant while moving the training window. A square represents a "part". A segment is the squares of the same colour in the figure.* 38

9.1 *An example of early stopping, the vertical dotted line indicates where the training is to be stopped.* 43

10.1 *A visual comparison of all methods' predictions averaged over 20 runs. The true values are visualized by the green line. Based on the same data as figure 10.1.* 46
10.2 *A visual comparison of both NNs' and naive method predictions averaged over 20 runs. The naive method is included here in order to clearly display the prediction intervals. The true values visualized by the green coloured line.* 46
10.3 *A visual comparison of both NNs', Random Forest and naive method predictions averaged over 20 runs. The naive method is included here in order to clearly display the prediction intervals. The true values visualized by the green coloured line.* 46
10.4 *A graph showing the learning of the model, illustrated by validation and training loss.* 47
10.5 *Final model predictions on a test set. The green graph represents the true values, while the predictions are red.* 48

List of Tables

4.1 *Outliers detected by Isolation Forest and which clusters the outliers belonged to, using KMeans with 5 clusters.* 24

9.1 *This table shows the MSE for CPU prediction after running the scikit-learn methods with and without feature selection. 30 points are used to predict 10 points, and it uses Min-Max Scaler from -3 to 3.* 42

10.1 *Comparison of the average MSE for each method over 20 runs. 30 previous points are used to predict 10 points ahead, and it uses Min-Max Scaler from 0 to 1. Features used: CPU, Porter, Node Load 1, JVM Memory, Audit BaseEvent and JMS Consumer Messages* 45

Bibliography

[1] *Sci-kit Learn Examples, Compare the effect of different scalers on data with outliers.* URL `https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html`.

[2] *Sci-kit Learn Documentation, Novelty and Outlier Detection*, 03 2019. URL `https://scikit-learn.org/stable/modules/outlier_detection.html`.

[3] 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, Dan 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. URL `http://tensorflow.org/`. Software available from tensorflow.org.

[4] Edgar Acuña and Carlos Rodríguez. A meta analysis study of outlier detection methods in classification. 2004.

[5] Ethem Alpaydin. *Introduction to Machine Learning.* The MIT Press, 2nd edition, 2010. ISBN 026201243X, 9780262012430.

[6] Autonomio. Talos. URL `https://github.com/autonomio/talos`.

[7] Joshua B. Tenenbaum, Vin Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 01 2000.

[8] Imad A. Basheer and Maha N. Hajmeer. Artificial neural networks: fundamentals, computing, design, and application. *Journal of microbiological methods*, 43 1:3–31, 2000.

[9] Souhaib Ben Taieb, Gianluca Bontempi, Amir Atiya, and Antti Sorjamaa. A review and comparison of strategies for multi-step ahead time series forecasting based on the nn5 forecasting competition. *Expert Systems with Applications*, 39, 08 2011. doi: 10.1016/j.eswa.2012.01.039.

[10] Pavel Berkhin. Survey of clustering data mining techniques. *A Survey of Clustering Data Mining Techniques. Grouping Multidimensional Data: Recent Advances in Clustering.*, 10, 08 2002. doi: 10.1007/3-540-28349-8_2.

[11] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001. ISSN 1573-0565. doi: 10.1023/A:1010933404324.

[12] Gunnar Carlsson and Facundo Mémoli. Characterization, stability and convergence of hierarchical clustering methods. *Journal of Machine Learning Research*, 11:1425–1470, 04 2010.

[13] François Chollet. keras. `https://github.com/fchollet/keras`, 2015.

[14] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Y Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. 12 2014.

[15] G E. Hinton. Visualizing high-dimensional data using t-sne. *Vigiliae Christianae*, 9:2579–2605, 01 2008.

[16] Jeffrey L. Elman. Finding structure in time. *COGNITIVE SCIENCE*, 14(2): 179–211, 1990.

[17] Martin Ester. *Density-based Clustering*, pages 795–799. Springer US, Boston, MA, 2009. ISBN 978-0-387-39940-9. doi: 10.1007/978-0-387-39940-9_605.

[18] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press, 1996.

[19] Tom Fawcett and Foster Provost. *Data Science for Business.* O'Reilly Media, Inc., 2013. ISBN 9781449374273.

[20] I.K. Fodor. A survey of dimension reduction techniques. *Meat Sci.*, 9:10–20, 01 2002. doi: 10.2172/15002155.

[21] J. A. Hartigan and M. A. Wong. Algorithm AS 136: A K-Means clustering algorithm. *Applied Statistics*, 28(1):100–108, 1979. ISSN 00359254. doi: 10.2307/2346830.

[22] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6:107–116, 04 1998. doi: 10.1142/S0218488598000094.

[23] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

[24] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

[25] Aapo Hyvrinen. Survey on independent component analysis. *Neural Computing Surveys*, 2, 07 1999.

[26] Eric Iturbide, Jaime Cerda, and Mario Graff. A comparison between lars and lasso for initialising the time-series forecasting auto-regressive equations. *Procedia Technology*, 7:282–288, 12 2013. doi: 10.1016/j.protcy.2013.04.035.

[27] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. URL `http://www.scipy.org/`.

[28] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 2342–2350. JMLR.org, 2015.

[29] Iebeling Kaastra and Milton Boyd. Designing a neural network for forecasting financial and economic time series. *Neurocomputing*, 10(3):215–236, 1996. doi: 10.1016/0925-2312(95)00039-9.

[30] Andrej Karpathy, Justin Johnson, and Fei Fei Li. Visualizing and understanding recurrent networks. *Cornell Univ. Lab.*, 06 2015.

[31] Kamaljeet Kaur and Atul Garg. Comparative study of outlier detection algorithms. *International Journal of Computer Applications*, 147:21–26, 08 2016. doi: 10.5120/ijca2016911176.

[32] Gülden Kaya Uyanık and Neşe Güler. A study on multiple linear regression analysis. *Procedia - Social and Behavioral Sciences*, 106:234–240, 12 2013. doi: 10.1016/j.sbspro.2013.12.027.

[33] Michelle Lacey. Multiple linear regression. URL `http://www.stat.yale.edu/Courses/1997-98/101/linmult.htm`.

[34] Ilias Maglogiannis Lazaros Iliadis and Harris Papadopoulos. *Artificial Intelligence Applications and Innovations*. Springer, 2014.

[35] Leo Liberti, Carlile Lavor, Nelson Maculan, and Antonio Mucherino. Euclidean distance geometry and applications. *SIAM Rev.*, 56(1):3–67, 2014. doi: 10.1137/120875909.

[36] F. T. Liu, K. M. Ting, and Z. Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422, Dec 2008. doi: 10.1109/ICDM.2008.17.

[37] Markus M. Breunig, Hans-Peter Kriegel, Raymond Ng, and Joerg Sander. Lof: Identifying density-based local outliers. volume 29, pages 93–104, 06 2000. doi: 10.1145/342009.335388.

[38] L. McInnes, J. Healy, and J. Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *ArXiv e-prints*, February 2018.

[39] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861, 2018.

[40] Qing He Ning Li1, Li Zeng and Zhongzhi Shi. Parallel implementation of apriori algorithm based on mapreduce, 08 2012.

[41] Salima Omar, Md Ngadi, Hamid H Jebur, and Salima Benqdara. Machine learning techniques for anomaly detection: An overview. *International Journal of Computer Applications*, 79, 10 2013. doi: 10.5120/13715-1478.

[42] S GOPAL PATRO and Kishore Kumar Sahu. Normalization: A preprocessing stage. *IARJSET*, 03 2015. doi: 10.17148/IARJSET.2015.2305.

[43] 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.

[44] 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. Sci-kit learn documentation, decision trees, 2019. URL `https://scikit-learn.org/stable/modules/tree.html`.

[45] Daniel Peña and Francisco Prieto. Multivariate outlier detection and robust covariance matrix estimation. *Technometrics*, 43, 08 2001. doi: 10.1198/004017001316975899.

[46] Ashis Pradhan. Support vector machine-a survey. *IJETAE*, 2, 09 2012. ISSN ISSN 2250-2459.

[47] Lutz Prechelt. Early stopping - but when? 03 2000. doi: 10.1007/3-540-49430-8_3.

[48] Peter Rousseeuw and Annick M. Leroy. *Robust Regression & Outlier Detection.* 09 1987. doi: 10.2307/2289958.

[49] Sebastian Ruder. An overview of gradient descent optimization algorithms. 09 2016.

[50] Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. Recent advances in recurrent neural networks. 12 2017.

[51] Bernhard Schölkopf, John C. Platt, John C. Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Comput.*, 13(7):1443–1471, July 2001. ISSN 0899-7667. doi: 10.1162/089976601750264965. URL `https://doi.org/10.1162/089976601750264965`.

[52] Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45:2673 – 2681, 12 1997. doi: 10.1109/78.650093.

[53] M Shridhar and Mahesh Parmar. Survey on association rule mining and its approaches. *International Journal of Computer Sciences and Engineering (IJCSE)*, 5:129–135, 03 2017.

[54] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, 4, 06 2012.

[55] Xiaomeng Su. *Business Analytics Techniques*. NTNU Blackboard, IINI3012 Big Data, 2018.

[56] W. M. P. van der Aalst, V. Rubin, H. M. W. Verbeek, B. F. van Dongen, E. Kindler, and C. W. Günther. Process mining: a two-step approach to balance between underfitting and overfitting. *Software & Systems Modeling*, 9(1):87, Nov 2008. ISSN 1619-1374. doi: 10.1007/s10270-008-0106-z.

[57] G. van Rossum. Python tutorial. Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, May 1995.

[58] Diego Vidaurre, Concha Bielza, and Pedro Larranaga. A survey of l1 regression. *International Statistical Review*, 81, 12 2013. doi: 10.1111/insr.12023.

[59] William Wei. *Time Series Analysis: Univariate and Multivariate Methods*, volume 33. 01 1989. ISBN 978-0-201-15911-0. doi: 10.2307/2289741.

[60] Wikimedia Commons, the free media repository. An agglomerative clustering dendogram example, 12 2009. URL `https://commons.wikimedia.org/wiki/File:Agglomerative_clustering_dendogram.png`. [Online; accessed April 20, 2019].

[61] Wikimedia Commons, the free media repository. Illustration of dbscan cluster analysis (minpts=3), 10 2011. URL `https://commons.wikimedia.org/wiki/File:DBSCAN-Illustration.svg`. [Online; accessed May 1, 2019].

[62] Wikimedia Commons, the free media repository. Maximum-margin hyperplane and margin for an svm trained on two classes. samples on margins are called support vectors., 10 2018. [Online; accessed May 10, 2019].

[63] Shuang Wu, Guoqi Li, Lei Deng, Liu Liu, Yuan Xie, and L.P. Shi. L1-norm batch normalization for efficient training of deep neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, PP, 02 2018. doi: 10.1109/TNNLS.2018.2876179.

[64] Dongkuan Xu and Yingjie Tian. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2):165–193, 2015. doi: 10.1007/s40745-015-0040-1.

[65] Xujun Zhou, Xianxia Zhang, and Bing Wang. Online support vector machine: A survey. 382:269–278, 01 2016. doi: 10.1007/978-3-662-47926-1_26.

Appendix A

Talos Run 1

The settings used by Talos:

- **Activation functions:** ReLU, ELU, tanh, sigmoid
- **Optimizers:** Nadam, Adam, rmsprop
- **Loss:** MSE
- **Layers:** 1, 2, 3
- **Size per layer:** 10, 30, 60, 120
- **Batch size:** 64, 128, 256
- **Bidirectional:** True, False
- **Layer type:** GRU, LSTM, RNN
- **Epochs:** 250, 300, 400

Sorted by validation loss

run	epochs	valLoss	loss	activation	optimizer	losses	layers	size	batchSize	bidirectional	layer
735	400	0.000044	0.000038	relu	Nadam	mse	1	60	64	True	gru
53	400	0.000048	0.000017	relu	Nadam	mse	3	120	64	True	gru
368	400	0.000048	0.000064	relu	rmsprop	mse	1	120	64	True	lstm
548	400	0.000048	0.000038	relu	Nadam	mse	1	60	64	True	lstm
751	250	0.000049	0.000053	tanh	Nadam	mse	3	60	64	True	lstm
253	400	0.000049	0.000041	elu	Nadam	mse	1	120	64	True	lstm
527	400	0.000049	0.000048	elu	Nadam	mse	2	120	64	True	lstm
710	400	0.000049	0.000042	relu	Adam	mse	3	120	64	False	gru
236	400	0.000049	0.000044	relu	Nadam	mse	1	120	64	True	lstm
123	250	0.000050	0.000046	relu	Adam	mse	2	120	64	True	gru
553	400	0.000050	0.000044	relu	Nadam	mse	1	120	64	False	lstm
637	400	0.000050	0.000041	tanh	Adam	mse	3	120	64	False	lstm
112	400	0.000051	0.000023	relu	Adam	mse	2	120	64	False	rnn
445	400	0.000051	0.000044	relu	Adam	mse	3	30	64	True	gru
426	400	0.000051	0.000048	elu	Nadam	mse	2	60	64	True	gru
472	300	0.000051	0.000089	relu	rmsprop	mse	2	120	64	True	lstm
297	250	0.000051	0.000052	relu	Adam	mse	3	60	64	True	gru
288	400	0.000051	0.000044	relu	Adam	mse	1	120	64	True	gru
159	400	0.000052	0.000062	elu	Adam	mse	3	120	64	False	gru
299	400	0.000052	0.000062	tanh	Adam	mse	2	30	64	True	lstm
286	400	0.000053	0.000056	relu	Adam	mse	2	60	64	False	gru
435	250	0.000053	0.000056	relu	Adam	mse	3	30	64	True	gru
524	400	0.000054	0.000064	elu	Adam	mse	2	120	64	False	gru
384	300	0.000054	0.000050	relu	Adam	mse	3	60	64	True	lstm
327	400	0.000055	0.000063	elu	Adam	mse	1	120	64	False	lstm
458	300	0.000055	0.000058	relu	Adam	mse	1	120	64	True	gru
214	250	0.000055	0.000062	relu	Adam	mse	1	60	64	True	gru
669	300	0.000055	0.000047	relu	Nadam	mse	3	30	64	True	gru
510	400	0.000056	0.000041	relu	Nadam	mse	1	60	64	True	rnn
731	300	0.000056	0.000069	elu	Nadam	mse	1	120	64	True	lstm
313	400	0.000056	0.000053	relu	Adam	mse	1	120	64	False	lstm
517	400	0.000056	0.000065	elu	Nadam	mse	2	30	64	True	rnn

