

Lappeenranta-Lahti University of Technology LUT
School of Engineering Science
Computational Engineering and Technical Physics
Computer Vision and Pattern Recognition

Felix Hildén

Unsupervised Profiling of Electricity Consumers for Load Forecasting Applications

Master's Thesis

Examiners: Professor, D.Sc. (Tech) Lasse Lensu
Associate Professor, D.Sc. (Tech) Matylda Jablonska-Sabuka

Supervisors: Associate Professor, D.Sc. (Tech) Matylda Jablonska-Sabuka
Associate Professor, D.Sc. (Tech) Arto Kaarna
Professor, D.Sc. (Tech) Lasse Lensu

Abstract

Lappeenranta-Lahti University of Technology LUT
School of Engineering Science
Computational Engineering and Technical Physics
Computer Vision and Pattern Recognition

Felix Hildén

Unsupervised Profiling of Electricity Consumers for Load Forecasting Applications

Master's Thesis

2020

72 pages, 30 figures, 5 tables, 4 appendices.

Examiners: Professor, D.Sc. (Tech) Lasse Lensu
 Associate Professor, D.Sc. (Tech) Matylda Jablonska-Sabuka

Keywords: load forecasting, deep learning, clustering, time series

Electric power suppliers are typically interested in profiling their customers. Different consumer categories or load profiles can give insight into behavior which affects the power grid and facilitate load forecasting and marketing efforts. Statistical methods and deep learning enable forecasting based on the historical consumption of individual customers. In this thesis, the possibilities of clustering and short-term load forecasting were examined with a dataset from a Finnish utility company. Short slices of consumers' historical load were clustered with k -means using features from an autoencoder. A state-of-the-art time series forecasting model, ForecastNet, was trained first for all consumers together and then separately for each cluster. Although reference models were outperformed, prior clustering did not improve forecasting results. Cluster hardening, an additional loss designed to force a good clustering result for the autoencoding process, was found to be ill-defined and as such was not used to supplement the dimensionality reduction for clustering.

Tiivistelmä

Lappeenrannan-Lahden teknillinen yliopisto LUT
School of Engineering Science
Laskennallinen tekniikka ja teknillinen fysiikka
Konenäkö ja hahmontunnistus

Felix Hildén

Sähkönkuluttajien ohjaamaton ryhmittely kulutuksen ennustamiseksi

Diplomityö

2020

72 sivua, 30 kuvaa, 5 taulukkoa, 4 liitettä.

Tarkastajat: Professori, TkT Lasse Lensu
 Tutkijaopettaja, Matylda Jablonska-Sabuka

Hakusanat: sähkönkulutuksen ennustaminen, syväoppiminen, ryhmittely, aikasarja

Sähkøyhtiöt ovat tavallisesti kiinnostuneita asiakkaidensa profiloinnista. Eri kuluttajaryhmät ja kulutustottumukset voivat valottaa sähköverkon asiakkaiden käyttäytymistä ja helpottaa kulutuksen ennustamista sekä markkinointia. Tilastollisten menetelmien ja syväoppimisen avulla kulutustietojen perusteella voidaan ennustaa yksittäisten asiakkaiden tulevaa kulutusta. Tässä työssä tarkasteltiin ryhmittelyn ja lyhyen aikavälin kulutuksen ennustamisen mahdollisuuksia erään suomalaisen sähkøyhtiön kulutustietoja käyttäen. Kuluttajien tiedoista leikattiin lyhyitä jaksoja, jotka ryhmiteltiin k -means-algoritmin avulla autoenkooderista saatuja piirteitä käyttäen. Tuore ForecastNet-malli koulutettiin ennustamaan ensin kaikkien jaksoiden tulevaa kulutusta, ja sitten erikseen joka ryhmälle. Vaikka ForecastNet suoriutui vertailumalleja paremmin, ryhmittely ei parantanut ennustustulosta. Autoenkooderin perusteella tehdyn ryhmittelyn parantamiseen tarkoitettua lisätavoitteen, Cluster hardeningin, huomattiin olevan määritelty väärin, eikä sitä käytetty ulottuvuuksien vähentämisessä.

CONTENTS

1	Introduction	8
1.1	Background	8
1.2	Objectives and delimitations	10
1.3	Structure of the thesis	11
2	Electricity consumption analysis	12
2.1	Load profiling	12
2.2	Load forecasting	13
3	Time series analysis	15
3.1	Definition of time series	15
3.2	Introduction to neural networks	15
3.3	Time series clustering	16
3.4	Evaluation of time series clustering	18
3.5	Time series forecasting	19
3.6	Evaluation of time series forecasting	21
4	Electricity consumption data	23
4.1	Time series and background data	23
4.2	Data quality and preprocessing	24
4.3	Forecast length and data splits	26
4.4	Data exploration with PCA	27
5	Proposed methods	31
5.1	Clustering	31
5.1.1	Autoencoder	31
5.1.2	k -means	32
5.1.3	Evaluation of clustering	33
5.2	Load forecasting	34
5.2.1	ForecastNet	34
5.2.2	Baseline models	34
5.2.3	Evaluation of forecasting	35
5.3	Neural network hyperoptimisation	36
6	Experiments	38
6.1	Implementation	38
6.2	Data division and normalisation	38
6.3	Clustering with shallow autoencoder	40

	5
6.4 Unclustered forecasting	44
6.5 Forecasting with clusters from dense autoencoder	45
6.6 Clustering with joint autoencoding and cluster hardening	47
7 Results	50
8 Discussion	54
9 Conclusion	56
References	57
Appendices	
Appendix 1: Clustering with shallow autoencoder figures	
Appendix 2: Unclustered forecasting figures	
Appendix 3: Forecasting with clusters from shallow autoencoder figures	
Appendix 4: Best and worst predictions of unclustered and clustered models	

Preface

It has been almost six months now, since I started to work on this thesis. In that time I've come to a humbling conclusion. *New* does not mean *better*. Nor does *complex*. This project relates to the Smart Services for Digitalisation (DIGI-USER) research platform at LUT University. They are about to read that predicting electricity consumption with the last day's values is almost as effective as throwing all the power of a modern neural network at it. I refuse to believe it is a failure. It's a long-winded confirmation of industry-tested practices.

I would like to thank Azat Garifullin for his help with debugging the cluster hardening method, and Samuli Honkapuro, Jukka Lassila and Petri Valtonen for lending their knowledge about the electricity industry. 'Thank you' to LENS and my wonderful supervisors for keeping me at work, to my family and friends for taking me away, and to Aalef OY for keeping me well fed. Most importantly, thank *you* for reading.

Lappeenranta, October 30, 2020

Felix Hildén

Acronyms

ARIMA	Autoregressive Integrated Moving Average
ARMA	Autoregressive Moving Average
CNN	Convolutional Neural Network
DNN	Deep Neural Network
DTC	Deep Temporal Clustering
DTW	Dynamic Time Warping
FCR	Fuzzy Relation
FKM	Fuzzy k -Means
LSH	Locality Sensitive Hashing
LSTM	Long Short-term Memory
MAAPE	Mean Arctangent Absolute Percentage Error
MAE	Mean Absolute Error
MASE	Mean Absolute Scaled Error
MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
PCA	Principal Component Analysis
ReLU	Rectified Linear Unit
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SVM	Support Vector Machine

1 Introduction

1.1 Background

Electric power suppliers are typically interested in profiling their consumers. It can give insight into human behavior which affects the power grid, aiding utility companies in load and infrastructure planning as well as marketing. Particularly, accurate forecasts of individual consumption and overall system load would be interesting for load planning. For this thesis project, an unexplored data set of individual consumers of electricity is available. It is used as a basis for examining the effect of clustering to load forecasting. This section continues with a brief introduction to electricity production for context, after which the objectives and delimitations of this thesis are presented.

Electric grids are networks of producers and consumers of electrical power [1]. The network itself leverages multiple stages: high-voltage lines for long-distance transmission and lower-voltage distribution lines that are connected to individual consumers. Meeting large demands of power over long distances is more economical using high-voltage lines, because their capacity for carrying power is greater. Additionally, transferring at high voltages is more efficient, because the corresponding current can be kept lower. As a result, losses to conductor resistance decrease and wires can be slim. When designing such grids, their demand and capacity must be carefully considered far into the future¹. City plans are typically consulted to assess the infrastructure needs.

At any given time the total power that suppliers produce is determined by the amount of electricity consumers use [2]. Significant imbalances could result in network instability or failures. Demand varies yearly [3], as depicted in Figure 1. Overall differences between production and consumption are explained by the import and export of electricity. Due to the cold winter in Finland, most of the variation during a year is explained by weather, but load may shift by up to 2 GW, or 10-20 % daily. The minimum demand during any time period is the baseload of a grid, here approximately 8 GW. The maximum load is called peak demand, here approximately 14 GW. Changes in consumption are met by adjusting production and using different types of storage. Excess power is most commonly stored as potential energy, in hydroelectric plants or with other mechanisms that can increase the potential of heavy objects and harness that energy later.

When demand cannot be met, load must be shed, resulting in blackouts or brownouts in

¹Information from interview with Samuli Honkapuro and Jukka Lassila.

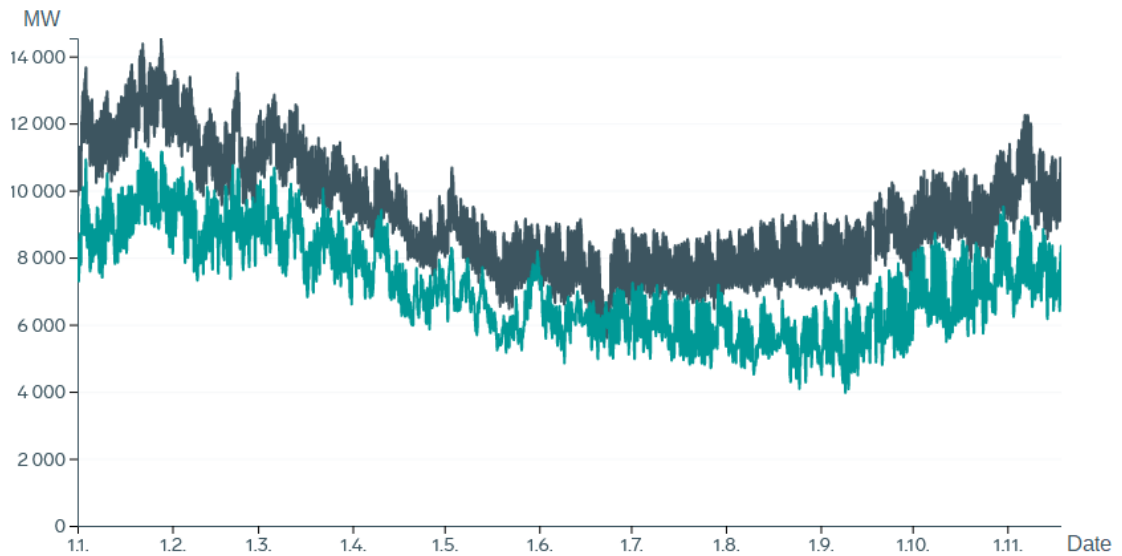


Figure 1. Grid load and generated power during 2019 as reported by Fingrid [3]. Load is drawn in dark blue, generation in light blue.

regions of a grid or the whole network. This is undesirable, so different measures are taken to prevent it. A producer may negotiate with large consumers in an attempt to reduce load at peak demand. The price of electricity may also be increased during peak times. This encourages consumers to wait until prices are lower again and in turn reduces load on the grid. Conversely, a network may also be overloaded with production, in which case users are encouraged or even paid to consume electricity.