run	epochs	valLoss	loss	activation	optimizer	losses	layers	size	batchSize	bidirectional	layer
708	300	0.000056	0.000059	tanh	Nadam	mse	1	60	64	True	gru
753	400	0.000056	0.000038	tanh	Nadam	mse	2	60	64	True	lstm
591	300	0.000057	0.000053	relu	Adam	mse	2	120	256	True	gru
218	400	0.000057	0.000053	elu	Nadam	mse	2	120	64	False	lstm
278	300	0.000057	0.000059	relu	Nadam	mse	1	60	64	False	rnn
141	300	0.000057	0.000069	tanh	Adam	mse	2	60	64	True	gru
663	250	0.000057	0.000046	relu	Adam	mse	2	60	64	True	lstm
383	400	0.000057	0.000032	relu	Nadam	mse	3	60	128	True	rnn
505	300	0.000057	0.000064	relu	Adam	mse	2	60	128	False	gru
728	300	0.000057	0.000073	tanh	Nadam	mse	2	30	64	False	gru
94	300	0.000058	0.000073	tanh	rmsprop	mse	3	30	64	False	lstm
453	250	0.000058	0.000060	relu	Adam	mse	2	120	128	True	lstm
182	400	0.000058	0.000073	elu	Adam	mse	2	60	128	False	gru
489	400	0.000058	0.000090	tanh	rmsprop	mse	3	120	64	False	gru
52	300	0.000058	0.000067	elu	Adam	mse	1	120	64	True	rnn
89	400	0.000058	0.000073	elu	Nadam	mse	3	10	64	True	gru
198	300	0.000058	0.000077	elu	Adam	mse	3	60	128	False	gru
175	400	0.000058	0.000068	tanh	Adam	mse	2	30	64	False	gru
259	400	0.000058	0.000061	relu	Nadam	mse	1	30	128	True	lstm
280	400	0.000058	0.000022	relu	Adam	mse	2	60	64	True	rnn
24	400	0.000058	0.000064	relu	Adam	mse	2	60	256	False	gru
399	300	0.000058	0.000073	tanh	Adam	mse	3	30	64	False	gru
592	250	0.000059	0.000086	tanh	rmsprop	mse	3	30	64	True	lstm
56	250	0.000059	0.000077	relu	Adam	mse	3	10	64	True	gru
337	400	0.000059	0.000067	relu	Nadam	mse	1	30	64	False	gru
210	300	0.000059	0.000080	elu	rmsprop	mse	3	120	64	True	lstm
17	250	0.000059	0.000080	elu	Nadam	mse	2	60	64	True	lstm
385	300	0.000059	0.000082	relu	rmsprop	mse	2	60	64	False	lstm
485	400	0.000059	0.000072	relu	Adam	mse	2	30	64	False	rnn
567	400	0.000060	0.000080	tanh	rmsprop	mse	1	120	64	False	gru
5	250	0.000060	0.000077	elu	Adam	mse	3	60	64	True	gru
564	300	0.000060	0.000075	elu	Adam	mse	2	30	64	False	gru
508	400	0.000060	0.000070	elu	Adam	mse	1	30	64	False	gru
655	400	0.000060	0.000062	relu	Nadam	mse	3	30	64	False	gru
204	400	0.000060	0.000073	elu	Adam	mse	3	120	128	True	gru
74	400	0.000060	0.000077	tanh	rmsprop	mse	2	60	128	False	lstm
321	400	0.000060	0.000038	relu	Adam	mse	2	120	128	False	rnn
86	400	0.000060	0.000062	elu	Nadam	mse	2	60	64	True	lstm
217	400	0.000060	0.000068	relu	Nadam	mse	1	30	64	False	lstm
771	300	0.000060	0.000104	elu	rmsprop	mse	2	60	64	True	lstm
375	400	0.000060	0.000073	elu	Nadam	mse	3	120	64	False	lstm
4	300	0.000060	0.000067	relu	Nadam	mse	2	30	64	False	rnn
47	300	0.000061	0.000075	elu	Nadam	mse	2	30	64	True	lstm
155	250	0.000061	0.000078	relu	Adam	mse	2	10	64	True	gru
179	400	0.000061	0.000070	tanh	Nadam	mse	3	10	64	True	gru
737	400	0.000061	0.000053	tanh	Adam	mse	3	60	64	True	rnn
543	400	0.000061	0.000057	relu	rmsprop	mse	2	60	64	False	lstm
613	300	0.000061	0.000077	tanh	Adam	mse	2	60	128	False	gru
712	300	0.000061	0.000070	tanh	Nadam	mse	1	60	64	True	lstm
599	400	0.000061	0.000056	elu	Adam	mse	3	60	64	True	lstm
260	400	0.000061	0.000078	elu	Adam	mse	3	120	256	False	gru
740	400	0.000061	0.000060	tanh	Nadam	mse	3	60	128	True	lstm
590	400	0.000062	0.000072	tanh	Adam	mse	1	60	128	True	lstm
683	400	0.000062	0.000073	relu	rmsprop	mse	3	60	64	False	gru
65	250	0.000062	0.000059	relu	Nadam	mse	2	60	128	True	lstm
526	300	0.000062	0.000067	relu	Adam	mse	3	60	128	False	gru
520	400	0.000062	0.000090	tanh	rmsprop	mse	3	30	128	True	lstm
266	300	0.000062	0.000071	tanh	Nadam	mse	1	120	64	False	lstm
262	400	0.000062	0.000074	elu	Nadam	mse	3	30	128	True	gru
579	400	0.000062	0.000029	relu	Nadam	mse	1	120	128	True	rnn
353	300	0.000062	0.000082	elu	Adam	mse	3	120	128	True	gru
16	400	0.000062	0.000062	elu	Nadam	mse	3	120	64	True	lstm
36	400	0.000062	0.000067	tanh	Adam	mse	1	120	128	False	gru
177	250	0.000062	0.000070	relu	Adam	mse	3	30	128	True	gru
516	400	0.000063	0.000064	relu	rmsprop	mse	1	120	64	True	rnn
125	300	0.000063	0.000066	relu	Adam	mse	2	30	64	False	gru
671	400	0.000063	0.000064	relu	Adam	mse	3	30	128	True	rnn
300	250	0.000063	0.000073	elu	Nadam	mse	3	60	64	False	lstm
608	400	0.000063	0.000088	elu	rmsprop	mse	3	60	128	True	lstm
365	250	0.000063	0.000076	relu	Nadam	mse	3	60	64	True	lstm
507	300	0.000063	0.000082	relu	Adam	mse	1	10	64	True	gru
673	250	0.000063	0.000085	relu	rmsprop	mse	1	120	64	True	gru
95	250	0.000063	0.000062	relu	Adam	mse	3	120	128	True	lstm
456	250	0.000063	0.000118	relu	rmsprop	mse	3	120	64	True	gru
146	300	0.000063	0.000069	relu	Adam	mse	1	60	64	False	rnn
506	250	0.000064	0.000074	elu	Nadam	mse	2	60	64	True	rnn
499	400	0.000064	0.000064	tanh	Nadam	mse	3	10	64	True	lstm
252	300	0.000064	0.000068	tanh	Adam	mse	3	60	64	True	rnn
341	250	0.000064	0.000075	elu	Nadam	mse	2	60	64	False	gru
326	400	0.000064	0.000057	relu	Nadam	mse	1	60	128	True	rnn
319	400	0.000064	0.000075	elu	Nadam	mse	1	10	64	False	lstm
349	400	0.000064	0.000069	tanh	Nadam	mse	3	30	128	True	lstm
763	400	0.000064	0.000076	relu	Adam	mse	3	30	256	False	gru
699	300	0.000064	0.000078	elu	Adam	mse	2	120	128	True	gru
163	300	0.000064	0.000079	tanh	Adam	mse	3	30	64	True	rnn
486	400	0.000064	0.000073	relu	Nadam	mse	3	10	64	True	lstm
448	300	0.000064	0.000077	relu	rmsprop	mse	3	120	64	False	lstm
34	300	0.000064	0.000085	tanh	Adam	mse	1	120	64	False	rnn
98	400	0.000064	0.000078	relu	Nadam	mse	1	120	256	False	lstm
642	400	0.000064	0.000076	elu	Adam	mse	2	60	256	False	gru
224	300	0.000065	0.000072	relu	Nadam	mse	3	10	64	True	gru
129	250	0.000065	0.000067	relu	Nadam	mse	1	120	64	False	rnn
638	250	0.000065	0.000052	relu	Nadam	mse	3	30	64	True	rnn
350	300	0.000065	0.000082	elu	Adam	mse	3	10	64	False	gru
724	300	0.000065	0.000068	relu	Adam	mse	2	60	256	True	gru
658	300	0.000065	0.000072	elu	Adam	mse	2	30	64	False	lstm
572	250	0.000065	0.000077	relu	Nadam	mse	1	30	64	True	rnn
414	300	0.000065	0.000080	elu	Adam	mse	3	10	64	True	gru

run	epochs	valLoss	loss	activation	optimizer	losses	layers	size	batchSize	bidirectional	layer
376	400	0.000065	0.000079	relu	Adam	mse	2	10	64	False	gru
27	300	0.000065	0.000081	relu	Adam	mse	2	30	128	False	gru
454	400	0.000065	0.000090	elu	rmsprop	mse	3	30	128	False	lstm
629	300	0.000065	0.000077	elu	Adam	mse	2	120	128	True	lstm
429	250	0.000065	0.000044	relu	Nadam	mse	2	120	64	False	rnn
117	400	0.000065	0.000087	elu	Nadam	mse	2	10	64	False	rnn
212	250	0.000065	0.000083	relu	Adam	mse	2	10	64	False	gru
100	300	0.000065	0.000089	relu	rmsprop	mse	1	60	64	True	rnn
164	400	0.000066	0.000078	elu	Adam	mse	1	120	128	False	rnn
336	250	0.000066	0.000074	relu	Adam	mse	3	120	256	False	gru
104	400	0.000066	0.000026	relu	Nadam	mse	3	60	64	False	rnn
765	300	0.000066	0.000082	elu	Adam	mse	3	60	256	False	gru
645	400	0.000066	0.000053	relu	Nadam	mse	2	60	128	False	rnn
415	400	0.000066	0.000084	relu	Adam	mse	2	30	128	False	rnn
589	400	0.000066	0.000081	elu	Nadam	mse	3	30	64	False	rnn
555	250	0.000066	0.000082	relu	Nadam	mse	2	120	128	False	gru
377	250	0.000066	0.000082	relu	Adam	mse	1	60	64	False	rnn
755	400	0.000066	0.000074	relu	Adam	mse	1	10	64	False	gru
115	300	0.000066	0.000077	tanh	Nadam	mse	2	10	64	True	gru
102	250	0.000066	0.000083	tanh	Adam	mse	3	120	256	False	gru
504	250	0.000067	0.000083	tanh	Nadam	mse	2	30	64	False	gru
393	400	0.000067	0.000105	tanh	rmsprop	mse	2	60	128	False	gru
250	250	0.000067	0.000072	relu	Nadam	mse	1	120	128	True	lstm
407	300	0.000067	0.000084	tanh	Adam	mse	2	30	64	True	rnn
103	400	0.000067	0.000081	elu	Adam	mse	3	30	256	False	gru
432	400	0.000067	0.000104	relu	rmsprop	mse	1	10	64	True	rnn
660	250	0.000067	0.000084	elu	Adam	mse	3	30	64	False	lstm
226	300	0.000067	0.000074	relu	Adam	mse	3	30	128	False	gru
726	250	0.000067	0.000128	elu	rmsprop	mse	1	60	64	True	rnn
314	300	0.000067	0.000085	elu	Adam	mse	1	60	64	False	rnn
202	300	0.000067	0.000081	relu	Adam	mse	3	10	64	False	gru
263	400	0.000068	0.000073	tanh	Nadam	mse	2	30	128	False	lstm
761	400	0.000068	0.000082	relu	rmsprop	mse	3	10	64	True	lstm
720	400	0.000068	0.000119	relu	rmsprop	mse	3	10	64	False	gru
185	300	0.000068	0.000085	elu	rmsprop	mse	2	30	64	False	lstm
77	300	0.000068	0.000073	relu	Nadam	mse	1	60	128	False	rnn
533	300	0.000068	0.000083	elu	Adam	mse	2	10	64	True	gru
197	400	0.000068	0.000097	elu	rmsprop	mse	1	30	64	True	gru
622	400	0.000068	0.000073	relu	Nadam	mse	2	30	256	True	gru
754	250	0.000068	0.000082	elu	Adam	mse	3	60	256	False	gru
741	300	0.000068	0.000070	relu	Adam	mse	3	30	128	False	rnn
509	400	0.000068	0.000085	relu	Adam	mse	1	10	64	True	rnn
48	250	0.000069	0.000076	relu	Nadam	mse	2	60	256	True	gru
235	400	0.000069	0.000081	elu	Adam	mse	3	60	256	True	gru
285	300	0.000069	0.000082	elu	Nadam	mse	1	10	64	False	gru
281	300	0.000069	0.000073	relu	Nadam	mse	3	120	256	True	gru
97	300	0.000069	0.000073	relu	Nadam	mse	3	120	128	False	gru
362	300	0.000069	0.000085	relu	Nadam	mse	1	10	64	True	lstm
661	300	0.000069	0.000082	elu	Nadam	mse	1	120	128	True	lstm
223	400	0.000069	0.000089	elu	Adam	mse	3	30	64	False	rnn
342	400	0.000069	0.000132	relu	rmsprop	mse	2	30	128	True	gru
41	400	0.000069	0.000071	relu	Nadam	mse	2	120	256	False	gru
267	300	0.000069	0.000083	tanh	Nadam	mse	1	10	64	True	gru
371	400	0.000070	0.000094	elu	rmsprop	mse	3	120	64	False	rnn
614	300	0.000070	0.000091	tanh	Nadam	mse	1	30	64	True	rnn
758	250	0.000070	0.000083	tanh	Adam	mse	1	120	256	False	gru
49	250	0.000070	0.000090	elu	Adam	mse	2	10	64	True	gru
358	300	0.000070	0.000090	tanh	Nadam	mse	2	30	128	False	gru
379	250	0.000070	0.000102	relu	rmsprop	mse	2	30	64	False	lstm
623	400	0.000070	0.000088	elu	Nadam	mse	1	30	128	False	rnn
618	250	0.000070	0.000085	elu	Nadam	mse	1	10	64	True	gru
340	250	0.000070	0.000084	tanh	Adam	mse	1	60	64	False	lstm
63	250	0.000070	0.000101	elu	rmsprop	mse	2	120	64	True	rnn
749	400	0.000070	0.000077	tanh	Adam	mse	3	60	128	True	rnn
121	250	0.000070	0.000084	elu	Adam	mse	3	10	64	False	gru
96	250	0.000070	0.000078	relu	Adam	mse	3	30	128	False	rnn
596	400	0.000070	0.000081	relu	Nadam	mse	1	10	64	True	lstm
247	400	0.000071	0.000082	tanh	Nadam	mse	1	30	128	True	lstm
729	300	0.000071	0.000111	elu	rmsprop	mse	1	30	64	False	gru
12	400	0.000071	0.000116	elu	rmsprop	mse	1	120	128	False	rnn
575	300	0.000071	0.000084	tanh	Nadam	mse	2	120	128	False	lstm
662	300	0.000071	0.000089	elu	Nadam	mse	2	60	128	True	lstm
308	400	0.000071	0.000084	elu	Nadam	mse	1	30	128	True	lstm
692	250	0.000071	0.000081	tanh	Adam	mse	2	120	128	True	lstm
114	250	0.000071	0.000077	tanh	Adam	mse	3	30	64	True	lstm
160	250	0.000072	0.000097	relu	rmsprop	mse	3	60	64	False	rnn
145	250	0.000072	0.000089	tanh	Nadam	mse	1	60	128	False	gru
513	250	0.000072	0.000075	relu	Nadam	mse	1	120	128	False	lstm
382	250	0.000072	0.000089	elu	Adam	mse	3	10	64	True	gru
594	400	0.000072	0.000085	relu	Nadam	mse	2	30	128	False	lstm
400	400	0.000072	0.000069	relu	Nadam	mse	3	60	256	True	gru
747	250	0.000072	0.000116	elu	rmsprop	mse	1	10	64	False	gru
44	400	0.000072	0.000109	tanh	rmsprop	mse	2	10	64	True	gru
650	250	0.000072	0.000076	relu	Adam	mse	1	60	256	True	lstm
440	300	0.000073	0.000091	relu	Nadam	mse	1	10	128	False	rnn
80	300	0.000073	0.000089	elu	Adam	mse	1	120	128	False	lstm
500	400	0.000073	0.000083	sigmoid	Nadam	mse	2	120	64	True	lstm
144	250	0.000073	0.000086	tanh	Adam	mse	3	30	64	True	rnn
29	300	0.000073	0.000116	elu	rmsprop	mse	1	10	64	True	gru
335	300	0.000073	0.000083	relu	Adam	mse	2	10	64	False	gru
678	250	0.000073	0.000082	relu	Adam	mse	3	30	256	False	rnn
32	400	0.000073	0.000088	elu	Nadam	mse	2	60	256	False	gru
397	250	0.000073	0.000117	tanh	rmsprop	mse	2	60	64	False	gru
58	300	0.000073	0.000100	relu	Nadam	mse	1	10	128	True	gru
60	250	0.000073	0.000085	relu	Nadam	mse	1	120	256	True	gru
279	400	0.000073	0.000121	tanh	rmsprop	mse	1	10	128	False	gru
559	250	0.000073	0.000139	elu	rmsprop	mse	2	10	64	True	gru
255	250	0.000073	0.000087	tanh	Adam	mse	1	30	128	True	gru