An increasing number of smaller parties participate in producing electricity for a network. This is called distributed generation [4]. If it is allowed, a home owner may want to sell electricity generated with e.g. solar panels during lucrative times, for example when the grid is at peak demand. This reduces the stress put on larger producers to meet peak demand with expensive equipment, and lowers prices. These ideas are further developed in smart electric grids. They are still being researched, but they would feature decentralised distribution and two-way connections between producers and consumers. Users would be equipped with smart meters that can encourage consumption at non-peak times or even appliances whose operation can be scheduled.

Predicted consumption is an important factor to consider in electricity producers' decision making, as noted by Tso and Yau [5]. Statistical approaches such as regression have been widely used in forecasts, but research on machine learning and Deep Neural Networks (DNNs) has surged in popularity, and the field of time series analysis has advanced with it. This has led to the application of deep learning methods to time series clustering [6–8] and forecasting [9–11]. While the most prominent applications of temporal deep learning

include domains such as video, sound and stock prediction [7], machine learning has been applied to electricity consumption too. Though most of the publications seem to consider simple feed-forward neural networks, Support Vector Machines (SVMs) and hybrid models [12–14], recently, Chen et al. used a more convoluted approach to clustering and forecasting electricity consumption with deep learning [15].

Given the utility of load forecasting, the prevalence of deep learning in the general data analysis community and the unexplored data set that was made available, it would be interesting to apply state-of-the-art methods to produce load forecasts. Laurinec et al. observed that making predictions based on appropriate groups of consumers rather than the whole population yielded better forecasts [16]. However, this was not true of deep learning methods, which achieved similar results without prior clustering, most likely due to their superior capacity of representing complex problems.

1.2 Objectives and delimitations

In this thesis, hourly consumption data recorded over a year from 3400 households is available for analysis. A method for clustering consumers and predicting their short-term energy use is implemented and evaluated, as visualised in Figure 2. Despite the findings in [16], a state-of-the-art deep learning method is used for forecasting to honestly evaluate the value of prior clustering for the data set in question. More concretely, the objectives of this thesis are as follows.

1. Review literature related to load forecasting and state-of-the-art time series analysis.
2. Develop a method for predicting consumption by using consumer clustering (Fig. 2). An autoencoder is trained to condense and reconstruct input series. Hidden states of that encoder are used as input to an additional clustering loss that affects the learning objective of the network. After training the autoencoder, the latent space is used as input to k -means with which a cluster is assigned to each series. Finally a predictive model is constructed for each cluster to predict consumption.
3. Experiment with the method using a real-world data set and evaluate its performance with quantitative metrics.

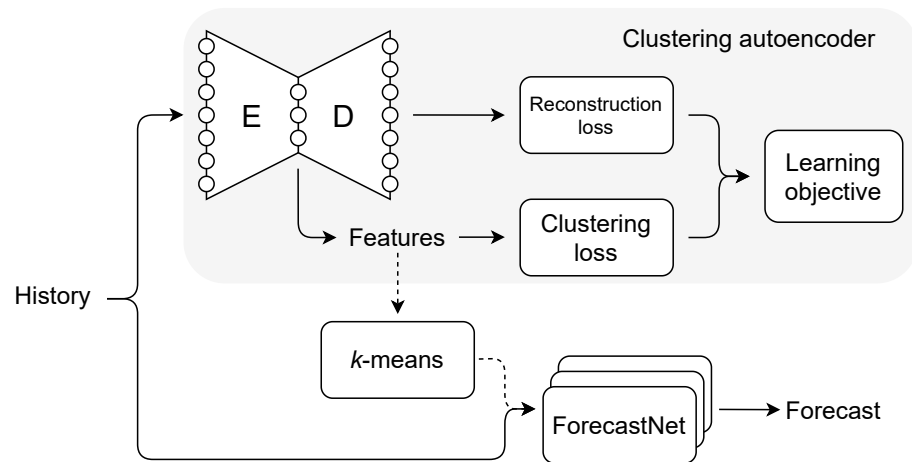


Figure 2. Proposed solution for clustered forecasting.

1.3 Structure of the thesis

The rest of this thesis is structured as follows. First, a literary review is carried out. Literature related to the domain of electricity consumption and load profile analysis is discussed in Chapter 2. Then a more generic view is presented in Chapter 3, where conventional and state-of-the-art methods for time series analysis are introduced.

The practical portion is opened in Chapter 4 with a description and tentative analysis of the data set that was used. A methodology for clustering and forecasting electricity consumption is proposed in Chapter 5, and used in Chapter 6 for experimentation. The results of those experiments are presented in Chapter 7 and discussed more broadly in Chapter 8 with analysis of the shortcomings of this thesis and ideas for future work. Finally, the thesis is concluded in Chapter 9.

2 Electricity consumption analysis

2.1 Load profiling

The aim of consumer profiling is to group them with respect to some relevant criteria. Load profiling specifically aims to extract information related to electricity consumption using background data of a consumer, the building that they occupy or past consumption, and to build a profile estimating the typical load of that consumer. There are conventionally two types of load profile models [17], ones based on the area of residency and others based on groups of similar consumers. For large-scale load forecasting it is typical to construct mean consumption profiles for each type of consumer taking into account weekends and holidays². Variations in behavior are then balanced by the number of consumers in the region of interest, making the model viable as a general forecast.

Even though load profiling is a very similar endeavour to clustering in general, the two should be distinguished. Profiles can be constructed without using clustering techniques. However, in the context of this thesis profiling and consumption-based clustering are practically synonymous. A plethora of methods have been proposed for determining consumer load profiles through clustering. As Chicco et al. summarise in [18], techniques range from statistical approaches and fuzzy logic to neural networks. In their research, two algorithms, Follow The Leader and Hierarchical Clustering, were reported to achieve the best results in terms of cluster validity. They produced a clear separation of clusters and isolated uncommon profiles. For example, representative load patterns can be formulated based on each group of consumers. Figure 3 visualises two such hypothetical patterns that could be found by averaging weekly consumption series.

In [19], Yao and Steemers proposed a method for predicting load profiles for electricity consumption and heating based on a number of preconceived occupancy scenarios and electrical appliances. However, this kind of hand-crafted model has a downside. It must be redesigned from scratch in a different environment or domain. McLoughlin et al. have named similar solutions "engineering approaches" in [20] and also note that they are difficult to generalise.

Zakaria et al. used fuzzy clustering in [17] to determine daily load profiles for various types of consumers. They compared the performance of fuzzy clustering based on Fuzzy Relation (FCR) to Fuzzy k -Means (FKM) with several indices. Of the two algorithms

²Information from interview with Samuli Honkapuro and Jukka Lassila.

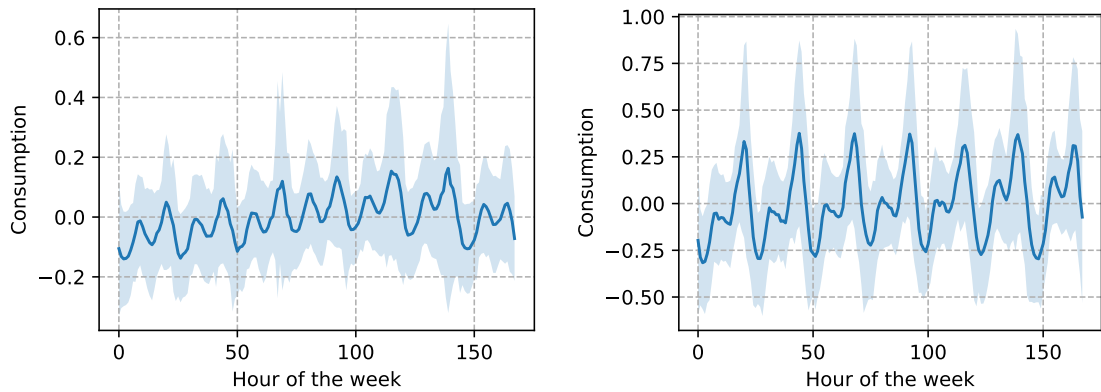


Figure 3. Example weekly load profiles with deviation (mean subtracted).

FKM was reportedly better. All three clusters corresponded with a category in the data background information.

A three-stage methodology was used in [20] by McLoughlin et al. to produce daily load profiles based on smart metering data and to classify individual consumers to one of the categories. First, clustering was applied to segment the data. Self-organising maps were preferred to k -means and k -medioids on the basis of Davies-Bouldin validity index. Then daily loads for each cluster were averaged to create a representative load pattern. Finally each day of a consumer was assigned to one of the load profiles. It was noted that customers could be assigned to many profiles depending on the day. To assign only one class to each customer the statistical mode of daily assignments was calculated.

Cluster features can be automatically extracted from consumption data using autoencoders. They are a class of neural networks designed to encode information about inputs, thus reducing dimensionality, and then decode it back from that representation as accurately as possible. In [21], Varga et al. used an autoencoder to extract relevant features from consumption data, and clustered consumers with Locality Sensitive Hashing (LSH). Their research also included a software solution for efficiently clustering new consumers.

2.2 Load forecasting

Load forecasting is essential particularly in the electricity market³. Different applications of load estimation can be categorised in two ways, length of prediction horizon and degree of aggregation. Short-term forecasting refers to prediction horizons ranging from hours

³Information from interview with Petri Valtonen.

up to one week, a suitable time period for load planning. Forecasts spanning from months (medium-term) to several years (long-term) are used for estimating future demand and infrastructure planning [12]. Forecasts can be made at many levels of analysis. They may target overall consumption at state or network level [22], load at specific distribution nodes, buildings [13, 23] or even individual appliances [5].

Hammad et al. have performed a comprehensive review on the available load forecasting literature [11]. Even though machine-learning-based approaches are favored in short-term forecasting, regression is still widely used and efficient long term. In practice, utility companies often employ these simpler approaches, like using historical load data and correcting for weather, because of their speed and accuracy with large masses of customers and a short time frame⁴. Various classes of techniques exist outside traditional statistical approaches and machine learning too, like state space modelling and frequency- or phase-space analysis [11].

In [5], Tso and Yau compared regression to neural networks and decision trees in a forecasting task and reported comparable performance between the methods. Similarly, Laurinec et al. compared various means of forecasting in [16], reporting that DNNs did not benefit from clustering consumers prior to forecasting, but neural networks had a similar performance to statistical methods. Both Ahmad et al. [12] and more recently Daut et al. [14] compared neural networks and SVMs in reviews of applications for forecasting the electricity consumption of buildings. Deb et al. [24] conducted an extensive review of research that used time series forecasting methods to predict building energy consumption, and concluded that machine learning methods fare well.

Chen et al. [15] used a more convoluted approach, combining autoencoders and exogenous data with forecasting. However, their article made a questionable architecture choice of using convolutions with spatially and temporally independent variables and lacked details about data usage. In general, the most recent methods in the broader field of time series forecasting, which are discussed in Chapter 3, seem to have seen little use in research specialising in load forecasting.

⁴Information from interview with Petri Valtonen.