run	epochs	valLoss	loss	activation	optimizer	losses	layers	size	batchSize	bidirectional	layer
334	250	0.000074	0.000115	relu	rmsprop	mse	3	10	64	True	lstm
607	300	0.000074	0.000090	elu	Nadam	mse	2	30	128	True	lstm
275	300	0.000074	0.000090	elu	Adam	mse	3	30	256	False	gru
261	400	0.000074	0.000089	elu	Adam	mse	1	10	64	False	lstm
684	400	0.000074	0.000092	relu	Nadam	mse	2	10	64	False	rnn
1	250	0.000074	0.000080	tanh	Adam	mse	1	120	256	True	gru
28	250	0.000074	0.000103	relu	rmsprop	mse	2	60	64	False	lstm
7	400	0.000074	0.000091	tanh	Nadam	mse	2	120	256	False	gru
478	300	0.000074	0.000121	elu	rmsprop	mse	3	30	64	True	rnn
632	250	0.000074	0.000146	elu	rmsprop	mse	1	30	128	False	rnn
651	400	0.000074	0.000077	tanh	Nadam	mse	3	30	64	True	rnn
462	400	0.000074	0.000088	relu	Adam	mse	1	60	256	False	rnn
705	400	0.000074	0.000133	tanh	rmsprop	mse	2	30	128	True	gru
733	250	0.000074	0.000085	tanh	Adam	mse	3	60	128	True	lstm
471	300	0.000075	0.000102	tanh	Adam	mse	2	60	128	True	rnn
408	400	0.000075	0.000092	elu	Nadam	mse	2	10	128	True	gru
339	400	0.000075	0.000087	tanh	Adam	mse	3	30	256	True	lstm
93	300	0.000075	0.000104	elu	Nadam	mse	3	10	64	False	rnn
549	300	0.000075	0.000143	tanh	rmsprop	mse	2	10	64	True	gru
681	300	0.000075	0.000142	elu	rmsprop	mse	2	30	128	True	lstm
552	300	0.000075	0.000096	tanh	Adam	mse	3	60	128	True	rnn
143	300	0.000075	0.000086	tanh	Nadam	mse	1	120	64	False	rnn
184	250	0.000075	0.000067	relu	Nadam	mse	1	120	128	True	rnn
374	250	0.000076	0.000123	tanh	rmsprop	mse	2	60	128	False	lstm
419	250	0.000076	0.000126	elu	rmsprop	mse	1	120	64	False	rnn
437	400	0.000076	0.000086	elu	Adam	mse	2	120	256	False	lstm
178	250	0.000076	0.000101	tanh	Adam	mse	3	30	128	True	rnn
654	300	0.000076	0.000130	elu	rmsprop	mse	2	120	128	True	rnn
290	400	0.000076	0.000092	relu	Nadam	mse	1	10	128	False	rnn
196	400	0.000076	0.000119	elu	rmsprop	mse	1	30	128	False	lstm
122	250	0.000076	0.000123	elu	rmsprop	mse	1	60	64	False	lstm
631	400	0.000076	0.000102	sigmoid	Nadam	mse	3	30	64	True	rnn
612	250	0.000076	0.000094	tanh	Nadam	mse	2	120	256	False	gru
35	400	0.000077	0.000104	elu	rmsprop	mse	2	120	128	True	lstm
201	250	0.000077	0.000209	tanh	rmsprop	mse	2	120	128	True	gru
580	300	0.000077	0.000092	elu	Adam	mse	1	60	128	True	rnn
676	300	0.000077	0.000120	tanh	rmsprop	mse	2	30	64	False	gru
55	250	0.000077	0.000097	elu	Nadam	mse	2	120	128	True	gru
19	300	0.000077	0.000170	elu	rmsprop	mse	3	120	128	True	rnn
546	250	0.000077	0.000085	relu	Adam	mse	3	120	256	False	lstm
403	400	0.000077	0.000093	sigmoid	Adam	mse	3	30	64	True	gru
518	250	0.000077	0.000089	tanh	Adam	mse	1	30	64	True	lstm
583	300	0.000077	0.000129	relu	rmsprop	mse	1	60	128	False	lstm
696	400	0.000077	0.000086	elu	Adam	mse	2	10	64	True	lstm
386	250	0.000077	0.000172	elu	rmsprop	mse	1	60	128	False	rnn
37	250	0.000077	0.000082	relu	Adam	mse	2	30	256	True	gru
424	250	0.000077	0.000069	relu	Adam	mse	3	60	256	True	lstm
15	250	0.000078	0.000118	elu	rmsprop	mse	3	30	128	False	lstm
457	250	0.000078	0.000104	relu	Nadam	mse	1	10	128	False	gru
562	250	0.000078	0.000156	tanh	rmsprop	mse	2	120	64	True	gru
477	250	0.000078	0.000118	tanh	rmsprop	mse	3	120	128	True	lstm
773	250	0.000078	0.000133	elu	rmsprop	mse	3	120	64	True	rnn
106	400	0.000078	0.000099	tanh	rmsprop	mse	3	120	256	False	lstm
373	400	0.000078	0.000093	elu	Nadam	mse	2	30	128	True	rnn
680	400	0.000078	0.000139	tanh	rmsprop	mse	2	60	128	True	gru
233	400	0.000078	0.000092	tanh	Nadam	mse	3	60	64	False	rnn
588	250	0.000078	0.000090	tanh	Adam	mse	2	30	64	False	lstm
719	400	0.000078	0.000105	sigmoid	Nadam	mse	2	30	64	False	lstm
441	400	0.000078	0.000105	sigmoid	Adam	mse	1	120	64	False	gru
234	400	0.000079	0.000117	relu	rmsprop	mse	1	30	256	False	gru
176	400	0.000079	0.000107	sigmoid	Nadam	mse	1	30	64	False	rnn
421	400	0.000079	0.000117	elu	rmsprop	mse	3	120	128	True	gru
294	250	0.000079	0.000087	tanh	Nadam	mse	2	60	128	True	lstm
444	250	0.000079	0.000123	elu	rmsprop	mse	1	120	64	False	lstm
2	300	0.000079	0.000094	tanh	Nadam	mse	1	120	128	True	rnn
251	400	0.000079	0.000096	tanh	Adam	mse	1	60	128	True	rnn
91	250	0.000079	0.000130	relu	rmsprop	mse	3	10	64	True	gru
571	250	0.000079	0.000100	tanh	Adam	mse	2	30	128	True	rnn
563	300	0.000079	0.000100	elu	Nadam	mse	3	60	128	False	gru
531	300	0.000079	0.000121	tanh	rmsprop	mse	3	10	64	True	gru
355	300	0.000079	0.000097	relu	Adam	mse	1	10	128	False	lstm
206	400	0.000079	0.000096	tanh	Adam	mse	2	60	128	False	rnn
706	250	0.000079	0.000122	elu	rmsprop	mse	2	120	128	True	rnn
578	250	0.000079	0.000091	elu	Adam	mse	2	60	128	False	lstm
438	400	0.000079	0.000104	sigmoid	Nadam	mse	2	30	64	True	lstm
558	400	0.000080	0.000144	sigmoid	rmsprop	mse	3	120	64	False	lstm
116	400	0.000080	0.000098	sigmoid	Nadam	mse	2	60	64	False	lstm
139	300	0.000080	0.000159	relu	rmsprop	mse	2	30	128	True	rnn
170	400	0.000080	0.000099	elu	Nadam	mse	1	30	256	False	gru
394	400	0.000080	0.000087	tanh	Adam	mse	3	60	256	True	lstm
213	300	0.000080	0.000094	elu	Adam	mse	1	10	64	False	lstm
422	300	0.000080	0.000143	tanh	rmsprop	mse	1	30	128	False	gru
484	400	0.000080	0.000090	elu	Nadam	mse	3	120	128	True	gru
430	300	0.000080	0.000099	tanh	Adam	mse	3	120	128	True	rnn
404	250	0.000080	0.000092	tanh	Adam	mse	1	120	128	False	lstm
501	400	0.000080	0.000096	tanh	Nadam	mse	1	120	64	False	rnn
282	300	0.000080	0.000122	elu	rmsprop	mse	2	60	128	False	lstm
14	400	0.000080	0.000094	tanh	Adam	mse	1	120	256	False	lstm
153	250	0.000080	0.000091	tanh	Nadam	mse	3	30	128	True	lstm
346	250	0.000081	0.000083	relu	Nadam	mse	1	120	256	False	gru
682	250	0.000081	0.000097	relu	Nadam	mse	2	30	256	False	gru
475	400	0.000081	0.000109	elu	Adam	mse	3	60	128	True	rnn
381	250	0.000081	0.000141	tanh	rmsprop	mse	1	120	128	False	gru
264	250	0.000081	0.000086	relu	Nadam	mse	1	30	128	True	gru
147	250	0.000081	0.000094	elu	Adam	mse	2	10	64	True	lstm
476	400	0.000081	0.000101	tanh	Nadam	mse	1	30	256	True	lstm
551	300	0.000081	0.000099	elu	Nadam	mse	1	60	128	False	rnn
150	300	0.000081	0.000090	relu	Nadam	mse	2	30	128	False	rnn