3 Time series analysis

3.1 Definition of time series

A discrete time series is a sequence of timestamp-observation pairs (t_i, \mathbf{x}_i) [6]. The timestamps are ordered in time, but need not be equally spaced. Observations may be uni- or multivariate, and contain arbitrary information. Two time series, one with equally-spaced points and another with an unequal sampling interval, are visualised in Figure 4. If they are equally spaced, the values of timestamps can be discarded and the data considered a simple sequence of points \mathbf{x}_i . When multiple series are analysed together, they must share the same sampling interval or have known absolute time stamps to be comparable. Additional complications for processing such data may arise from missing points or noise.

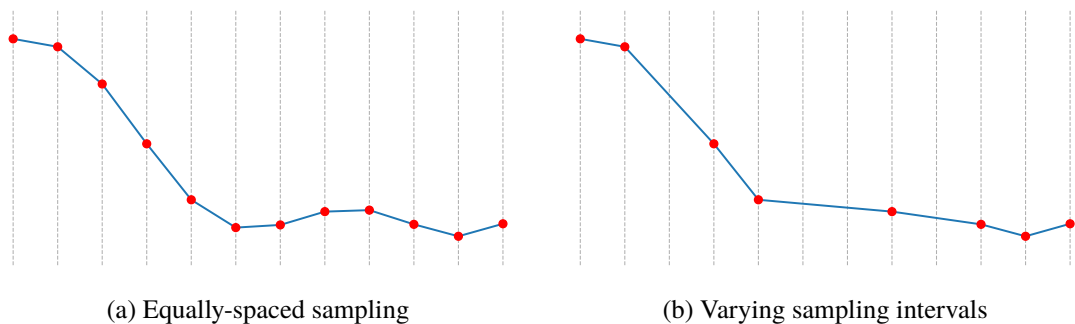


Figure 4. Example time series with different sampling.

3.2 Introduction to neural networks

Neural networks are a class of computational devices suitable for generic function approximation [25]. As such they can be used in various time series analysis tasks. They have become a go-to tool in many fields since computing power and memory has become relatively cheap, allowing for increasingly large networks to perform more complex tasks. Solving a classification or regression problem is done by backpropagation, wherein data is repeatedly fed through a network and compared to a target. The deviation of current output from a target is defined by a loss function. Error signals are then fed back through the network, changing parameter values towards a more suitable configuration. This utilising of inputs and targets in optimising a model is more broadly known as supervised learning.

Neural networks are composed of layers – computing blocks – with which an input is

processed until an output is produced. The most basic layer is a dense layer, also known as a fully-connected layer, which performs a matrix multiplication with a set of learned weights to produce an output for the next layer to consume. Ordinarily, after each such layer a non-polynomial *activation* function is applied. They allow a sufficiently large network to represent arbitrarily complex functions [25], unlike a network with only dense layers, which could be reduced to a single matrix multiplication and thus be capable of only representing linear models. An activation function is sometimes also represented as a separate layer. Other commonly used layers include the convolutional layer, in which a convolution operator is applied to the incoming data with a set of learned kernels, and the recurrent layer, which has an internal state that is combined with an input and updated at each time step. These types of layers can be used when dealing with space- or time-dependent data. Recurrent and convolutional layers define the way a network processes input, which is why architectures that use them are broadly named Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs).

There are two principal challenges to training neural networks. The first, overfitting [26], is an issue with function approximation itself. A sufficiently complex model can memorise data instead of learning its relevant characteristics, leading to poor performance when novel data is presented. Many regularisation techniques exist for mitigating the overfitting problem, like reducing model complexity, gathering more data or slightly perturbing it to produce more instances from each sample – known as data augmentation, weight decay, dropout, mini-batch learning and adjusting learning rate. The second problem, gradient vanishing [27], relates to the optimisation of a deep network via backpropagation. If error signals are propagated through layers whose partial derivatives have a tendency to drive gradients towards zero, especially after several layers, learning becomes slow. Gradient vanishing can be prevented by architectural choices like residual connections and selecting a suitable activation function like a Rectified Linear Unit (ReLU). On the other hand, gradient values can also explode in some situations leading to unstable learning. Remedies are similar to the vanishing gradient case.

3.3 Time series clustering

Clustering is a form of unsupervised learning, where in contrast to supervised learning, no target function is available for modelling. As such it is suitable for discovering relationships in the data as opposed to learning a specific task. Using time series in clustering is challenging in a number of ways in comparison to conventional clustering tasks. They are by nature high-dimensional, as they can be used to represent multiple variables across

long spans of time or with short sampling intervals. Additionally, choosing a suitable similarity measure is not trivial. For example, a series is intuitively similar with a time-shifted version of itself, but e.g. Euclidean distance does not reflect this. Warren Liao summarises these challenges in [28] and categorises approaches to time series clustering to model-based, feature-based and raw-data-based methods. Each is a different way of representing the series in question, with either parameters of some model applied to the data, features computed from the data, or the raw series themselves.

In [6], Wang et al. compared different time series representation methods and distance measures. Their comparison was more comprehensive than in many other publications introducing new methods in the sense that they carefully validated results across a wide variety of data sets from different domains. Different representation methods were evaluated based on their indexing power, that is, the tightness of their lower bounds. Representation methods had similar indexing power in all the studied cases. Likewise, novel distance measures were reported to perform at the level of long-established methods like Euclidean distance and Dynamic Time Warping (DTW). They can be more accurate and quicker to compute with small data sets, but as the data set grows, their computational cost approaches that of Euclidean distance and DTW, while Euclidean distance and DTW catch up in accuracy.

Aljalbout et al. proposed a taxonomy for clustering with deep learning in [8]. They consolidated different approaches to feature extraction, loss functions, their combinations and training schedules. These approaches are visualised in Figure 5. Different network architectures that reduce the input dimensionality can be trained with either a non-clustering loss, a clustering loss or a combination thereof. Features may be extracted either from a single layer of the network or in some cases from multiple layers for a richer representation. They also proposed a new method based on the taxonomy, training a convolutional autoencoder with reconstruction loss and cluster hardening loss.

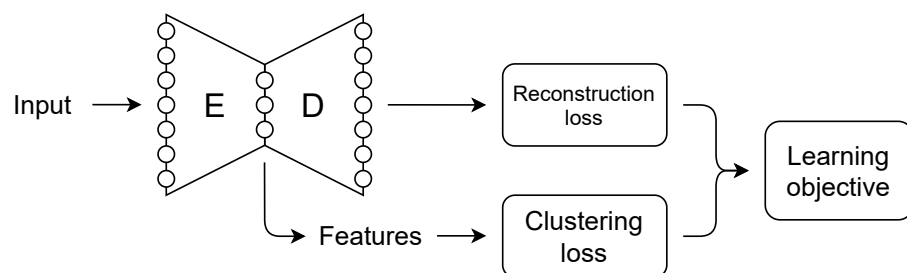


Figure 5. Components of deep clustering [8].

An autoencoder is a neural network designed to encode information into a condensed

representation and decode it back to match the original input. As such, they learn a feature set capable of representing an input in reduced detail. This latent space of features can then be used, for example, in clustering. The architecture of an autoencoder is arbitrary. It can even be non-symmetric, but having a similar architecture in both the encoder and decoder enforces that neither of them become a bottleneck in the learning process.

In accordance with the taxonomy in [8], Deep Temporal Clustering (DTC), an algorithm proposed by Madiraju et al. in [29] utilises unsupervised feature learning for time series clustering. DTC uses an autoencoder to create a lower-dimensional representation of a series. The encoder uses a convolutional layer and bidirectional Long Short-term Memory (LSTM) layers to construct the latent representation and an upsampling layer followed by a deconvolution to decode the representation. A temporal clustering layer is fed with representations. Learning is done by optimising two losses, the error of decoding and the clustering metric. Thus, the learned features are able to represent the series well, but also to separate different series. In contrast, performing dimensionality reduction and clustering independently could lead to worse results, because the reconstruction error is minimised first and the separation is optimised only after that. Using DTC with a distance measure was shown to consistently improve results over clustering with the measure alone. Evaluation was done by using the clusters as a classifier.

3.4 Evaluation of time series clustering

Aghabozorgi et al. have surveyed various ways of evaluating time series clustering results in [30]. They are broadly categorised into two groups, external metrics which utilise outside information to assess the clustering result, and internal metrics derived from data or the clustering method itself to evaluate cluster validity. Most commonly, external evaluation involves using labelled data for which the labels directly correspond to the desired clusters. A natural metric arises, *cluster purity*, which measures the ratio of correctly assigned labels to the whole data set or each individual cluster. In the best case, each group would only contain data with a single label. This has an obvious limitation, however, as increasing the number of clusters leads to a perfect result. But if the appropriate number of clusters is known, purity can be used to a great effect. A plethora of other external metrics are also widely used, each focusing on a different aspect like cluster similarity, the F-measure or entropy.

Without ground truth, as is common with many data sets, internal metrics need to be used. Liao notes in [28], that they can be further divided into two categories based on

whether a priori knowledge about the appropriate number of clusters exists or not. Many clustering algorithms like k -means requires the number of clusters to be specified as an input, but by examining time-tested indices like Silhouette [31] or Davies-Bouldin [32] one can determine the optimal number of clusters.

3.5 Time series forecasting

As with clustering, time series forecasting has a unique set of challenges. Observations are often highly dependent on values at previous time steps, which can be scarce in comparison to the problem's complexity. Series can be dynamic in the sense that not only do their values change, but over time their nature may also change. Additionally, any number of known or unknown external factors may contribute to the realisation of subsequent time steps. These challenges have led an already rich field to move from exploring statistical methods to now utilising deep learning in order to represent the complex nature of many time series better.

In the simplest case forecasts utilise a history to estimate future values (Fig. 6). The length of this forecast is called a *horizon*. Both statistical methods and deep learning are capable of expanding the history to include exogenous variables – external factors that affect the forecast – either as additional series or constant values.

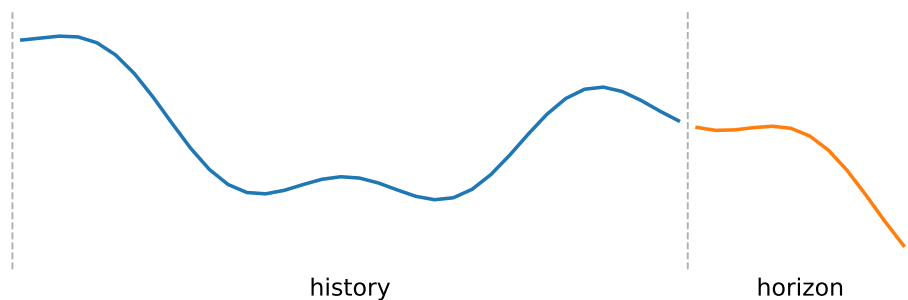


Figure 6. Forecast components.

Autoregressive Moving Average (ARMA) models describe a stationary stochastic process with two components, an autoregressive (AR) term of past values and a moving average (MA) term of previous residuals [33]. Both are expressed as a linear combination up to a certain number of previous values. ARMA can be generalised to non-stationary processes with an integration term, yielding the Autoregressive Integrated Moving Average (ARIMA) model, where the data is initially differenced a number of times until the process

becomes stationary. It can be further extended with seasonal or exogenous terms, and is used in time series forecasting [24].

In [34], Connor et al. have shown feed-forward neural networks to be a special case of non-linear AR models, and RNNs to be a special case of non-linear ARMA models. They also introduced a robust filtering algorithm to be used in training to reduce the effect of outliers in the data and reported RNNs to achieve superior performance in comparison to feed-forward architectures in load forecasting. Moreover, a finite-sized RNN can simulate a Turing machine [35], so they can be used to model a wide range of problems involving time series.

Long Short-term Memory networks were introduced by Hochreiter and Schmidhuber in [36] to alleviate the problem of vanishing gradients that Recurrent Neural Networks have. They introduced novel components to the model, namely memory cells and gate units. Together they achieve constant error flow through time and a memory that is protected from irrelevant information. Both RNNs and LSTMs can be used to model a wide array of tasks [37]. They can have varying amounts of inputs and outputs depending on the task at hand. These differences are visualised in Figure 7. Inputs shown in red are fed to the recurrent model depicted in green. As time steps advance horizontally, model memory is passed on and updated. Outputs are visualised with blue. For example, when predicting the next time step based on many previous steps, a many-to-one scheme should be used. If instead several time steps are to be predicted, a many-to-many approach could be taken. Alternatively a many-to-one training with a multidimensional output could be used.

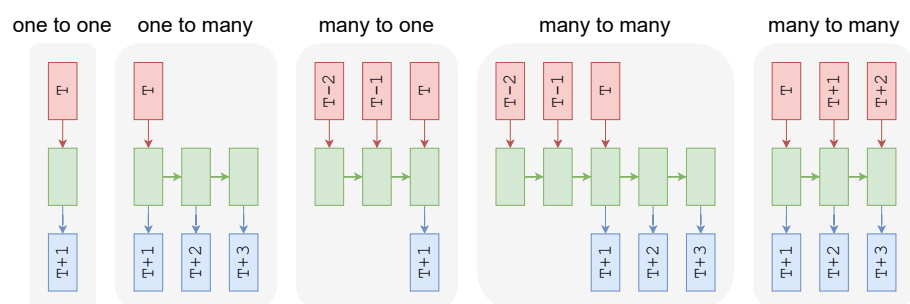


Figure 7. Different Recurrent Neural Network schemes, adapted from [37].

More recently in [38], Dabrowski et al. introduced ForecastNet, a time-variant model for time series forecasting. As opposed to the time-invariant RNNs and LSTMs models which share parameters for all time steps, ForecastNet utilises interleaved outputs which means a separate output exists for each point in the forecast horizon, as visualised in Figure 8.

Therefore, it is a sequential – though not recurrent – feed-forward architecture, and the whole forecast history is fed in at every step. Shortcut connections are made from the input layer to each hidden cell, which increases the network’s size and memory consumption, but according to the authors, leads to better representation across different time scales. It was compared to several state-of-the-art architectures and consistently outperformed them.

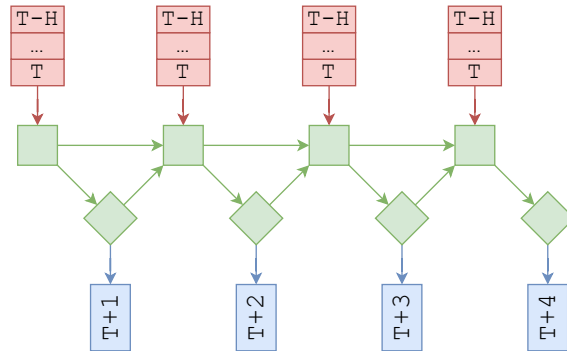


Figure 8. ForecastNet architecture, adapted from [38].

3.6 Evaluation of time series forecasting

Different ways of evaluating forecast results were reviewed by Hyndman and Koehler in [39]. There are two prototype metrics, namely Mean Absolute Error (MAE) and Mean Squared Error (MSE), on which many other metrics are based. MAE is arguably more interpretable of the two, representing simply the average deviation of a forecast from true known values. However, MSE penalises large errors more than small ones, which is often desirable. Using Root Mean Squared Error (RMSE) is common too. Taking the square root of MSE scales it to be on the same scale as the data. MAE is more suitable when describing uniformly distributed errors, while using RMSE is appropriate when they follow a zero-mean normal distribution [40]. While the conventional metrics are useful in evaluating the performance of different models on a single data set, their scale-dependence makes them unusable for comparing different data sets.

Achieving scale-independence is generally done by normalising a scale-dependent metric in some manner. Hyndman and Koehler introduced Mean Absolute Scaled Error (MASE) based on scaling errors with the average of differences between previous time steps to combat scaling and numerical issues with other methods [39]. They proposed that MASE should be the new standard. This was confirmed and further analysed by Franses [41], who

noted that MASE also shares attractive statistical properties with absolute and squared error metrics like MAE and RMSE. However, MASE cannot be computed when the forecasting history is constant. Hyndman and Koehler dismiss this case as irrelevant, but the data set used in this thesis contains many such cases (more in Section 4). For example, forecasts of a simple square wave whose wavelength is twice the length of the forecast horizon would be easy to predict, but impossible to evaluate with MASE. In the same vein, Kim and Kim proposed Mean Arctangent Absolute Percentage Error (MAAPE) in [42] as a means of dealing with normalisation issues. MAAPE measures relative performance with an inverse tangent, limiting evaluations between zero and $\pi/2$ instead of possibly infinite or undefined values.

4 Electricity consumption data

4.1 Time series and background data

Electricity consumption data provided by a electric power company in Finland was available for this thesis project. The data set contains hourly recordings of 3445 locations during one year. Some series also contained background data, the types of which are detailed in Table 2. Figure 9 contains the consumption series of two example consumers.

Table 2. Number of available background labels.

Label	Count
Construction year	972
Residence type	972
Resident count	972
Residence size	972
Heating type	969
Heating method	967
Heated size	556
Additional heating	539
Water heating	145
Wood consumption	668
Ventilation	965
Sauna	819
Sauna usage	880

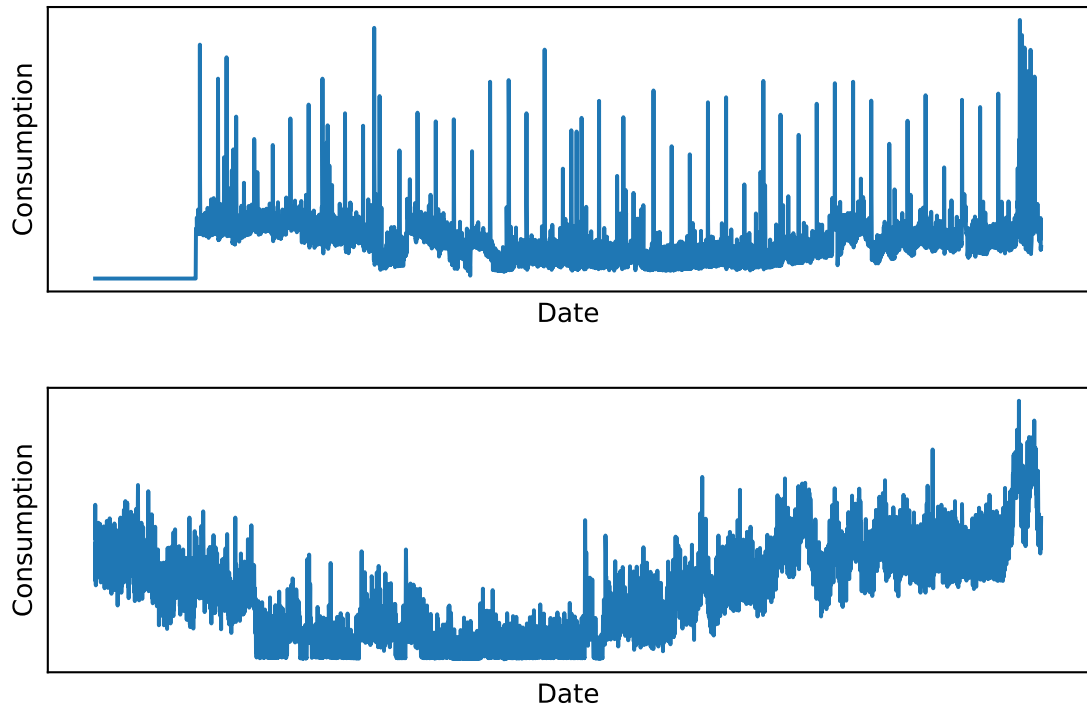


Figure 9. Example series.

4.2 Data quality and preprocessing

The data was unprocessed and contained many problematic series. Figure 10 depicts data quality by the number of missing, negative and zero values. The index is sorted by the number of these defects. Less than a thousand series, or approximately 25 % of the total contain none of these errors.

Missing and negative values are naturally considered as errors in the data. Whether zeros should be valid or not is unclear. Some of the series contain sudden drops to zero for very few observations, after which an ordinary pattern continues. This is likely an error and should be corrected for. Reliably identifying these instances, however, is not trivial. It is entirely possible for a series to contain mostly zeros. Secondary residences – or more commonly in Finland, summer cottages – can be vacant for most of the year with no electricity consumption. Thus the length of a zero-consumption period might indicate whether it is valid. Short but frequent zero values should also be considered valid, because locations that have low consumption might alternate between the lowest measurable values and zero. Figure 12 visualises the total amount of zeros with the longest consecutive string of zeros. Brighter yellow indicates more instances, while dark purple corresponds to none. Point (1, 1) on the graph was removed to better observe the image,

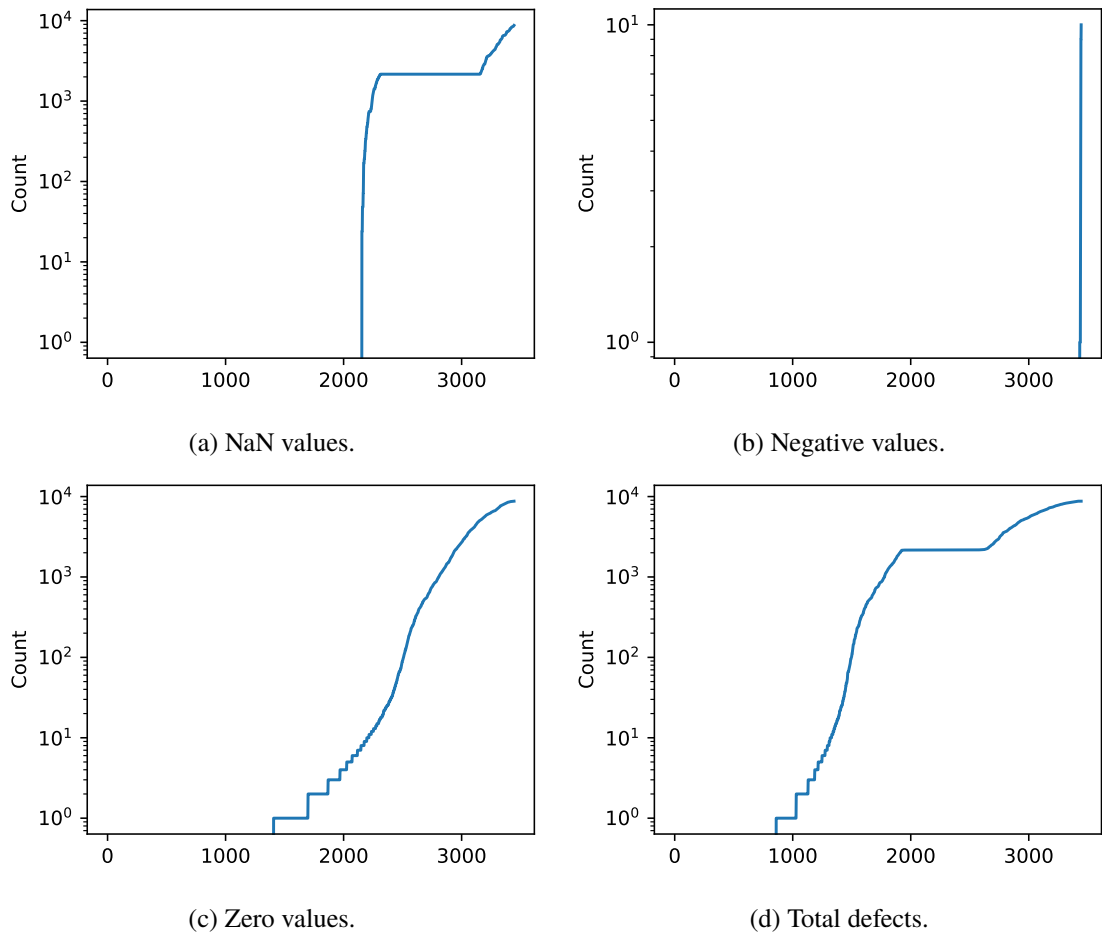


Figure 10. Data quality per customer.

because its value dominated others. Having frequent but separate zeros is not common, as most of the series lay close to the diagonal.