run	epochs	valLoss	loss	activation	optimizer	losses	layers	size	batchSize	bidirectional	layer
244	300	0.000081	0.000088	tanh	Nadam	mse	3	30	64	False	rnn
276	250	0.000082	0.000101	tanh	Nadam	mse	3	10	128	False	gru
292	300	0.000082	0.000124	elu	rmsprop	mse	2	10	64	True	gru
630	400	0.000082	0.000101	sigmoid	Adam	mse	3	120	64	True	gru
759	250	0.000082	0.000142	elu	rmsprop	mse	2	120	128	False	gru
693	300	0.000082	0.000095	tanh	Nadam	mse	1	60	256	True	gru
428	250	0.000082	0.000140	relu	rmsprop	mse	3	10	64	True	rnn
467	250	0.000082	0.000100	elu	Nadam	mse	1	10	64	True	lstm
668	250	0.000082	0.000137	elu	rmsprop	mse	2	30	128	False	lstm
332	400	0.000082	0.000104	tanh	Nadam	mse	3	120	64	True	gru
330	400	0.000083	0.000096	elu	Adam	mse	2	10	128	False	gru
750	250	0.000083	0.000127	tanh	rmsprop	mse	2	60	128	True	lstm
136	250	0.000083	0.000166	tanh	rmsprop	mse	1	120	256	True	gru
82	250	0.000083	0.000106	relu	Nadam	mse	2	10	128	True	lstm
246	300	0.000083	0.000102	sigmoid	Adam	mse	3	60	64	True	gru
309	300	0.000083	0.000117	tanh	rmsprop	mse	2	10	64	False	lstm
128	300	0.000083	0.000096	tanh	Adam	mse	3	10	128	True	lstm
154	400	0.000084	0.000103	sigmoid	Adam	mse	3	30	64	False	gru
205	250	0.000084	0.000099	elu	Nadam	mse	3	30	128	False	gru
626	300	0.000084	0.000111	elu	Nadam	mse	3	10	64	True	rnn
514	300	0.000084	0.000096	tanh	Adam	mse	3	30	256	True	gru
193	400	0.000084	0.000103	elu	rmsprop	mse	2	120	256	False	gru
124	300	0.000084	0.000102	elu	Nadam	mse	2	120	256	True	lstm
767	250	0.000084	0.000095	elu	Nadam	mse	3	60	128	True	lstm
416	400	0.000084	0.000091	tanh	Nadam	mse	3	10	128	False	gru
610	300	0.000085	0.000129	tanh	rmsprop	mse	1	120	128	False	lstm
135	250	0.000085	0.000126	elu	rmsprop	mse	2	10	64	False	lstm
523	400	0.000085	0.000106	tanh	Adam	mse	3	60	128	False	rnn
151	300	0.000085	0.000098	tanh	Adam	mse	1	60	256	True	lstm
302	400	0.000085	0.000106	sigmoid	Adam	mse	2	30	64	True	gru
343	250	0.000085	0.000100	elu	Adam	mse	2	30	256	True	gru
725	400	0.000085	0.000143	elu	rmsprop	mse	3	60	128	True	gru
23	250	0.000085	0.000152	tanh	rmsprop	mse	1	120	128	False	lstm
521	300	0.000085	0.000122	tanh	rmsprop	mse	1	60	64	True	rnn
616	400	0.000085	0.000104	elu	Nadam	mse	1	10	128	False	lstm
694	250	0.000086	0.000096	tanh	Adam	mse	2	10	64	True	lstm
50	250	0.000086	0.000133	relu	rmsprop	mse	1	60	128	True	rnn
257	250	0.000086	0.000096	elu	Adam	mse	1	60	128	False	lstm
173	250	0.000086	0.000108	relu	Nadam	mse	2	10	256	False	rnn
99	250	0.000086	0.000155	relu	rmsprop	mse	3	30	128	True	lstm
704	300	0.000087	0.000126	relu	rmsprop	mse	1	10	128	True	gru
88	400	0.000087	0.000116	sigmoid	Adam	mse	3	120	64	True	rnn
372	300	0.000087	0.000099	elu	Nadam	mse	3	120	128	True	lstm
199	400	0.000087	0.000116	sigmoid	Nadam	mse	1	120	64	True	lstm
691	400	0.000087	0.000123	sigmoid	Nadam	mse	1	120	64	True	rnn
405	300	0.000087	0.000099	tanh	Nadam	mse	3	120	128	False	gru
296	300	0.000087	0.000105	elu	Adam	mse	2	30	128	False	rnn
73	250	0.000087	0.000104	elu	Adam	mse	2	30	256	False	gru
57	400	0.000087	0.000095	tanh	Nadam	mse	1	120	128	True	rnn
191	250	0.000087	0.000104	tanh	Nadam	mse	2	30	256	False	gru
530	300	0.000087	0.000120	sigmoid	Nadam	mse	3	30	64	False	rnn
703	300	0.000087	0.000095	relu	Nadam	mse	1	60	256	True	lstm
229	400	0.000087	0.000099	tanh	Nadam	mse	1	120	256	False	rnn
315	250	0.000087	0.000188	tanh	rmsprop	mse	3	120	128	True	gru
700	300	0.000088	0.000100	sigmoid	Nadam	mse	3	60	64	False	lstm
200	400	0.000088	0.000119	tanh	Nadam	mse	2	30	64	False	rnn
317	400	0.000088	0.000113	sigmoid	Adam	mse	2	120	64	True	lstm
70	250	0.000088	0.000121	tanh	Nadam	mse	2	30	64	False	rnn
746	400	0.000088	0.000115	sigmoid	Adam	mse	3	30	64	False	rnn
545	300	0.000088	0.000105	elu	Nadam	mse	1	30	256	True	gru
194	300	0.000088	0.000095	elu	Nadam	mse	1	120	256	True	gru
272	300	0.000088	0.000103	tanh	Nadam	mse	3	30	256	False	gru
174	300	0.000088	0.000094	elu	Adam	mse	1	120	256	True	lstm
13	300	0.000088	0.000116	tanh	Nadam	mse	1	60	128	False	rnn
769	300	0.000089	0.000119	sigmoid	Adam	mse	3	120	64	True	gru
356	250	0.000089	0.000151	relu	rmsprop	mse	2	10	64	True	gru
459	400	0.000089	0.000111	elu	rmsprop	mse	3	120	256	False	rnn
92	300	0.000089	0.000119	tanh	rmsprop	mse	2	120	256	True	lstm
762	400	0.000089	0.000140	elu	rmsprop	mse	1	10	128	False	rnn
536	250	0.000089	0.000106	relu	Adam	mse	1	10	64	False	lstm
79	250	0.000089	0.000111	elu	Nadam	mse	2	10	64	False	rnn
621	400	0.000089	0.000122	sigmoid	Nadam	mse	1	60	64	True	rnn
347	400	0.000089	0.000133	tanh	rmsprop	mse	2	10	128	False	rnn
43	300	0.000089	0.000103	relu	Nadam	mse	1	30	256	False	lstm
367	250	0.000089	0.000107	elu	Adam	mse	2	30	128	True	rnn
323	250	0.000089	0.000107	relu	Adam	mse	1	10	128	True	gru
687	300	0.000090	0.000110	tanh	Nadam	mse	1	120	256	True	lstm
258	250	0.000090	0.000098	relu	Adam	mse	3	10	64	True	rnn
423	300	0.000090	0.000131	tanh	rmsprop	mse	3	60	256	True	lstm
240	400	0.000090	0.000117	sigmoid	Nadam	mse	1	60	64	True	lstm
166	400	0.000091	0.000139	elu	rmsprop	mse	2	60	256	True	lstm
401	400	0.000091	0.000108	elu	Nadam	mse	2	10	256	True	gru
352	400	0.000091	0.000160	relu	rmsprop	mse	3	10	128	True	rnn
615	400	0.000091	0.000120	sigmoid	Adam	mse	2	120	64	True	rnn
528	300	0.000091	0.000109	tanh	Nadam	mse	3	120	256	False	lstm
774	300	0.000092	0.000122	sigmoid	Nadam	mse	2	10	64	True	gru
378	250	0.000092	0.000150	relu	rmsprop	mse	3	10	64	False	rnn
641	300	0.000092	0.000094	relu	Adam	mse	1	60	256	True	rnn
189	250	0.000092	0.000117	elu	Nadam	mse	2	10	128	True	lstm
71	400	0.000092	0.000099	relu	Nadam	mse	2	10	128	False	lstm
31	250	0.000092	0.000118	elu	Nadam	mse	2	60	128	False	rnn
766	300	0.000092	0.000107	relu	Adam	mse	1	10	128	True	gru
473	300	0.000092	0.000111	elu	Nadam	mse	2	60	256	False	lstm
561	250	0.000092	0.000161	relu	rmsprop	mse	2	10	128	False	gru
293	250	0.000092	0.000149	relu	rmsprop	mse	3	60	128	True	lstm
738	250	0.000093	0.000091	elu	Adam	mse	2	60	256	True	lstm
305	250	0.000093	0.000115	tanh	Adam	mse	1	60	128	True	rnn
388	300	0.000093	0.000121	sigmoid	Adam	mse	2	30	64	False	gru

run	epochs	valLoss	loss	activation	optimizer	losses	layers	size	batchSize	bidirectional	layer
496	300	0.000093	0.000145	tanh	rmsprop	mse	1	30	128	True	gru
480	400	0.000093	0.000111	elu	Adam	mse	3	60	128	False	rnn
716	250	0.000094	0.000114	tanh	Nadam	mse	1	30	128	True	lstm
611	400	0.000094	0.000104	relu	Adam	mse	1	30	256	False	rnn
380	250	0.000094	0.000108	tanh	rmsprop	mse	2	120	256	True	lstm
364	400	0.000094	0.000109	tanh	Adam	mse	1	60	256	False	lstm
709	250	0.000094	0.000132	elu	Nadam	mse	3	30	64	True	rnn
547	250	0.000094	0.000106	relu	Nadam	mse	3	30	128	False	lstm
20	400	0.000095	0.000129	elu	rmsprop	mse	1	120	256	False	gru
470	300	0.000095	0.000125	tanh	Adam	mse	2	30	128	False	rnn
550	300	0.000095	0.000110	relu	Nadam	mse	2	10	256	False	gru
593	300	0.000095	0.000118	sigmoid	Adam	mse	3	10	64	True	gru
491	250	0.000095	0.000146	tanh	rmsprop	mse	1	10	128	True	lstm
736	400	0.000095	0.000167	sigmoid	rmsprop	mse	1	30	64	False	gru
639	300	0.000095	0.000109	tanh	Adam	mse	1	120	256	False	lstm
635	400	0.000095	0.000099	relu	Nadam	mse	3	60	256	True	lstm
602	300	0.000096	0.000122	sigmoid	Nadam	mse	1	30	64	False	lstm
697	300	0.000096	0.000128	tanh	rmsprop	mse	3	10	128	False	lstm
466	300	0.000096	0.000132	sigmoid	Nadam	mse	2	60	64	True	rnn
609	400	0.000096	0.000122	sigmoid	Nadam	mse	3	10	128	False	rnn
702	300	0.000096	0.000118	tanh	Nadam	mse	3	10	256	True	gru
582	250	0.000097	0.000132	tanh	rmsprop	mse	3	10	128	False	lstm
203	250	0.000097	0.000124	sigmoid	Adam	mse	2	30	64	False	rnn
220	400	0.000097	0.000097	relu	Adam	mse	3	30	256	False	rnn
11	400	0.000097	0.000111	elu	Nadam	mse	2	60	256	False	rnn
351	400	0.000097	0.000124	tanh	Nadam	mse	3	10	128	False	rnn
119	400	0.000097	0.000115	elu	Nadam	mse	1	30	256	True	lstm
664	250	0.000097	0.000199	relu	rmsprop	mse	3	30	128	True	gru
208	250	0.000097	0.000129	tanh	Adam	mse	1	60	128	False	rnn
215	250	0.000097	0.000120	elu	Nadam	mse	1	30	256	False	lstm
670	300	0.000098	0.000120	sigmoid	Adam	mse	2	60	64	False	gru
744	250	0.000098	0.000125	sigmoid	Adam	mse	3	30	64	False	rnn
603	300	0.000098	0.000131	elu	rmsprop	mse	1	120	256	True	rnn
301	300	0.000098	0.000132	sigmoid	Nadam	mse	1	60	64	False	lstm
688	250	0.000098	0.000124	sigmoid	Adam	mse	3	120	64	True	lstm
291	400	0.000098	0.000129	sigmoid	Nadam	mse	2	10	128	False	rnn
69	400	0.000098	0.000151	elu	rmsprop	mse	3	120	256	False	lstm
188	400	0.000098	0.000093	relu	Adam	mse	3	30	256	False	lstm
232	400	0.000098	0.000121	elu	Adam	mse	1	10	128	False	rnn
542	250	0.000099	0.000134	sigmoid	Nadam	mse	3	10	64	True	rnn
425	300	0.000099	0.000122	sigmoid	Adam	mse	2	120	64	False	gru
237	250	0.000099	0.000112	relu	Nadam	mse	1	30	128	True	rnn
772	300	0.000099	0.000126	sigmoid	Nadam	mse	2	30	64	False	lstm
249	250	0.000099	0.000132	sigmoid	Nadam	mse	3	30	64	True	lstm
157	300	0.000099	0.000104	relu	Adam	mse	3	30	256	False	lstm
222	300	0.000099	0.000123	elu	Nadam	mse	1	10	128	False	rnn
452	250	0.000100	0.000123	sigmoid	Nadam	mse	2	10	64	False	lstm
652	400	0.000100	0.000115	tanh	Adam	mse	1	30	256	False	lstm
556	250	0.000101	0.000119	tanh	Adam	mse	3	60	256	True	rnn
732	250	0.000101	0.000110	tanh	Nadam	mse	2	120	256	False	lstm
554	250	0.000101	0.000127	sigmoid	Adam	mse	3	60	64	False	lstm
324	300	0.000101	0.000120	tanh	Adam	mse	3	30	256	True	rnn
665	300	0.000101	0.000130	sigmoid	Adam	mse	2	120	64	False	lstm
534	250	0.000101	0.000199	tanh	rmsprop	mse	2	30	128	True	gru
225	400	0.000101	0.000127	sigmoid	Adam	mse	1	60	128	False	gru
566	400	0.000101	0.000131	tanh	rmsprop	mse	1	10	256	False	rnn
38	250	0.000101	0.000112	tanh	Nadam	mse	3	10	128	True	lstm
195	250	0.000101	0.000130	sigmoid	Adam	mse	3	120	64	False	rnn
569	300	0.000102	0.000132	tanh	rmsprop	mse	2	10	256	False	gru
647	300	0.000102	0.000134	tanh	rmsprop	mse	2	30	256	False	lstm
130	400	0.000102	0.000122	sigmoid	Nadam	mse	3	30	128	False	lstm
427	300	0.000103	0.000154	elu	rmsprop	mse	1	10	128	False	lstm
646	250	0.000103	0.000208	relu	rmsprop	mse	2	30	128	True	rnn
434	400	0.000103	0.000183	sigmoid	rmsprop	mse	2	30	64	False	gru
469	300	0.000103	0.000114	elu	Nadam	mse	2	120	256	False	lstm
344	250	0.000103	0.000105	elu	Adam	mse	3	30	128	False	lstm
138	300	0.000103	0.000127	sigmoid	Adam	mse	1	30	64	False	lstm
451	250	0.000103	0.000120	tanh	Adam	mse	2	60	128	False	rnn
439	300	0.000104	0.000106	relu	Nadam	mse	1	30	256	True	lstm
354	400	0.000104	0.000234	sigmoid	rmsprop	mse	1	30	64	True	gru
33	400	0.000104	0.000128	sigmoid	Adam	mse	3	120	128	False	rnn
689	400	0.000104	0.000140	elu	rmsprop	mse	1	60	256	True	rnn
241	250	0.000104	0.000130	relu	Nadam	mse	1	10	256	False	gru
760	250	0.000104	0.000124	tanh	Adam	mse	1	10	64	True	rnn
165	250	0.000104	0.000119	elu	Adam	mse	2	10	128	False	lstm
502	400	0.000104	0.000123	tanh	Adam	mse	3	60	256	False	rnn
775	400	0.000105	0.000139	tanh	rmsprop	mse	3	10	256	False	gru
219	400	0.000105	0.000126	relu	rmsprop	mse	2	120	256	False	gru
239	400	0.000105	0.000128	sigmoid	Adam	mse	3	60	128	False	rnn
152	400	0.000105	0.000134	elu	rmsprop	mse	1	10	256	False	gru
51	250	0.000105	0.000122	tanh	Adam	mse	3	120	128	False	rnn
183	250	0.000105	0.000149	sigmoid	Nadam	mse	1	60	64	False	rnn
595	400	0.000105	0.000138	elu	rmsprop	mse	1	30	256	True	gru
54	300	0.000105	0.000122	elu	Adam	mse	3	30	128	True	rnn
674	400	0.000105	0.000148	elu	rmsprop	mse	2	120	256	True	gru
649	400	0.000105	0.000130	sigmoid	Adam	mse	3	30	128	True	rnn
455	400	0.000105	0.000140	sigmoid	Adam	mse	3	120	128	True	rnn
620	400	0.000105	0.000163	sigmoid	rmsprop	mse	2	30	64	True	rnn
142	400	0.000105	0.000135	sigmoid	Adam	mse	2	120	128	False	rnn
587	250	0.000106	0.000130	tanh	Adam	mse	3	30	256	False	rnn
357	300	0.000106	0.000147	tanh	rmsprop	mse	1	10	128	False	lstm
411	300	0.000106	0.000156	relu	rmsprop	mse	2	60	128	False	rnn
303	300	0.000106	0.000135	tanh	Adam	mse	1	120	256	True	rnn
328	250	0.000106	0.000166	relu	rmsprop	mse	2	30	128	False	lstm
216	250	0.000107	0.000127	sigmoid	Adam	mse	1	10	64	True	lstm
570	300	0.000107	0.000140	tanh	rmsprop	mse	2	30	256	True	lstm
133	300	0.000107	0.000145	sigmoid	Nadam	mse	1	120	64	True	gru
90	250	0.000107	0.000176	sigmoid	rmsprop	mse	2	60	64	True	gru

run	epochs	valLoss	loss	activation	optimizer	losses	layers	size	batchSize	bidirectional	layer
679	400	0.000108	0.000136	tanh	Nadam	mse	1	30	256	False	lstm
268	300	0.000108	0.000124	elu	Nadam	mse	3	60	128	False	rnn
574	250	0.000109	0.000161	relu	rmsprop	mse	2	60	128	False	lstm
61	250	0.000109	0.000216	elu	rmsprop	mse	3	30	128	False	rnn
777	300	0.000109	0.000131	sigmoid	Adam	mse	2	10	64	False	lstm
256	400	0.000109	0.000162	sigmoid	rmsprop	mse	3	10	64	True	rnn
392	300	0.000109	0.000117	relu	Nadam	mse	2	120	256	False	lstm
168	250	0.000110	0.000139	sigmoid	Nadam	mse	2	30	64	True	lstm
396	300	0.000110	0.000130	sigmoid	Nadam	mse	3	120	128	False	gru
672	400	0.000110	0.000140	sigmoid	Adam	mse	2	30	128	True	rnn
180	400	0.000110	0.000161	sigmoid	rmsprop	mse	3	10	64	True	lstm
695	300	0.000110	0.000124	tanh	Adam	mse	1	30	256	False	gru
633	300	0.000111	0.000157	sigmoid	rmsprop	mse	1	10	64	False	rnn
10	300	0.000111	0.000124	elu	Adam	mse	3	60	128	False	rnn
493	250	0.000111	0.000130	relu	rmsprop	mse	1	10	256	False	lstm
711	250	0.000111	0.000136	tanh	Adam	mse	2	30	128	False	rnn
140	250	0.000111	0.000162	sigmoid	rmsprop	mse	2	10	64	False	lstm
134	300	0.000111	0.000136	sigmoid	Adam	mse	3	30	128	False	rnn
492	400	0.000111	0.000137	sigmoid	Adam	mse	1	60	256	False	gru
449	400	0.000111	0.000135	sigmoid	Nadam	mse	3	10	128	True	gru
274	250	0.000111	0.000118	relu	Adam	mse	2	30	256	False	rnn
284	400	0.000111	0.000130	relu	rmsprop	mse	1	10	256	True	gru
348	400	0.000111	0.000142	elu	rmsprop	mse	1	10	256	False	lstm
190	400	0.000111	0.000174	tanh	rmsprop	mse	3	10	128	False	rnn
483	250	0.000111	0.000123	relu	Adam	mse	2	10	128	False	rnn
30	300	0.000111	0.000139	sigmoid	Nadam	mse	1	30	128	False	lstm
8	400	0.000112	0.000127	sigmoid	Adam	mse	1	10	128	True	gru
162	300	0.000112	0.000166	relu	rmsprop	mse	1	60	256	True	gru
640	400	0.000113	0.000137	sigmoid	Adam	mse	2	10	128	False	lstm
734	250	0.000113	0.000236	elu	rmsprop	mse	2	30	128	False	rnn
624	250	0.000113	0.000235	sigmoid	rmsprop	mse	1	120	64	True	rnn
227	250	0.000113	0.000178	sigmoid	rmsprop	mse	2	10	64	True	gru
109	250	0.000113	0.000140	tanh	Nadam	mse	2	10	128	True	rnn
468	300	0.000113	0.000186	sigmoid	rmsprop	mse	3	60	64	True	lstm
465	400	0.000113	0.000149	tanh	Nadam	mse	3	60	128	True	rnn
84	250	0.000114	0.000136	tanh	Adam	mse	2	60	256	False	rnn
565	250	0.000114	0.000144	sigmoid	Nadam	mse	3	10	128	False	rnn
359	300	0.000114	0.000159	sigmoid	rmsprop	mse	1	10	64	True	gru
248	250	0.000114	0.000207	sigmoid	rmsprop	mse	3	30	64	True	gru
644	300	0.000114	0.000182	sigmoid	rmsprop	mse	1	30	128	False	lstm
627	400	0.000114	0.000158	sigmoid	Adam	mse	1	120	128	True	rnn
522	300	0.000114	0.000142	sigmoid	Adam	mse	2	60	128	False	gru
406	300	0.000115	0.000159	tanh	rmsprop	mse	1	60	256	True	gru
739	400	0.000115	0.000135	sigmoid	Adam	mse	3	30	256	True	lstm
643	250	0.000115	0.000141	sigmoid	Nadam	mse	2	10	128	False	lstm
270	300	0.000115	0.000154	sigmoid	rmsprop	mse	2	10	128	False	gru
619	300	0.000116	0.000157	sigmoid	rmsprop	mse	1	10	128	True	lstm
442	400	0.000116	0.000245	sigmoid	rmsprop	mse	1	60	64	True	gru
126	300	0.000117	0.000141	sigmoid	Nadam	mse	3	60	128	False	lstm
474	250	0.000117	0.000132	tanh	rmsprop	mse	3	60	256	True	lstm
730	400	0.000117	0.000147	sigmoid	Adam	mse	1	120	256	False	lstm
461	300	0.000117	0.000172	tanh	rmsprop	mse	3	10	256	True	gru
498	250	0.000117	0.000136	elu	Adam	mse	3	10	256	True	gru
540	300	0.000117	0.000141	tanh	Adam	mse	2	30	256	False	rnn
418	300	0.000118	0.000149	sigmoid	Nadam	mse	1	120	128	False	lstm
87	400	0.000118	0.000140	elu	Adam	mse	1	10	128	True	rnn
659	400	0.000118	0.000158	sigmoid	rmsprop	mse	3	10	128	False	gru
76	400	0.000118	0.000139	sigmoid	Adam	mse	3	60	256	False	lstm
207	300	0.000118	0.000153	elu	rmsprop	mse	1	120	256	False	gru
460	400	0.000118	0.000148	sigmoid	Adam	mse	1	120	256	True	lstm
463	250	0.000119	0.000147	tanh	Adam	mse	1	120	256	False	rnn
604	300	0.000119	0.000144	relu	rmsprop	mse	1	30	256	False	lstm
298	250	0.000119	0.000139	elu	rmsprop	mse	2	120	256	True	lstm
412	300	0.000120	0.000163	tanh	rmsprop	mse	1	30	256	True	gru
667	300	0.000120	0.000170	relu	rmsprop	mse	3	60	128	False	rnn
6	250	0.000120	0.000135	tanh	Adam	mse	1	10	128	False	gru
541	300	0.000120	0.000182	sigmoid	rmsprop	mse	1	10	64	True	rnn
283	250	0.000121	0.000139	sigmoid	Adam	mse	3	10	64	True	gru
494	250	0.000121	0.000151	tanh	Nadam	mse	2	10	256	True	rnn
606	300	0.000121	0.000204	sigmoid	rmsprop	mse	3	30	64	True	rnn
398	400	0.000121	0.000158	tanh	Nadam	mse	3	30	256	True	rnn
600	300	0.000121	0.000152	elu	rmsprop	mse	2	60	256	True	lstm
131	300	0.000122	0.000157	sigmoid	Adam	mse	2	30	128	True	rnn
271	250	0.000122	0.000163	sigmoid	rmsprop	mse	1	10	256	False	lstm
495	400	0.000122	0.000201	sigmoid	rmsprop	mse	3	60	64	True	rnn
717	400	0.000122	0.000178	tanh	rmsprop	mse	1	10	256	True	rnn
479	300	0.000123	0.000129	relu	Adam	mse	3	10	128	True	rnn
132	300	0.000123	0.000152	sigmoid	Adam	mse	3	120	256	False	rnn
488	250	0.000123	0.000141	relu	rmsprop	mse	1	120	256	True	gru
420	250	0.000123	0.000200	sigmoid	rmsprop	mse	1	30	64	True	lstm
231	400	0.000124	0.000168	sigmoid	Adam	mse	1	120	256	False	rnn
287	250	0.000124	0.000137	tanh	Nadam	mse	3	30	256	False	lstm
46	250	0.000124	0.000152	sigmoid	Adam	mse	3	60	128	False	rnn
338	250	0.000125	0.000146	elu	Adam	mse	1	10	128	False	gru
745	400	0.000125	0.000211	sigmoid	rmsprop	mse	1	30	128	False	rnn
482	300	0.000125	0.000159	sigmoid	Adam	mse	2	120	128	False	rnn
727	300	0.000125	0.000158	sigmoid	Nadam	mse	3	120	128	True	rnn
78	300	0.000125	0.000159	sigmoid	Nadam	mse	3	60	128	True	lstm
713	400	0.000126	0.000164	sigmoid	Adam	mse	1	30	256	False	lstm
361	250	0.000126	0.000142	sigmoid	Adam	mse	3	120	256	True	gru
586	300	0.000126	0.000175	sigmoid	Nadam	mse	1	120	256	False	lstm
40	300	0.000126	0.000151	sigmoid	Adam	mse	3	10	128	False	lstm
464	400	0.000126	0.000168	sigmoid	Adam	mse	1	60	256	False	rnn
369	400	0.000126	0.000160	sigmoid	Adam	mse	2	10	128	False	rnn
757	300	0.000127	0.000143	sigmoid	Nadam	mse	2	120	128	True	gru
289	250	0.000127	0.000166	tanh	Nadam	mse	1	10	256	False	lstm
481	400	0.000127	0.000245	sigmoid	rmsprop	mse	3	120	128	True	lstm
370	250	0.000128	0.000209	relu	rmsprop	mse	2	60	256	False	gru