These defects were removed from the data set before proceeding with further investigations as follows. All problematic cases defined above were marked as missing data and linearly interpolated for at most 4 observations and 99 in total. Any series that had more consecutive or total zeros, or other defects were discarded, leaving 1723 clean series, or approximately 50 % of the total to be used. The limits were largely arbitrary, designed to strike a balance between being able to use most series that contain a few defects while excluding more problematic cases.

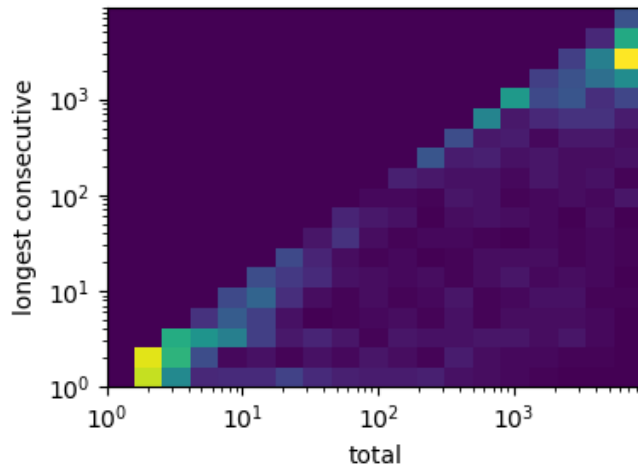


Figure 12. Zeros in data.

4.3 Forecast length and data splits

It would be useful if the methods used in this thesis could be applied to new customers without a long delay for enough data to be acquired. For this reason we assume a forecast history of one week. It makes it possible to generate predictions for new customers rather quickly while still providing data from each week day for forecasting. The same history is also used for reference models which could require less data, because the same exact data set should be considered when evaluating models.

A short forecast history also allows for generating several training examples from a single consumer. Seven-day histories are generated for each full day used as the forecasting target. It would be possible to generate histories from each hour forward, but that would multiply the amount of data to impractical levels. With clean data the number of 192-dimensional samples would increase from 770 000 to 18.5 million, although most of the information would be redundant. Additionally, it could decrease the effectiveness of models as they would not have information about which hour of day the data begins. This hypothetically adverse effect could be alleviated by feeding in additional information about the daily or weekly cycle, but it is beyond the scope of this thesis.

Clustering of consumers should also utilise the aforementioned short histories for reasons related to both practical applications and avoiding leaking information to forecasting models. If the groups would be determined based on the whole series of a consumer the mere assignment to a group could skew the results of early forecasts. This also has the added benefit of naturally addressing problems related to possibly changing consumption patterns.

Lastly, to provide appropriate evaluations, the data is split into training and testing data by consumer before splitting each series into individual samples for clustering and forecasting. Testing data is used only in evaluating final results. All this allows for fair apples-to-apples comparisons with practical applications in mind.

4.4 Data exploration with PCA

It is not obvious whether there are distinct groups that consumers or short forecast histories fall into or not, as applying most clustering methods will produce results regardless of their quality. The data set should be tentatively explored to find out if further cluster analysis could be fruitful. This exploration may also provide a basis for choosing a suitable set of methods for the actual clustering.

Principal Component Analysis (PCA) is a staple of dimensionality reduction [43], and it can be used in data visualisation and as a noise reduction technique to provide other methods with a more condensed representation of data. PCA represents data with an orthogonal basis, whose components represent the directions of maximal variance left in the data after subtracting previous components. By dropping the components which contribute less to the overall variance its effective dimensionality can be reduced.

As can be seen in Figure 13, the variance explained by different components quickly decreases after the most significant principal component. However, when reconstructing series using fewer components, the reconstruction error decreases approximately linearly when adding further components after the first few (Fig. 14). This suggests that while the first components capture most of the variation, the data is not easily reducible to a small subset of principal components.

Figure 15 visualises some principal components graphed as unit vectors. Examining the most significant ones reveals that they represent general weekly tendencies, either daily variations or week-long cycles. By contrast, the less significant components are harder to interpret and get progressively more chaotic. The last components seem to take the form of short pulses. Notably, all dimensions of the most significant component are positive, unlike other components, intuitively indicating that it describes the overall consumption level of most series effectively.

This analysis alone suggests that the series are regular and similar enough to warrant further examination. Using PCA to reduce the 168-dimensional vectors to two of their most

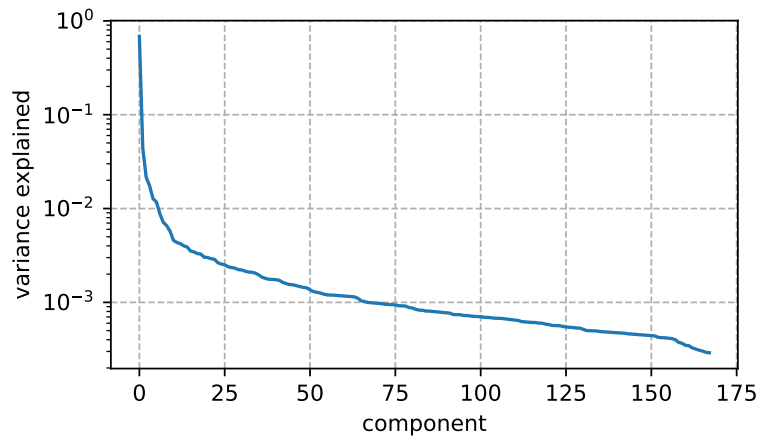


Figure 13. Principal Component Analysis explained variance.

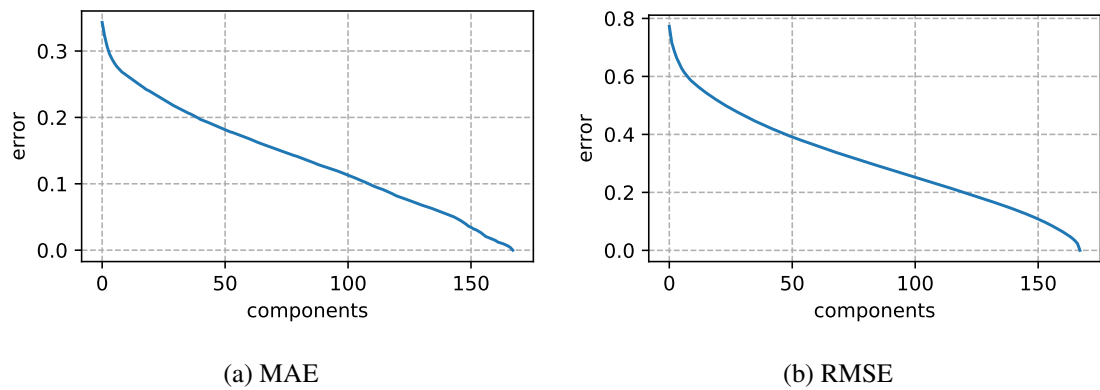


Figure 14. Principal Component Analysis reconstruction error when increasing the number of components.

significant components, and examining how consumers behave in that space across the year, should further indicate whether consumer group information can easily be extracted from the samples or not.

It is intuitively true that overlapping samples sliced from a time series should be similar by any method that extracts meaningful information. That is, if the series themselves also contain such information. Therefore, if we were to represent all series of a single consumer with some very low-dimensional vector, depending on the representation technique consumers would act in seemingly chaotic or ordered ways. If a method was successful in reducing dimensionality meaningfully, either all representations of a consumer's series would be very close to one another, or if the series change over time, a path between the points representing these different modes should be observed. This must be true of dimensionality reduction techniques and clustering methods alike. Conversely, if no such information can be extracted, representations of the series should be located all over the

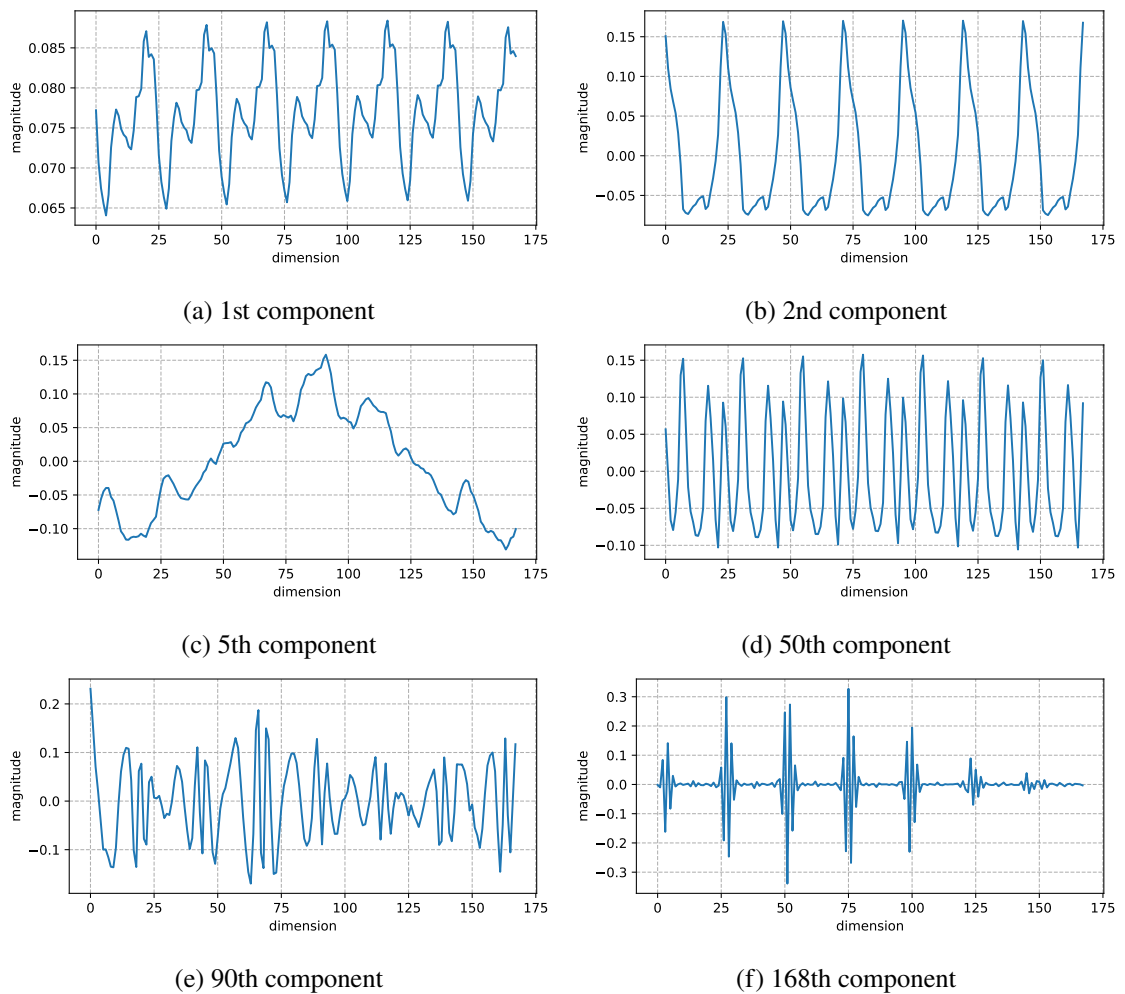


Figure 15. Examples of Principal Component Analysis unit components.

space of possible values.