run	epochs	valLoss	loss	activation	optimizer	losses	layers	size	batchSize	bidirectional	layer
39	400	0.000128	0.000164	sigmoid	Nadam	mse	1	30	256	True	lstm
390	250	0.000128	0.000140	elu	rmsprop	mse	3	120	256	False	rnn
768	250	0.000128	0.000193	tanh	rmsprop	mse	1	120	256	False	lstm
172	250	0.000129	0.000197	tanh	rmsprop	mse	1	60	256	True	gru
9	250	0.000129	0.000174	sigmoid	Nadam	mse	2	30	128	False	rnn
158	250	0.000129	0.000159	sigmoid	Adam	mse	3	30	128	True	rnn
653	250	0.000129	0.000116	relu	Nadam	mse	1	30	128	True	lstm
307	400	0.000129	0.000178	sigmoid	Nadam	mse	2	10	256	False	rnn
538	250	0.000129	0.000155	sigmoid	Nadam	mse	3	30	128	True	lstm
111	250	0.000129	0.000219	sigmoid	rmsprop	mse	3	60	64	False	rnn
756	250	0.000129	0.000143	elu	Nadam	mse	3	60	128	False	rnn
137	250	0.000130	0.000210	tanh	rmsprop	mse	1	10	64	True	rnn
22	400	0.000130	0.000147	elu	Nadam	mse	1	10	128	False	gru
597	300	0.000130	0.000152	elu	Adam	mse	2	10	256	False	gru
573	400	0.000130	0.000120	relu	rmsprop	mse	3	60	256	False	lstm
535	250	0.000131	0.000157	sigmoid	Adam	mse	2	120	256	False	gru
21	400	0.000131	0.000157	sigmoid	Nadam	mse	1	10	256	True	gru
601	250	0.000131	0.000193	sigmoid	rmsprop	mse	1	10	128	True	rnn
59	250	0.000131	0.000172	sigmoid	rmsprop	mse	1	10	256	False	gru
577	250	0.000131	0.000298	sigmoid	rmsprop	mse	3	120	64	True	rnn
409	250	0.000132	0.000152	tanh	Nadam	mse	3	10	256	True	gru
511	300	0.000132	0.000163	sigmoid	Adam	mse	3	30	256	False	gru
161	400	0.000132	0.000208	sigmoid	rmsprop	mse	2	10	128	True	rnn
764	300	0.000132	0.000251	sigmoid	rmsprop	mse	2	30	128	False	rnn
395	300	0.000133	0.000143	elu	Nadam	mse	3	120	256	False	lstm
211	300	0.000133	0.000170	sigmoid	Nadam	mse	1	10	128	True	rnn
363	300	0.000133	0.000226	tanh	rmsprop	mse	2	10	128	True	rnn
677	250	0.000133	0.000182	elu	rmsprop	mse	2	30	256	False	lstm
329	300	0.000134	0.000164	relu	rmsprop	mse	3	120	128	True	rnn
721	300	0.000134	0.000251	sigmoid	rmsprop	mse	2	60	128	False	gru
101	300	0.000134	0.000161	sigmoid	Nadam	mse	3	30	256	False	gru
67	250	0.000134	0.000163	sigmoid	Nadam	mse	3	120	128	True	gru
628	300	0.000134	0.000194	relu	rmsprop	mse	1	30	256	True	lstm
436	300	0.000135	0.000143	relu	Nadam	mse	2	10	256	False	rnn
18	250	0.000135	0.000227	elu	rmsprop	mse	3	60	256	False	rnn
634	400	0.000136	0.000193	elu	rmsprop	mse	3	10	256	False	rnn
318	250	0.000136	0.000172	sigmoid	Nadam	mse	3	30	256	False	gru
230	300	0.000136	0.000157	elu	Nadam	mse	3	30	128	False	rnn
617	300	0.000137	0.000157	sigmoid	Adam	mse	3	60	256	True	lstm
242	250	0.000137	0.000178	sigmoid	Adam	mse	1	120	256	True	lstm
698	300	0.000137	0.000168	sigmoid	Nadam	mse	3	60	256	False	gru
156	300	0.000137	0.000209	sigmoid	rmsprop	mse	1	10	128	True	rnn
149	250	0.000138	0.000175	sigmoid	Nadam	mse	1	30	256	True	lstm
209	250	0.000138	0.000198	relu	rmsprop	mse	2	120	128	False	rnn
81	250	0.000138	0.000162	sigmoid	Adam	mse	3	120	256	True	lstm
770	300	0.000139	0.000161	elu	Adam	mse	3	60	256	True	rnn
85	250	0.000139	0.000240	sigmoid	rmsprop	mse	3	30	64	True	rnn
722	250	0.000140	0.000191	sigmoid	Adam	mse	2	120	256	True	rnn
497	300	0.000141	0.000220	elu	rmsprop	mse	3	60	256	True	gru
431	300	0.000143	0.000205	relu	rmsprop	mse	2	120	128	True	rnn
625	400	0.000143	0.000255	tanh	rmsprop	mse	1	30	128	True	rnn
186	300	0.000143	0.000208	relu	rmsprop	mse	3	10	256	True	lstm
560	250	0.000144	0.000196	sigmoid	Nadam	mse	1	60	256	False	rnn
238	300	0.000145	0.000180	sigmoid	Adam	mse	1	10	256	True	lstm
167	250	0.000145	0.000229	sigmoid	rmsprop	mse	3	10	128	True	rnn
443	400	0.000145	0.000187	sigmoid	Nadam	mse	1	60	256	True	lstm
656	400	0.000145	0.000226	sigmoid	rmsprop	mse	3	30	128	True	lstm
685	300	0.000146	0.000219	elu	rmsprop	mse	1	60	256	True	rnn
360	400	0.000146	0.000161	relu	rmsprop	mse	3	120	256	False	lstm
331	300	0.000146	0.000178	sigmoid	Adam	mse	2	10	128	False	gru
66	250	0.000147	0.000264	sigmoid	rmsprop	mse	2	120	128	True	lstm
26	300	0.000148	0.000169	elu	Adam	mse	1	10	256	False	gru
148	250	0.000148	0.000176	tanh	Nadam	mse	3	30	128	True	rnn
62	250	0.000149	0.000197	sigmoid	Adam	mse	3	30	256	False	lstm
584	300	0.000153	0.000201	elu	Nadam	mse	3	60	128	True	rnn
557	400	0.000153	0.000232	sigmoid	rmsprop	mse	1	30	256	False	rnn
105	400	0.000154	0.000169	elu	Nadam	mse	2	10	256	False	rnn
312	250	0.000156	0.000216	relu	rmsprop	mse	2	120	256	False	gru
25	250	0.000157	0.000205	sigmoid	Nadam	mse	1	60	256	True	gru
433	250	0.000157	0.000201	sigmoid	Nadam	mse	2	10	256	True	lstm
345	400	0.000158	0.000180	sigmoid	Adam	mse	1	30	256	False	gru
245	300	0.000159	0.000184	sigmoid	Nadam	mse	1	10	256	False	gru
120	250	0.000160	0.000269	sigmoid	rmsprop	mse	3	60	128	False	lstm
316	300	0.000161	0.000297	sigmoid	rmsprop	mse	3	60	128	True	lstm
366	300	0.000162	0.000186	sigmoid	Nadam	mse	2	10	256	False	gru
72	300	0.000163	0.000181	tanh	Adam	mse	1	30	256	True	rnn
387	250	0.000163	0.000205	sigmoid	Adam	mse	3	10	128	True	rnn
410	250	0.000164	0.000211	relu	rmsprop	mse	3	120	128	True	rnn
118	300	0.000164	0.000189	relu	rmsprop	mse	2	30	256	False	lstm
450	250	0.000164	0.000203	sigmoid	Nadam	mse	3	10	256	True	lstm
42	250	0.000164	0.000235	sigmoid	Adam	mse	2	10	128	False	lstm
310	300	0.000165	0.000221	sigmoid	Nadam	mse	1	120	256	True	lstm
107	400	0.000165	0.000196	sigmoid	Nadam	mse	3	10	256	False	gru
113	300	0.000168	0.000187	elu	rmsprop	mse	3	10	256	True	gru
320	400	0.000169	0.000207	relu	rmsprop	mse	3	10	256	False	rnn
776	400	0.000170	0.000307	sigmoid	rmsprop	mse	1	30	128	True	gru
581	400	0.000170	0.000230	tanh	rmsprop	mse	2	30	256	False	rnn
532	300	0.000171	0.000223	tanh	Nadam	mse	1	10	256	False	rnn
447	250	0.000171	0.000226	sigmoid	Nadam	mse	1	120	256	True	lstm
743	250	0.000173	0.000258	tanh	rmsprop	mse	3	10	256	True	rnn
657	300	0.000174	0.000201	elu	Adam	mse	1	10	64	True	rnn
503	300	0.000175	0.000292	sigmoid	rmsprop	mse	2	30	128	True	gru
417	250	0.000175	0.000260	sigmoid	rmsprop	mse	2	30	128	False	gru
544	250	0.000175	0.000256	relu	rmsprop	mse	2	60	256	False	lstm
311	400	0.000175	0.000157	relu	Adam	mse	2	10	256	True	rnn
64	250	0.000177	0.000212	sigmoid	Nadam	mse	3	120	128	True	rnn
83	300	0.000178	0.000261	sigmoid	Nadam	mse	2	10	256	False	rnn
273	250	0.000181	0.000366	sigmoid	rmsprop	mse	1	60	128	True	lstm

run	epochs	valLoss	loss	activation	optimizer	losses	layers	size	batchSize	bidirectional	layer
295	250	0.000181	0.000216	elu	Nadam	mse	3	120	64	False	rnn
228	400	0.000182	0.000196	relu	rmsprop	mse	3	60	256	True	gru
110	250	0.000183	0.000197	relu	Nadam	mse	1	10	256	True	rnn
701	250	0.000183	0.000261	sigmoid	Adam	mse	2	30	256	False	lstm
515	250	0.000184	0.000247	sigmoid	rmsprop	mse	3	10	256	True	gru
723	400	0.000185	0.000212	relu	rmsprop	mse	3	120	256	False	rnn
519	300	0.000185	0.000237	sigmoid	Nadam	mse	3	60	256	True	rnn
636	300	0.000186	0.000258	sigmoid	rmsprop	mse	2	10	256	True	rnn
715	400	0.000187	0.000192	elu	Nadam	mse	3	120	256	False	rnn
68	300	0.000187	0.000269	sigmoid	rmsprop	mse	3	30	256	False	gru
304	300	0.000189	0.000339	sigmoid	rmsprop	mse	1	60	128	False	rnn
568	300	0.000192	0.000299	elu	rmsprop	mse	2	10	256	True	rnn
675	250	0.000194	0.001892	tanh	rmsprop	mse	2	120	256	True	rnn
265	400	0.000196	0.000350	sigmoid	rmsprop	mse	3	30	256	True	rnn
690	300	0.000200	0.000273	relu	rmsprop	mse	2	10	256	True	rnn
413	250	0.000200	0.000281	relu	rmsprop	mse	2	60	256	True	rnn
402	250	0.000201	0.000365	sigmoid	rmsprop	mse	2	30	256	False	rnn
333	300	0.000206	0.000223	relu	Nadam	mse	2	10	256	True	rnn
748	400	0.000207	0.000256	sigmoid	rmsprop	mse	3	60	256	False	lstm
598	250	0.000210	0.000339	sigmoid	rmsprop	mse	1	30	256	True	lstm
3	250	0.000211	0.000240	elu	Adam	mse	3	30	256	False	rnn
525	300	0.000211	0.000238	tanh	rmsprop	mse	2	30	256	True	gru
605	300	0.000214	0.000383	sigmoid	rmsprop	mse	3	30	128	True	rnn
742	250	0.000215	0.000294	elu	rmsprop	mse	2	10	256	True	rnn
325	300	0.000215	0.000380	sigmoid	rmsprop	mse	2	60	256	False	rnn
127	250	0.000216	0.000348	sigmoid	rmsprop	mse	3	60	128	True	lstm
446	300	0.000216	0.000248	elu	rmsprop	mse	3	10	256	False	rnn
707	250	0.000219	0.000329	tanh	rmsprop	mse	1	30	256	False	rnn
537	250	0.000220	0.000288	relu	rmsprop	mse	3	120	256	False	lstm
45	250	0.000226	0.000317	relu	rmsprop	mse	3	30	256	False	rnn
322	300	0.000234	0.000299	sigmoid	Adam	mse	1	10	256	True	rnn
243	250	0.000241	0.000258	tanh	Nadam	mse	1	30	256	True	rnn
306	250	0.000242	0.000300	tanh	Nadam	mse	1	10	256	False	rnn
169	400	0.000246	0.000348	sigmoid	rmsprop	mse	2	120	256	False	rnn
192	300	0.000249	0.000383	sigmoid	rmsprop	mse	1	120	256	False	lstm
666	300	0.000250	0.000249	elu	Nadam	mse	3	120	256	True	rnn
181	400	0.000250	0.000476	sigmoid	rmsprop	mse	3	120	256	True	gru
585	250	0.000250	0.000267	elu	Nadam	mse	3	60	256	False	rnn
487	250	0.000266	0.000377	sigmoid	Nadam	mse	2	10	256	False	lstm
648	250	0.000269	0.000309	elu	Nadam	mse	3	10	256	False	rnn
752	250	0.000281	0.000475	sigmoid	rmsprop	mse	1	120	128	False	gru
686	300	0.000283	0.006180	tanh	rmsprop	mse	2	120	64	True	rnn
539	250	0.000300	0.000358	relu	rmsprop	mse	2	120	256	True	rnn
389	250	0.000320	0.001396	tanh	rmsprop	mse	2	120	64	False	rnn
269	250	0.000320	0.000381	relu	rmsprop	mse	2	120	256	False	rnn
108	300	0.000322	0.000694	tanh	rmsprop	mse	2	30	64	True	rnn
187	400	0.000327	0.000614	tanh	rmsprop	mse	2	60	128	False	rnn
75	250	0.000334	0.000473	sigmoid	rmsprop	mse	3	120	256	False	lstm
718	300	0.000342	0.001883	tanh	rmsprop	mse	2	120	64	False	rnn
529	250	0.000342	0.000354	relu	Adam	mse	1	10	256	True	rnn
221	250	0.000343	0.000487	sigmoid	Adam	mse	2	10	256	True	rnn
714	250	0.000360	0.000361	tanh	Nadam	mse	2	60	256	True	rnn
254	300	0.000405	0.000440	elu	Adam	mse	1	10	128	True	rnn
171	250	0.000430	0.000456	elu	Nadam	mse	2	10	256	True	rnn
576	300	0.000479	0.000751	tanh	rmsprop	mse	3	60	256	False	rnn
490	400	0.000572	0.000707	sigmoid	rmsprop	mse	1	120	256	False	gru
391	300	0.001111	0.001780	tanh	rmsprop	mse	2	60	256	True	rnn
512	250	0.001713	0.002920	tanh	rmsprop	mse	3	120	256	False	rnn
277	300	0.002318	0.003301	tanh	rmsprop	mse	3	60	256	True	rnn

Appendix B

Talos Run 2

The settings used by Talos:

- **Activation functions:** ReLU, ELU
- **Optimizers:** Nadam, Adam, rmsprop
- **Loss:** MSE
- **Layers:** 1, 2, 3
- **Size per layer:** 30, 60, 120
- **Batch size:** 128
- **Bidirectional:** True, False
- **Layer type:** GRU, LSTM
- **Epochs:** 400
- **(Recurrent) Dropout:** 0.0, 0.2
- **Extra dense layer size:** 0, 30, 90