Figure 16 depicts example consumers in the reduced PCA space. Each point is a single sample sliced from a consumer's series. Histories starting from each day are colored with a gradient from cyan corresponding to winter, from green during spring, red in summer and purple in fall back to cyan. Even though the examples are hand-picked to represent different kinds of consumers, it is obvious that samples *can* exhibit meaningful patterns. They also align with intuitions derived from the first component in Figure 15. Samples that correspond to cold months seem to be located further right in the figures, which can be interpreted as increased overall consumption. Given these initial results, there is potential in clustering analysis with this particular data set. However, no clearly separated clusters can be seen in the data based on PCA. Instead, the data appears to form a single group with a dense center and some outliers.

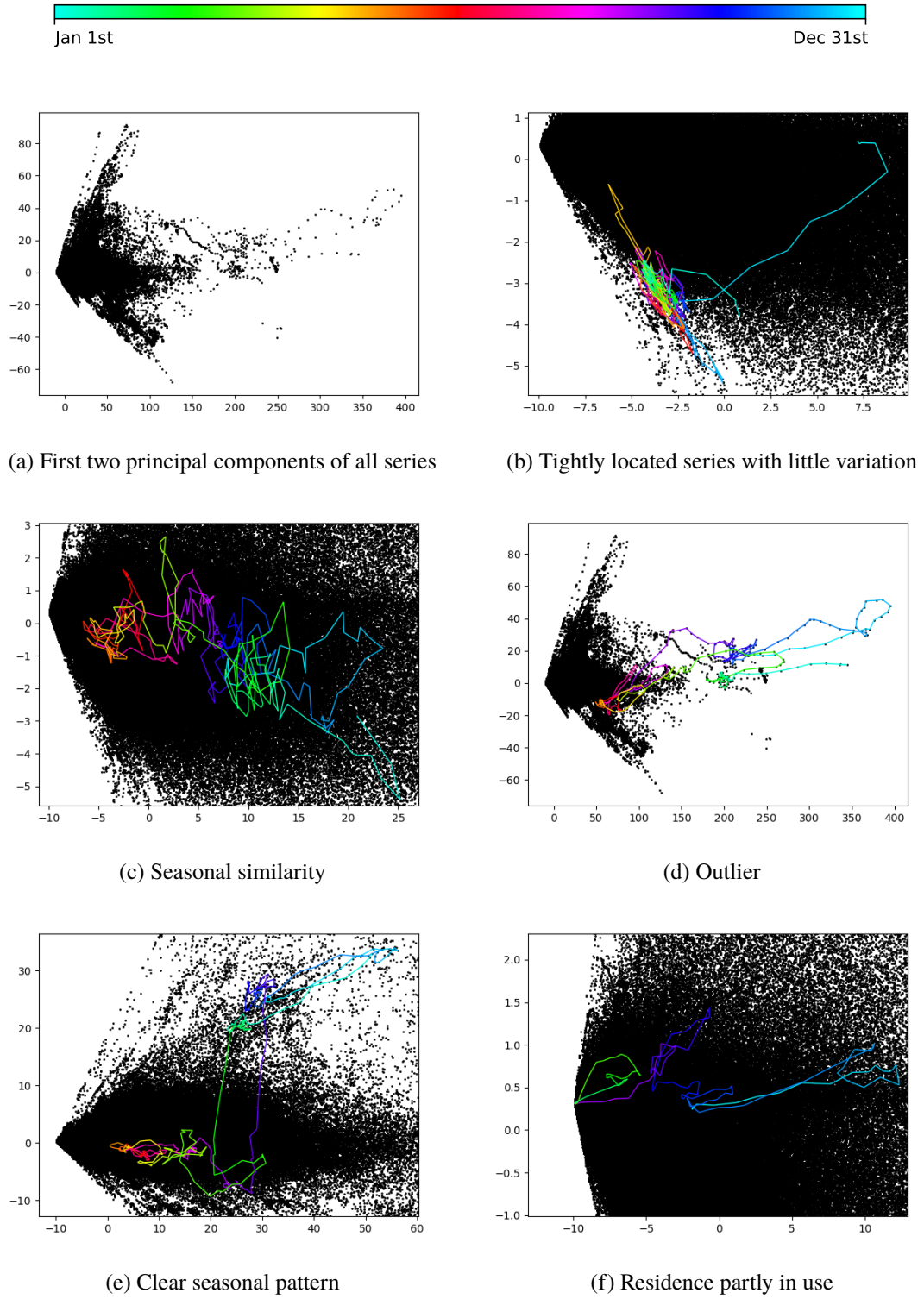


Figure 16. Examples of consumers in Principal Component Analysis space.

5 Proposed methods

5.1 Clustering

Based on [8] and given the results of DTC in [29], the feasibility of joint unsupervised feature learning and clustering is examined in this study. An autoencoder is trained, and the latent representation of series is used as features for a clustering algorithm. This process is repeated varying the network size and the number of clusters produced. Then each result is evaluated with metrics presented below and the appropriate number of clusters determined.

5.1.1 Autoencoder

Two types of autoencoders are used, one with and another without joint optimisation for clustering. They share the same shallow densely-connected architecture, which consists of two layers, one for encoding and another for decoding. The data flows through the two layers as follows:

$$H = \alpha(XW_1 + b_1) \quad (1)$$

$$\hat{X} = HW_2 + b_2, \quad (2)$$

where W and b are the weights and biases of each layer, α is the activation function of the first layer, resulting in H as the latent representation of data after encoding and \hat{X} as the reconstruction after decoding. The network is optimised using MSE as the loss function:

$$\text{MSE} = \frac{1}{D} \sum_{i=1}^D (X_i - \hat{X}_i)^2. \quad (3)$$

Joint optimisation is performed by including another loss to the learning scheme, as done in [8]. Cluster hardening loss utilises the Kullback-Leibler divergence of cluster assignment probabilities Q from a target distribution to enforce harder assignments. Student's t -distribution is used to measure distances from points to centroids:

$$q_{ij} = \frac{(1 + \|z_i - \mu_j\|^2/\nu)^{-\frac{\nu+1}{2}}}{\sum_{j'} (1 + \|z_i - \mu_{j'}\|^2/\nu)^{-\frac{\nu+1}{2}}}, \quad (4)$$

where z_i are encoded features, μ_j cluster centroids in the encoding space, and ν is a constant. The target distribution P is constructed by making these assignment probabilities stricter:

$$p_{ij} = \frac{q_{ij}^2 / \sum_i q_{ij}}{\sum_{j'} (q_{ij'}^2 / \sum_i q_{ij'})}. \quad (5)$$

Finally, the KL-divergence is calculated and included in the total loss of the network L with a weight γ :

$$\text{KL}(P||Q) = \sum_{i,j} p_{ij} \log \frac{p_{ij}}{q_{ij}} \quad (6)$$

$$L = \text{MSE} + \gamma \cdot \text{KL}. \quad (7)$$

5.1.2 k -means

Data exploration alone was not able to reveal any distinct groups in the data. However, an important criterion for choosing a clustering method arises from data usage considerations. The algorithm must be capable of labelling novel samples to make evaluation with testing data possible.

k -means is a clustering algorithm that represents groups of samples as a set of centroids [44]. Each sample belongs to the closest centroid. It is iteratively trained to minimise the deviation of samples from centroids that they belong to. k -means is able to handle data which is not in distinct groups and novel samples can easily be assigned to clusters. It is also highly interpretable, making it suitable for this task.

Given a set of K cluster centers M_k that represent members X_i with their mean μ_k , k -means minimises the distance from each center to samples that have been assigned to that center [44] as follows:

$$J(C) = \sum_{k=1}^K \sum_{X_i \in M_k} \|X_i - \mu_k\|^2. \quad (8)$$

Naturally, any distance measure can be used in place of the common L_2 norm. Samples belong to the closest cluster.

5.1.3 Evaluation of clustering

Some series in the data set are labelled with background information. They cover various aspects of a site, such as its size, heating and ventilation methods. However, almost none of the series have all these values, nor is it self-evident that any of them should be considered to generate meaningful groups. In the absence of a single, reliable label to evaluate a clustering result with, internal metrics need to be considered. Therefore, both the latent dimensionality and the appropriate number of clusters are chosen by examining losses as well as Silhouette and Davies-Bouldin indices.

The Davies-Bouldin index $D(c)$ of a cluster is a measure of spread relative to the distances of cluster centroids [32]:

$$s(c) = \left(\frac{1}{|C_c|} \sum_{i \in C_c} |X_i - A_c|^p \right)^{1/p} \quad (9)$$

$$m(c_1, c_2) = \|A_{c_2} - A_{c_1}\|^p \quad (10)$$

$$D(c) = \max_{k \neq c} \frac{s(c) + s(k)}{m(c, k)} \quad (11)$$

where C_c are sets of samples in cluster c with a centroid A_c , and X are samples. A lower score is better, and it means that a cluster is compact while also being distant from other clusters. The Davies-Bouldin index for all clusters is the mean value of D .

The Silhouette index $S(i)$ of a sample i is similar to Davies-Bouldin [31]. It measures how well a sample is assigned to a cluster while being separated from the nearest cluster it is not a part of:

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i} d(X_i, X_j) \quad (12)$$

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(X_i, X_j) \quad (13)$$

$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \quad (14)$$

where C_i are sets of samples that are in the cluster sample X_i belongs to. $d(x_1, x_2)$ is a distance metric. In the edge case where $|C_i| = 1$, $S(i) = 0$. $S(i)$ is also constrained to the interval $-1 \leq s(i) \leq 1$. A higher score is better. These scores can be averaged to represent the quality of the whole clustering result with one scalar value.

5.2 Load forecasting

5.2.1 ForecastNet

Given the state-of-the-art performance of ForecastNet in [38], it is used as the main forecasting model for this thesis. As illustrated previously in Figure 8 in Chapter 3.5, ForecastNet's input is exposed to each hidden cell, which are further connected to outputs and subsequent hidden cells. Finally, the output layers themselves are connected to the following hidden cells as well. The cells need not be ordinary Multi-Layer Perceptrons (MLPs), but they can be implemented as CNNs or any other architecture which may also vary across cells. The authors proposed that the output layers of ForecastNet could model a Gaussian distribution. Instead of returning single values, they would return the parameters of a normal distribution – mean and standard deviation – to estimate the uncertainty of forecasts. Four variations of ForecastNet were considered, varying the choice of densely-connected or convolutional layers in hidden cells, and outputs consisting of Gaussian mixture density or linear layers.

Originally, Dabrowski et al. fed known prediction targets to subsequent hidden layers instead of model outputs [38]. This *teacher forcing* [45] may lead to faster initial convergence and prevent instability arising from basing predictions on other predicted values, but it also changes the learning objective, as error signals are not propagated from hidden layers to previous outputs and the network is trained with true values instead of its predictions. With this in mind, if stability problems are encountered during training, teacher forcing is considered as a remedy. Otherwise, it is not used.

5.2.2 Baseline models

A set of baseline models is beneficial for evaluating the performance of more complex prediction models. They allow for considering if performance is improved enough to warrant the added complexity. Four basic models are used, each slightly more sophisticated than the last. Firstly, a constant consumption of zero is predicted. Then, predictions are made using the last known value, repeating it indefinitely. Thirdly, a similar static prediction is made using the average consumption of the last 24 hours. Finally, the individual hourly values of the previous day are used as a prediction for the next day.

ARIMA models are used as the most advanced baseline [11, 33]. Its parameters are

denoted as $ARIMA(p, d, q)$, where p is the number of autoregressive components, d is the order of differencing and q is the number of moving average components. Because the models are used as a baseline and there is no reason to assume that electricity consumption follows any well-defined statistical process, parameters are selected by choosing the best-performing combination. A separate model is created for each series, and its parameters are set as follows. First the appropriate order of differencing d is selected by performing Kwiatkowski-Phillips-Schmidt-Shin stationarity tests [46]. Then p and q are varied from 0 to 3 choosing the model that minimises the Akaike Information Criterion [47].

5.2.3 Evaluation of forecasting

To effectively assess the performance of predictive models MAE, RMSE and MAAPE are used as error metrics. Additionally, the forecasts should be evaluated in terms of usability in load forecasting. Predictions on any given hour can be aggregated across consumers to provide an estimate for the total load on the network, after which MASE can be applied too.

Given a true consumption Y_t and a prediction \hat{Y}_t for a forecasting horizon $t \in [1..T]$, MAE and RMSE are calculated as follows [39]:

$$e_t = Y_t - \hat{Y}_t \quad (15)$$

$$MAE = \frac{1}{T} \sum_{t=1}^T |e_t| \quad (16)$$

$$RMSE = \sqrt{\frac{1}{T} \sum_{t=1}^T e_t^2}. \quad (17)$$

To avoid extreme values, MAAPE uses an inverse tangent function to scale the relative error [42]:

$$MAAPE = \frac{1}{T} \sum_{t=1}^T \arctan \left(\left| \frac{y_t - \hat{y}_t}{y_t} \right| \right). \quad (18)$$

The inverse tangent function would produce values in the range of $[0, \pi/2]$. In order to make the metric more interpretable as a percentage error, it is normalised to have values between $[0, 1]$ and converted to percentages. Wherever undefined values that arise from dividing zero errors by true values of zero would occur, they are replaced with a zero.

MASE incorporates differences between historical time steps to normalise error values. Given an observation history of length H ($t \in [-H + 1..0]$), MASE is calculated as follows [39]:

$$q_t = \frac{e_t}{\frac{1}{H} \sum_{h=-H+1}^0 |Y_h - Y_{h-1}|} \quad (19)$$

$$\text{MASE} = \frac{1}{T} \sum_{t=1}^T |q_t|. \quad (20)$$

The observation history is the same for all points in a forecast horizon.

5.3 Neural network hyperoptimisation

All the architectures presented above have tuneable hyperparameters, namely the choice of architecture, learning rate, mini-batch size and possible use of weight decay and dropout. Recently in [48], Smith details best practices for hyperparameter optimisation, also known as hyperoptimisation. Selecting a high-enough learning rate with the right configuration of other hyperparameters is expected to lead to quicker convergence.

Mini-batch learning is used with the batch size initially set to the maximal value constrained by memory [48]. The terms *mini-batch* and *batch* are used interchangeably throughout the rest of this thesis. Smith noted that if the whole data set could fit in memory at once during training, other batch sizes could be considered. Mainly large convolutional architectures, whose memory consumption is significantly higher by virtue of input dimensionality, were considered. Therefore, increasing the number of batches in an epoch may indeed be appropriate, because it introduces more variation during training, which in turn may lead to a better optimum.

A learning rate range test [48] is performed before training to determine a suitable learning rate. The test is performed by training various network configurations with an increasing learning rate, and recording performance. As the learning rate becomes too large, training will destabilise. The highest viable learning rate is found before that point. A cyclical learning rate schedule oscillating between the highest possible value and a lower one is used.

Other forms of regularisation are not considered unless a need to avoid overfitting arises. The data set is rather large, which will already provide some regularisation if smaller architectures are preferred.

ReLU is chosen to be the activation function across all architectures for its ease of computation and resistance to gradient vanishing. ReLU is defined as follows:

$$\text{ReLU}(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0. \end{cases} \quad (21)$$

6 Experiments

6.1 Implementation

As stated in Chapter 1, the following experiments aim to build a set of models to forecast electricity consumption, and evaluate the effectiveness of clustering the data before forecasting. A model is trained for each cluster separately.

All experiments in this thesis were conducted with an environment based on the open-source programming language Python. It is widely used in the scientific community largely thanks to popular third party packages that enable efficient mathematical operations. Table 3 contains a list of those packages.

Table 3. Python libraries used.

Name	Description
numpy	efficient numerical operations
pandas	database-like indexed data structure
scipy	scientific computing
sklearn	data analysis and machine learning algorithms
tensorflow	GPU-computing & neural networks
h5py	data serialisation
pmdarima	ARIMA model
matplotlib	plotting figures
seaborn	data visualisation

6.2 Data division and normalisation

Before other experiments, a validation set with a similar size to the test set was extracted from training data. Split sizes were chosen largely arbitrarily to leave plenty of data for training while providing enough for testing and validation to reliably assess performance. There was no shortage of data, so cross-validation is not considered. The splits are detailed in Table 4.

Normalisation is desirable for neural networks to effectively learn to represent series at different scales. There are two options for normalisation: using a single factor for the whole data set or calculating one for each series. The latter would allow a model to more easily

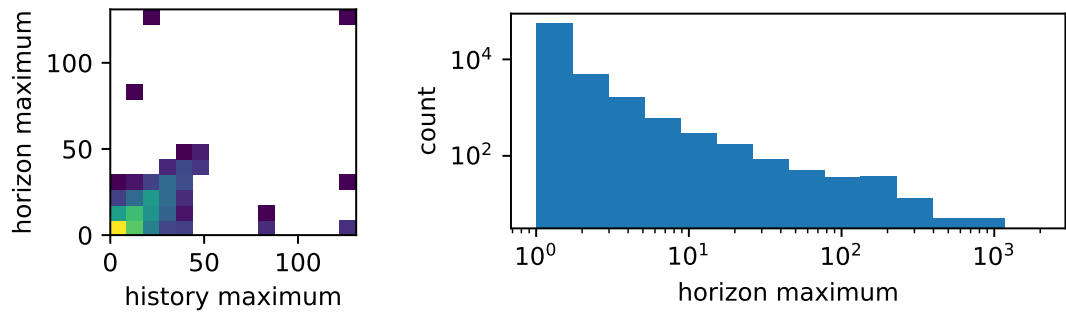
Table 4. Data splits.

Split	Share	Count
Training	0.6	462 625
Validation	0.2	154 209
Test	0.2	154 298

handle series that are similar in form but not in scale, but it introduces a challenge with long sequences of zero or near-zero values. One must avoid leaking information about the prediction horizon, which naturally is not available when predicting truly unknown values. Therefore, if each sample was normalised separately, both the history and horizon should be scaled with a value calculated from the history alone. Series with small historical values paired with large prediction targets, which inevitably exist in this data set, will dominate the evaluation of a network. At the very extreme, a constant zero history cannot be normalised at all.

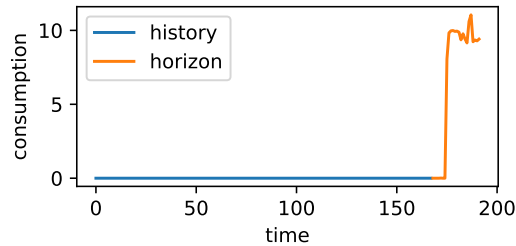
As can be seen in Figure 17, the scale of most series is quite contained, but for some the historical scale varies greatly from the horizon, as after normalising with each history some horizons are orders of magnitude larger than the average. Series whose maximum was smaller than 1 are not shown in 17b. Therefore, in further experiments the samples are normalised using a single value calculated from the data set. The 99th percentile value found in any of the training histories is used to reduce the impact of the largest consumers on the operation. Using it amounted to a 20-fold decrease in the normalisation factor from 130 to 6.3.

In principle, normalisation in clustering could be performed for each sample separately, but it is apparent that overall scale is an important factor in the data. Therefore, samples are normalised using the same single factor prior to clustering experiments.



(a) Maxima in histories and horizons.

(b) Horizon maxima after normalising with history.



(c) Example series with problematic scale.

Figure 17. Self-normalisation of the training set.

6.3 Clustering with shallow autoencoder

Using a shallow autoencoder for clustering, there are three hyperparameters to set: latent dimensionality, learning rate and batch size. After training a network, the appropriate number of clusters should be selected. It is noteworthy that both the reconstruction and cluster evaluation metrics tend to produce better results when the latent dimensionality or number of clusters is increased. Therefore, modest configurations that achieve significant gains in comparison to even smaller setups should be preferred over strictly better performance.

A learning rate range test was performed for various latent dimensionalities across varying batch sizes to determine the appropriate learning rates to use (Fig. 18). Figures presented in this Section are for the case of 8 batches. See Appendix 1 for a complete set of figures. Each network was trained for three epochs per learning rate with a logarithmically increasing schedule. Increasing the number of batches in each epoch seems to make very small learning rates more viable, but that is likely due to the number of parameter updates compared to cases with fewer batches. The upper limit for the learning rate was similar across batch sizes.

Setting the cyclical learning rate to a suitable range for each latent dimensionality, net-

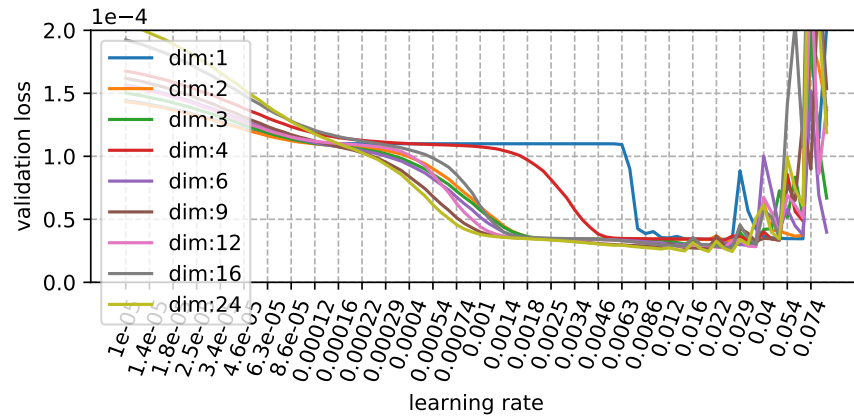


Figure 18. Learning rate range test for different network sizes.

works were trained until convergence. In practice training could have been interrupted when the loss stopped improving significantly, but for the sake of visualisations across dimensionalities and batch sizes, each network was trained for 100 epochs. Figure 19 shows the validation loss for different latent dimensionalities. It should be noted that increasing the number of batches leads to a lower loss for larger networks more easily, while affecting smaller ones less.