B.1 Sorted by validation loss

run	epochs	valLoss	loss	activation	optimizer	losses	layers	size	bidirectional	dropout	dense	layer	testLoss
123	380	0.000559	0.000270	elu	Adam	mse	1	60	True	0	90	gru	0.377228
124	329	0.000560	0.000356	elu	Nadam	mse	2	120	False	0.2	0	lstm	0.324017
11	197	0.000563	0.000310	elu	Nadam	mse	3	120	False	0.2	90	gru	0.340967
81	263	0.000563	0.000248	elu	Adam	mse	3	120	True	0.2	0	gru	0.325456
84	317	0.000571	0.000241	elu	Nadam	mse	3	120	False	0.2	90	lstm	0.355334
102	309	0.000571	0.000321	elu	Adam	mse	2	120	True	0.2	0	gru	0.342152
65	400	0.000579	0.000250	elu	Adam	mse	2	30	True	0	90	gru	0.354498
89	400	0.000580	0.000426	elu	rmsprop	mse	2	60	True	0.2	90	gru	0.349329
103	400	0.000584	0.000416	elu	Nadam	mse	1	120	False	0.2	0	gru	0.310417
112	400	0.000587	0.000477	elu	Adam	mse	2	60	False	0.2	90	gru	0.262152
93	355	0.000593	0.000305	elu	Nadam	mse	3	60	False	0.2	30	gru	0.323147
26	169	0.000594	0.000226	elu	Adam	mse	3	120	True	0	90	gru	0.326938
27	155	0.000595	0.000282	relu	Nadam	mse	1	120	True	0	0	gru	0.315502
128	400	0.000601	0.000503	elu	rmsprop	mse	1	120	True	0.2	0	gru	0.303748
23	264	0.000609	0.000320	elu	rmsprop	mse	3	30	True	0	0	gru	0.335117
117	400	0.000618	0.000283	relu	Nadam	mse	2	120	False	0.2	0	lstm	0.321786
83	400	0.000622	0.000497	elu	rmsprop	mse	1	120	False	0.2	0	lstm	0.271461
12	312	0.000623	0.000328	relu	Adam	mse	3	120	False	0.2	0	lstm	0.266114
122	293	0.000630	0.000298	elu	Adam	mse	2	60	False	0	0	gru	0.341811
61	262	0.000631	0.000485	elu	Adam	mse	2	120	True	0.2	30	lstm	0.286525
63	215	0.000635	0.000176	elu	Nadam	mse	1	120	False	0	0	gru	0.353077
72	258	0.000638	0.000400	elu	Nadam	mse	3	120	True	0	90	gru	0.320142
46	339	0.000638	0.000418	relu	Adam	mse	2	120	False	0.2	0	gru	0.263046
17	196	0.000638	0.000348	elu	Adam	mse	1	120	True	0	0	lstm	0.249039
127	235	0.000641	0.000466	elu	rmsprop	mse	3	60	True	0.2	30	gru	0.303977
54	400	0.000642	0.000359	elu	Adam	mse	1	60	False	0	0	gru	0.314058
98	399	0.000644	0.000491	elu	Adam	mse	3	30	False	0.2	30	lstm	0.105100
114	286	0.000646	0.000384	relu	Adam	mse	2	60	False	0	0	gru	0.305044
125	318	0.000648	0.000433	elu	Adam	mse	2	60	True	0.2	0	gru	0.329163
104	172	0.000649	0.000267	elu	Nadam	mse	3	60	False	0	90	gru	0.337518
75	216	0.000652	0.000186	elu	Nadam	mse	2	60	True	0	0	lstm	0.313548
60	166	0.000654	0.000315	relu	Nadam	mse	2	60	False	0	90	gru	0.318951
86	400	0.000657	0.000385	elu	Nadam	mse	3	30	False	0.2	0	lstm	0.246972
94	400	0.000657	0.000548	elu	Adam	mse	2	30	True	0.2	30	lstm	0.135762
19	400	0.000662	0.000483	relu	Nadam	mse	1	60	True	0.2	0	lstm	0.263520
0	400	0.000671	0.000596	elu	Nadam	mse	1	60	False	0.2	30	gru	0.297633
15	265	0.000673	0.000269	elu	Adam	mse	2	60	True	0	0	lstm	0.254788
79	160	0.000676	0.000228	elu	Nadam	mse	2	120	False	0	90	lstm	0.311716

run	epochs	valLoss	loss	activation	optimizer	losses	layers	size	bidirectional	dropout	dense	layer	testLoss
36	237	0.000676	0.000173	relu	Nadam	mse	2	120	False	0	30	lstm	0.294147
51	359	0.000677	0.000382	elu	rmsprop	mse	1	60	True	0	0	gru	0.316340
25	383	0.000678	0.000351	relu	Adam	mse	3	120	False	0.2	0	gru	0.327694
111	364	0.000679	0.000414	elu	Adam	mse	1	30	True	0	90	lstm	0.286464
100	316	0.000679	0.000396	relu	Nadam	mse	3	60	False	0.2	0	lstm	0.147575
119	260	0.000680	0.000433	relu	Nadam	mse	2	60	True	0.2	0	gru	0.254822
107	400	0.000683	0.000419	elu	Adam	mse	2	30	False	0	0	gru	0.315303
38	329	0.000684	0.000241	relu	Adam	mse	3	60	False	0	30	lstm	0.211797
68	400	0.000685	0.000543	elu	rmsprop	mse	1	120	False	0.2	90	lstm	0.255893
22	324	0.000686	0.000306	relu	Adam	mse	2	120	True	0.2	30	gru	0.309502
28	189	0.000690	0.000317	elu	rmsprop	mse	2	120	False	0	90	lstm	0.329985
32	272	0.000694	0.000178	relu	Nadam	mse	3	120	False	0	0	lstm	0.303562
6	173	0.000694	0.000306	elu	Adam	mse	3	60	True	0	0	lstm	0.159340
67	347	0.000695	0.000445	relu	Nadam	mse	1	30	True	0	0	gru	0.338427
45	177	0.000699	0.000327	elu	rmsprop	mse	3	120	False	0	0	gru	0.316596
99	400	0.000700	0.000577	elu	Adam	mse	1	60	True	0.2	90	lstm	0.206794
49	400	0.000702	0.000475	elu	rmsprop	mse	1	30	True	0	0	lstm	0.310804
96	237	0.000703	0.000299	relu	Nadam	mse	1	60	True	0	30	gru	0.338350
40	309	0.000707	0.000208	elu	rmsprop	mse	2	120	False	0	0	gru	0.327171
101	173	0.000707	0.000219	elu	Nadam	mse	3	60	False	0	30	lstm	0.277087
39	400	0.000709	0.000381	elu	Adam	mse	2	30	False	0	90	gru	0.297933
48	212	0.000710	0.000398	relu	Adam	mse	3	30	True	0	30	lstm	0.099257
21	373	0.000714	0.000315	elu	Adam	mse	2	30	False	0	30	lstm	0.261694
14	400	0.000716	0.000453	elu	Adam	mse	1	30	True	0	30	gru	0.312362
34	361	0.000718	0.000614	elu	Adam	mse	3	30	False	0.2	0	gru	0.254599
42	400	0.000721	0.000390	relu	rmsprop	mse	1	60	False	0	0	lstm	0.252239
4	191	0.000724	0.000372	relu	Nadam	mse	3	120	False	0.2	30	lstm	0.153979
58	351	0.000724	0.000554	elu	rmsprop	mse	1	30	True	0	0	gru	0.317311
69	372	0.000726	0.000473	relu	Nadam	mse	2	30	True	0.2	0	lstm	0.177666
41	242	0.000726	0.000161	relu	Adam	mse	2	120	True	0	90	gru	0.320478
108	226	0.000727	0.000391	relu	Adam	mse	1	120	False	0	90	gru	0.300244
53	257	0.000727	0.000296	elu	rmsprop	mse	2	120	False	0	90	gru	0.314131
59	342	0.000729	0.000473	relu	Adam	mse	2	30	False	0	90	gru	0.274595
18	400	0.000730	0.000562	relu	rmsprop	mse	1	120	False	0.2	30	lstm	0.185067
29	329	0.000730	0.000346	elu	rmsprop	mse	3	30	False	0	30	gru	0.301207
118	210	0.000730	0.000360	relu	Adam	mse	2	60	True	0	90	lstm	0.150735
77	253	0.000730	0.000369	relu	Nadam	mse	1	60	False	0	90	lstm	0.149891
37	200	0.000738	0.000408	relu	rmsprop	mse	1	120	True	0	30	gru	0.234496
82	275	0.000739	0.000625	elu	rmsprop	mse	2	60	False	0.2	90	gru	0.279774
70	346	0.000742	0.000538	relu	Nadam	mse	1	120	False	0.2	90	lstm	0.172193
24	160	0.000744	0.000394	elu	Nadam	mse	2	30	True	0	0	lstm	0.143464
74	384	0.000744	0.000534	relu	Adam	mse	3	30	False	0	0	gru	0.249823
97	320	0.000747	0.000500	relu	rmsprop	mse	1	60	False	0	0	gru	0.257723
44	400	0.000747	0.000664	elu	Adam	mse	2	30	False	0.2	90	lstm	0.088270
120	175	0.000750	0.000352	relu	Nadam	mse	2	120	False	0	90	lstm	0.100401
52	265	0.000752	0.000301	relu	Nadam	mse	2	120	True	0.2	30	lstm	0.218848
55	400	0.000754	0.000717	elu	Adam	mse	1	60	False	0.2	30	gru	0.226035
71	232	0.000755	0.000379	relu	Nadam	mse	3	30	False	0	90	lstm	0.094111
20	317	0.000756	0.000521	relu	Nadam	mse	1	60	True	0.2	90	gru	0.238386
78	281	0.000761	0.000468	relu	rmsprop	mse	3	30	False	0	0	gru	0.235401
115	227	0.000763	0.000364	elu	rmsprop	mse	3	30	False	0	0	lstm	0.238088
90	280	0.000766	0.000406	relu	Adam	mse	2	120	False	0.2	90	lstm	0.111439
91	400	0.000781	0.000731	elu	rmsprop	mse	2	30	False	0.2	30	gru	0.175891
57	400	0.000786	0.000587	relu	Adam	mse	1	30	False	0	90	lstm	0.104289
30	271	0.000787	0.000590	relu	rmsprop	mse	3	60	False	0.2	30	gru	0.065359
10	149	0.000790	0.000421	relu	rmsprop	mse	2	120	False	0	0	gru	0.224381
76	244	0.000798	0.000372	relu	rmsprop	mse	2	60	True	0	30	gru	0.219530
3	274	0.000803	0.000514	relu	rmsprop	mse	3	60	False	0.2	30	lstm	0.140990
33	327	0.000810	0.000428	relu	Nadam	mse	2	30	True	0.2	90	lstm	0.129975
80	400	0.000813	0.000717	elu	Nadam	mse	1	30	True	0.2	0	gru	0.243995
73	308	0.000815	0.000589	relu	Nadam	mse	2	30	False	0.2	90	lstm	0.090874
31	191	0.000821	0.000806	elu	Nadam	mse	1	60	True	0.2	0	lstm	0.245856
64	238	0.000821	0.000241	relu	Nadam	mse	3	30	True	0	90	gru	0.264734
113	207	0.000821	0.000497	relu	rmsprop	mse	3	120	False	0.2	90	lstm	0.102530
92	157	0.000828	0.000548	relu	rmsprop	mse	2	60	False	0	90	gru	0.092564
88	221	0.000832	0.000434	relu	rmsprop	mse	3	120	False	0.2	0	lstm	0.180532
1	400	0.000832	0.000825	relu	Nadam	mse	3	120	True	0.2	0	lstm	0.145229
5	224	0.000835	0.000371	relu	rmsprop	mse	3	60	False	0	0	gru	0.278175
50	151	0.000840	0.000673	relu	Nadam	mse	1	30	True	0	30	gru	0.173106
85	187	0.000849	0.000441	relu	Nadam	mse	2	30	True	0	0	lstm	0.144560
66	259	0.000854	0.000549	relu	Adam	mse	2	30	True	0.2	30	lstm	0.118987
106	231	0.000855	0.000460	relu	rmsprop	mse	3	60	True	0.2	30	gru	0.132336
2	373	0.000857	0.000626	relu	Adam	mse	1	60	True	0.2	90	gru	0.223455
43	206	0.000858	0.000554	elu	rmsprop	mse	2	30	True	0	90	lstm	0.137286
95	165	0.000858	0.000411	relu	Adam	mse	2	60	True	0	0	lstm	0.119521
62	172	0.000879	0.000591	relu	Adam	mse	2	60	True	0.2	90	lstm	0.120663
109	335	0.000885	0.000721	relu	Adam	mse	1	30	False	0	0	gru	0.166450
9	168	0.000889	0.000499	relu	rmsprop	mse	2	60	False	0	30	lstm	0.131224
87	292	0.000911	0.000653	relu	Adam	mse	1	60	True	0.2	30	lstm	0.102552
35	176	0.000931	0.000583	relu	rmsprop	mse	2	120	True	0.2	90	gru	0.143602
121	141	0.000969	0.000486	relu	rmsprop	mse	3	120	False	0	90	gru	0.101326
8	129	0.000989	0.000520	relu	rmsprop	mse	3	60	True	0	30	gru	0.066600
47	315	0.000995	0.000944	elu	Adam	mse	1	30	True	0.2	30	lstm	0.108665
7	145	0.001046	0.000751	relu	rmsprop	mse	2	30	False	0	90	lstm	0.076396
110	272	0.001063	0.000795	relu	Adam	mse	1	30	True	0.2	90	lstm	0.091134
56	308	0.001086	0.000851	relu	rmsprop	mse	1	30	True	0.2	0	lstm	0.141503
116	113	0.001095	0.001055	elu	Nadam	mse	2	60	True	0.2	0	lstm	0.140773
126	297	0.001096	0.000954	relu	Adam	mse	2	30	False	0.2	0	gru	0.137186
105	295	0.001110	0.000912	relu	Adam	mse	1	30	False	0	30	gru	0.133213
13	379	0.001117	0.000915	relu	rmsprop	mse	2	30	False	0.2	0	gru	0.140778
16	182	0.001174	0.001071	relu	rmsprop	mse	1	30	True	0.2	30	gru	0.126193

B.2 Sorted by test loss

run	epochs	valLoss	loss	activation	optimizer	losses	layers	size	bidirectional	dropout	dense	layer	testLoss
30	271	0.000787	0.000590	relu	rmsprop	mse	3	60	False	0.2	30	gru	0.065359
8	129	0.000989	0.000520	relu	rmsprop	mse	3	60	True	0	30	gru	0.066600
7	145	0.001046	0.000751	relu	rmsprop	mse	2	30	False	0	90	lstm	0.076396
44	400	0.000747	0.000664	elu	Adam	mse	2	30	False	0.2	90	lstm	0.088270
73	308	0.000815	0.000589	relu	Nadam	mse	2	30	False	0.2	90	lstm	0.090874
110	272	0.001063	0.000795	relu	Adam	mse	1	30	True	0.2	90	lstm	0.091134
92	157	0.000828	0.000548	relu	rmsprop	mse	2	60	False	0	90	gru	0.092564
71	232	0.000755	0.000379	relu	Nadam	mse	3	30	False	0	90	lstm	0.094111
48	212	0.000710	0.000398	relu	Adam	mse	3	30	True	0	30	lstm	0.099257
120	175	0.000750	0.000352	relu	Nadam	mse	2	120	False	0	90	lstm	0.100401
121	141	0.000969	0.000486	relu	rmsprop	mse	3	120	False	0	90	gru	0.101326
113	207	0.000821	0.000497	relu	rmsprop	mse	3	120	False	0.2	90	lstm	0.102530
87	292	0.000911	0.000653	relu	Adam	mse	1	60	True	0.2	30	lstm	0.102552
57	400	0.000786	0.000587	relu	Adam	mse	1	30	False	0	90	lstm	0.104289
98	399	0.000644	0.000491	elu	Adam	mse	3	30	False	0.2	30	lstm	0.105100
47	315	0.000995	0.000944	elu	Adam	mse	1	30	True	0.2	30	lstm	0.108665
90	280	0.000766	0.000406	relu	Adam	mse	2	120	False	0.2	90	lstm	0.111439
66	259	0.000854	0.000549	relu	Adam	mse	2	30	True	0.2	30	lstm	0.118987
95	165	0.000858	0.000411	relu	Adam	mse	2	60	True	0	0	lstm	0.119521
62	172	0.000879	0.000591	relu	Adam	mse	2	60	True	0.2	90	lstm	0.120663
16	182	0.001174	0.001071	relu	rmsprop	mse	1	30	True	0.2	30	gru	0.126193
33	327	0.000810	0.000428	relu	Nadam	mse	2	30	True	0.2	90	lstm	0.129975
9	168	0.000889	0.000499	relu	rmsprop	mse	2	60	False	0	30	lstm	0.131224
106	231	0.000855	0.000460	relu	rmsprop	mse	3	60	True	0.2	30	gru	0.132336
105	295	0.001110	0.000912	relu	Adam	mse	1	30	False	0	30	gru	0.133213
94	400	0.000657	0.000548	elu	Adam	mse	2	30	True	0.2	30	lstm	0.135762
126	297	0.001096	0.000954	relu	Adam	mse	2	30	False	0.2	0	gru	0.137186
43	206	0.000858	0.000554	elu	rmsprop	mse	2	30	True	0	90	lstm	0.137286
116	113	0.001095	0.001055	elu	Nadam	mse	2	60	True	0.2	0	lstm	0.140773
13	379	0.001117	0.000915	relu	rmsprop	mse	2	30	False	0.2	0	gru	0.140778
3	274	0.000803	0.000514	relu	rmsprop	mse	3	60	False	0.2	30	lstm	0.140990
56	308	0.001086	0.000851	relu	rmsprop	mse	1	30	True	0.2	0	lstm	0.141503
24	160	0.000744	0.000394	elu	Nadam	mse	2	30	True	0	0	lstm	0.143464
35	176	0.000931	0.000583	relu	rmsprop	mse	2	120	True	0.2	90	gru	0.143602
85	187	0.000849	0.000441	relu	Nadam	mse	2	30	True	0	0	lstm	0.144560
1	400	0.000832	0.000825	relu	Nadam	mse	3	120	True	0.2	0	lstm	0.145229
100	316	0.000679	0.000396	relu	Nadam	mse	3	60	False	0.2	0	lstm	0.147575
77	253	0.000730	0.000369	relu	Nadam	mse	1	60	False	0	90	lstm	0.149891
118	210	0.000730	0.000360	relu	Adam	mse	2	60	True	0	90	lstm	0.150735
4	191	0.000724	0.000372	relu	Nadam	mse	3	120	False	0.2	30	lstm	0.153979
6	173	0.000694	0.000306	elu	Adam	mse	3	60	True	0	0	lstm	0.159340
109	335	0.000885	0.000721	relu	Adam	mse	1	30	False	0	0	gru	0.166450
70	346	0.000742	0.000538	relu	Nadam	mse	1	120	False	0.2	90	lstm	0.172193
50	151	0.000840	0.000673	relu	Nadam	mse	1	30	True	0	30	gru	0.173106
91	400	0.000781	0.000731	elu	rmsprop	mse	2	30	False	0.2	30	gru	0.175891
69	372	0.000726	0.000473	relu	Nadam	mse	2	30	True	0.2	0	lstm	0.177666
88	221	0.000832	0.000434	relu	rmsprop	mse	3	120	False	0.2	0	lstm	0.180532
18	400	0.000730	0.000562	relu	rmsprop	mse	1	120	False	0.2	30	lstm	0.185067
99	400	0.000700	0.000577	elu	Adam	mse	1	60	True	0.2	90	lstm	0.206794
38	329	0.000684	0.000241	relu	Adam	mse	3	60	False	0	30	lstm	0.211797
52	265	0.000752	0.000301	relu	Nadam	mse	2	120	True	0.2	30	lstm	0.218848
76	244	0.000798	0.000372	relu	rmsprop	mse	2	60	True	0	30	gru	0.219530
2	373	0.000857	0.000626	relu	Adam	mse	1	60	True	0.2	90	gru	0.223455
10	149	0.000790	0.000421	relu	rmsprop	mse	2	120	False	0	0	gru	0.224381
55	400	0.000754	0.000717	elu	Adam	mse	1	60	False	0.2	30	gru	0.226035
37	200	0.000738	0.000408	relu	rmsprop	mse	1	120	True	0	30	gru	0.234496
78	281	0.000761	0.000468	relu	rmsprop	mse	3	30	False	0	0	gru	0.235401
115	227	0.000763	0.000364	elu	rmsprop	mse	3	30	False	0	0	lstm	0.238088
20	317	0.000756	0.000521	relu	Nadam	mse	1	60	True	0.2	90	gru	0.238386
80	400	0.000813	0.000717	elu	Nadam	mse	1	30	True	0.2	0	gru	0.243995
31	191	0.000821	0.000806	elu	Nadam	mse	1	60	True	0.2	0	lstm	0.245856
86	400	0.000657	0.000385	elu	Nadam	mse	3	30	False	0.2	0	lstm	0.246972
17	196	0.000638	0.000348	elu	Adam	mse	1	120	True	0	0	lstm	0.249039
74	384	0.000744	0.000534	relu	Adam	mse	3	30	False	0	0	gru	0.249823
42	400	0.000721	0.000390	relu	rmsprop	mse	1	60	False	0	0	lstm	0.252239
34	361	0.000718	0.000614	elu	Adam	mse	3	30	False	0.2	0	gru	0.254599
15	265	0.000673	0.000269	elu	Adam	mse	2	60	True	0	0	lstm	0.254788
119	260	0.000680	0.000433	relu	Nadam	mse	2	60	True	0.2	0	gru	0.254822
68	400	0.000685	0.000543	elu	rmsprop	mse	1	120	False	0.2	90	lstm	0.255893
97	320	0.000747	0.000500	relu	rmsprop	mse	1	60	False	0	0	gru	0.257723
21	373	0.000714	0.000315	elu	Adam	mse	2	30	False	0	30	lstm	0.261694
112	400	0.000587	0.000477	elu	Adam	mse	2	60	False	0.2	90	gru	0.262152
46	339	0.000638	0.000418	relu	Adam	mse	2	120	False	0.2	0	gru	0.263046
19	400	0.000662	0.000483	relu	Nadam	mse	1	60	True	0.2	0	lstm	0.263520
64	238	0.000821	0.000241	relu	Nadam	mse	3	30	True	0	90	gru	0.264734
12	312	0.000623	0.000328	relu	Adam	mse	3	120	False	0.2	0	lstm	0.266114
83	400	0.000622	0.000497	elu	rmsprop	mse	1	120	False	0.2	0	lstm	0.271461
59	342	0.000729	0.000473	relu	Adam	mse	2	30	False	0	90	gru	0.274595
101	173	0.000707	0.000219	elu	Nadam	mse	3	60	False	0	30	lstm	0.277087
5	224	0.000835	0.000371	relu	rmsprop	mse	3	60	False	0	0	gru	0.278175
82	275	0.000739	0.000625	elu	rmsprop	mse	2	60	False	0.2	90	gru	0.279774
111	364	0.000679	0.000414	elu	Adam	mse	1	30	True	0	90	lstm	0.286464
61	262	0.000631	0.000485	elu	Adam	mse	2	120	True	0.2	30	lstm	0.286525
36	237	0.000676	0.000173	relu	Nadam	mse	2	120	False	0	30	lstm	0.294147
0	400	0.000671	0.000596	elu	Nadam	mse	1	60	False	0.2	30	gru	0.297633
39	400	0.000709	0.000381	elu	Adam	mse	2	30	False	0	90	gru	0.297933
108	226	0.000727	0.000391	relu	Adam	mse	1	120	False	0	90	gru	0.300244
29	329	0.000730	0.000346	elu	rmsprop	mse	3	30	False	0	30	gru	0.301207
32	272	0.000694	0.000178	relu	Nadam	mse	3	120	False	0	0	lstm	0.303562
128	400	0.000601	0.000503	elu	rmsprop	mse	1	120	True	0.2	0	gru	0.303748
127	235	0.000641	0.000466	elu	rmsprop	mse	3	60	True	0.2	30	gru	0.303977
114	286	0.000646	0.000384	relu	Adam	mse	2	60	False	0	0	gru	0.305044
22	324	0.000686	0.000306	relu	Adam	mse	2	120	True	0.2	30	gru	0.309502

run	epochs	valLoss	loss	activation	optimizer	losses	layers	size	bidirectional	dropout	dense	layer	testLoss
103	400	0.000584	0.000416	elu	Nadam	mse	1	120	False	0.2	0	gru	0.310417
49	400	0.000702	0.000475	elu	rmsprop	mse	1	30	True	0	0	lstm	0.310804
79	160	0.000676	0.000228	elu	Nadam	mse	2	120	False	0	90	lstm	0.311716
14	400	0.000716	0.000453	elu	Adam	mse	1	30	True	0	30	gru	0.312362
75	216	0.000652	0.000186	elu	Nadam	mse	2	60	True	0	0	lstm	0.313548
54	400	0.000642	0.000359	elu	Adam	mse	1	60	False	0	0	gru	0.314058
53	257	0.000727	0.000296	elu	rmsprop	mse	2	120	False	0	90	gru	0.314131
107	400	0.000683	0.000419	elu	Adam	mse	2	30	False	0	0	gru	0.315303
27	155	0.000595	0.000282	relu	Nadam	mse	1	120	True	0	0	gru	0.315502
51	359	0.000677	0.000382	elu	rmsprop	mse	1	60	True	0	0	gru	0.316340
45	177	0.000699	0.000327	elu	rmsprop	mse	3	120	False	0	0	gru	0.316596
58	351	0.000724	0.000554	elu	rmsprop	mse	1	30	True	0	0	gru	0.317311
60	166	0.000654	0.000315	relu	Nadam	mse	2	60	False	0	90	gru	0.318951
72	258	0.000638	0.000400	elu	Nadam	mse	3	120	True	0	90	gru	0.320142
41	242	0.000726	0.000161	relu	Adam	mse	2	120	True	0	90	gru	0.320478
117	400	0.000618	0.000283	relu	Nadam	mse	2	120	False	0.2	0	lstm	0.321786
93	355	0.000593	0.000305	elu	Nadam	mse	3	60	False	0.2	30	gru	0.323147
124	329	0.000560	0.000356	elu	Nadam	mse	2	120	False	0.2	0	lstm	0.324017
81	263	0.000563	0.000248	elu	Adam	mse	3	120	True	0.2	0	gru	0.325456
26	169	0.000594	0.000226	elu	Adam	mse	3	120	True	0	90	gru	0.326938
40	309	0.000707	0.000208	elu	rmsprop	mse	2	120	False	0	0	gru	0.327171
25	383	0.000678	0.000351	relu	Adam	mse	3	120	False	0.2	0	gru	0.327694
125	318	0.000648	0.000433	elu	Adam	mse	2	60	True	0.2	0	gru	0.329163
28	189	0.000690	0.000317	elu	rmsprop	mse	2	120	False	0	90	lstm	0.329985
23	264	0.000609	0.000320	elu	rmsprop	mse	3	30	True	0	0	gru	0.335117
104	172	0.000649	0.000267	elu	Nadam	mse	3	60	False	0	90	gru	0.337518
96	237	0.000703	0.000299	relu	Nadam	mse	1	60	True	0	30	gru	0.338350
67	347	0.000695	0.000445	relu	Nadam	mse	1	30	True	0	0	gru	0.338427
11	197	0.000563	0.000310	elu	Nadam	mse	3	120	False	0.2	90	gru	0.340967
122	293	0.000630	0.000298	elu	Adam	mse	2	60	False	0	0	gru	0.341811
102	309	0.000571	0.000321	elu	Adam	mse	2	120	True	0.2	0	gru	0.342152
89	400	0.000580	0.000426	elu	rmsprop	mse	2	60	True	0.2	90	gru	0.349329
63	215	0.000635	0.000176	elu	Nadam	mse	1	120	False	0	0	gru	0.353077
65	400	0.000579	0.000250	elu	Adam	mse	2	30	True	0	90	gru	0.354498
84	317	0.000571	0.000241	elu	Nadam	mse	3	120	False	0.2	90	lstm	0.355334
123	380	0.000559	0.000270	elu	Adam	mse	1	60	True	0	90	gru	0.377228