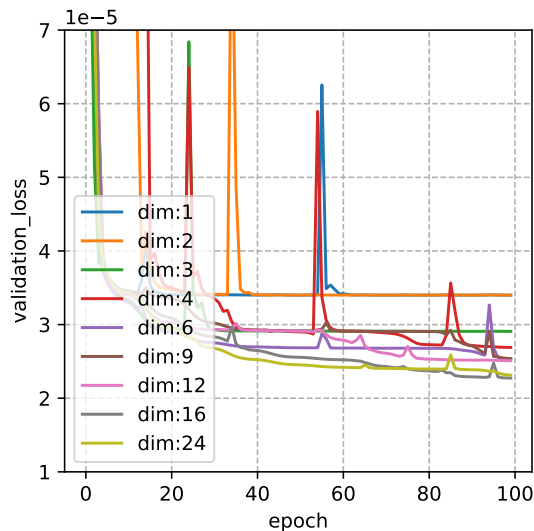


Figure 19. Validation losses for different network sizes.

As hypothesised, a similar situation of diminishing returns arose with clustering evaluation metrics as did with latent dimensionality (Fig. 20). It is clear that one- and two-dimensional latent spaces are poor choices, which is further strengthened by examining the results with other batch sizes in Figure A1.3. Using three-dimensions seems to be a

good balance of performance and simplicity together with choosing the number of clusters to be six.

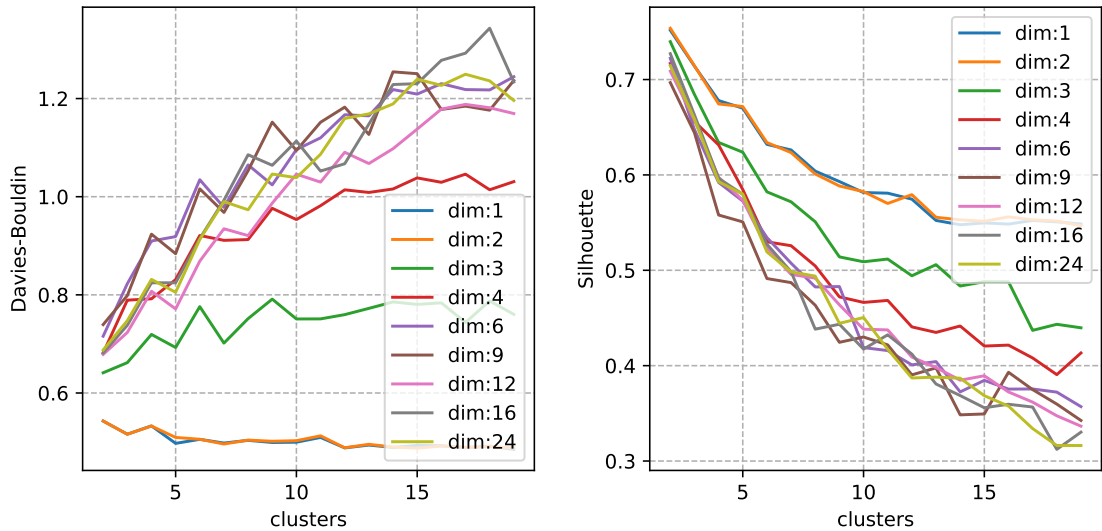


Figure 20. Metrics for different network sizes.

The rough appropriate number of batches was then selected by training the network with different batch sizes. As seen in Figure 21, performance varied across runs, but dividing the dataset into 64 batches had the potential to achieve the best results. The loss had a few modes rather than a continuously reducing value, suggesting that the data contains series which share common characteristics or a few samples whose loss dominates the learning objective.

Examining the single training result in Figure 22, it is clear that overfitting did not occur. The validation subset was easier, possibly containing less problematic cases. The groups produced by k -means were quite imbalanced. Interestingly, the PCA representation of the produced latent space resembled the PCA space shown in Figure 16 in Chapter 4.4, indicating that the network learned relevant information, but also that it would have been readily available with simpler methods.

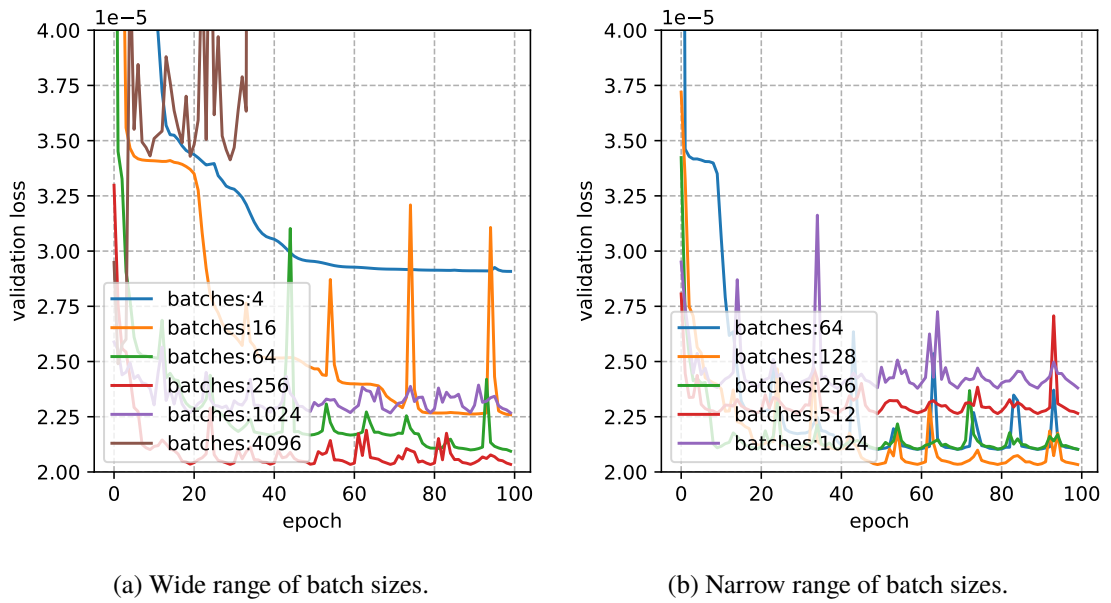
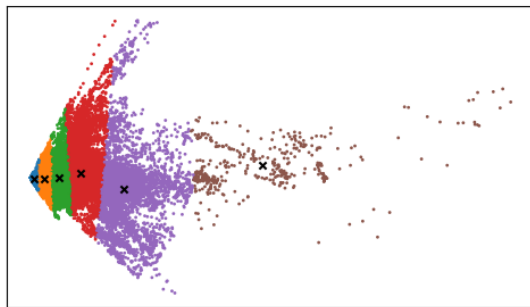
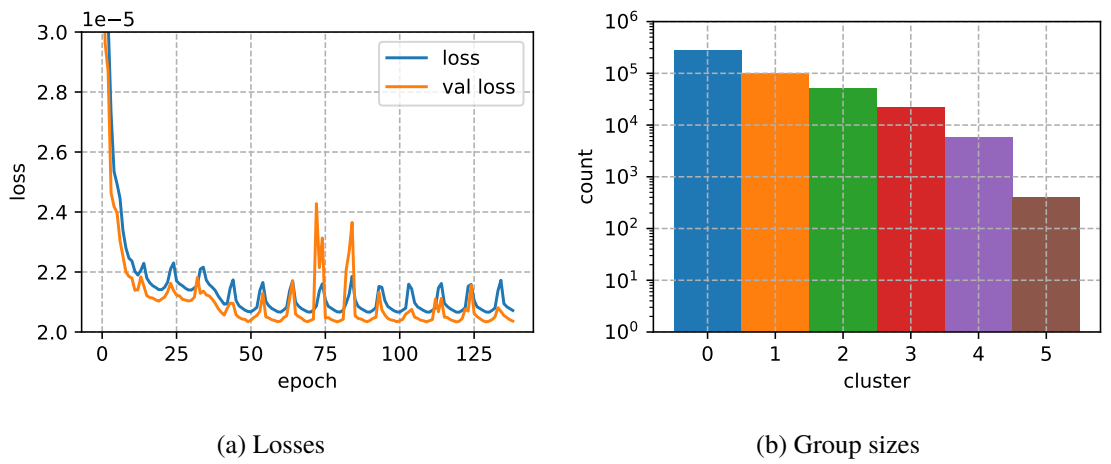
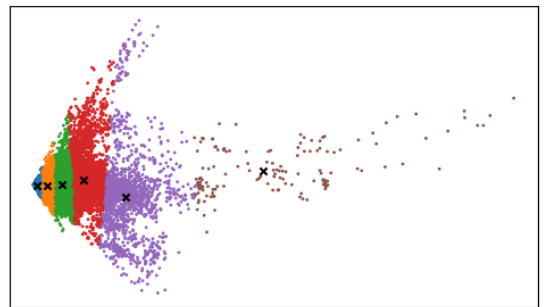


Figure 21. Batch sizes.



(c) PCA of latent training data.



(d) PCA of latent validation data.

Figure 22. 3-dimensional encoding with 64 batches.

6.4 Unclustered forecasting

A similar methodology of selecting hyperparameters as described in the previous experiment was employed for forecasting. First, appropriate learning rates were selected for a number of different batch and network sizes using a learning rate range test. Its results can be found in Appendix 2. The figures show behavior similar to clustering hyperoptimisation.

After selecting suitable learning rate ranges, the configurations were trained. Their validation losses are shown in Figure 23. It seems that returns from increasing the network size diminish very fast, as the largest architectures achieved tiny improvements over modest configurations. Because of this, a hidden dimensionality of 16 nodes was selected. Choosing a smaller architecture had another benefit too. It reduced training times significantly, firstly and obviously because of the reduced size of matrix multiplications in the network, and secondly because larger batch sizes can potentially be used, as they fit into memory more easily during training. Whether a similar result could be achieved using the same number of model updates regardless of batch size was not considered.

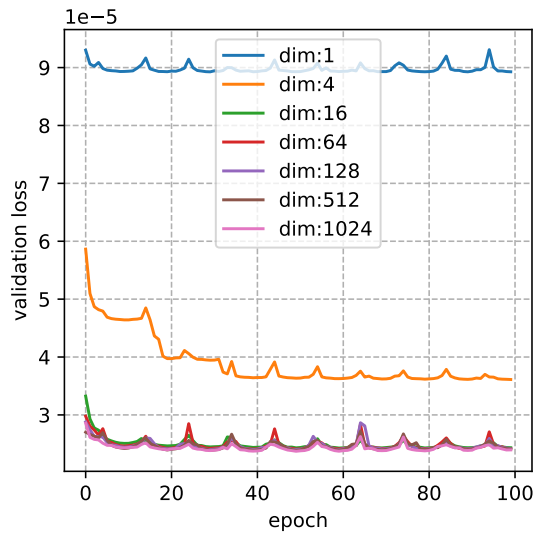


Figure 23. Validation losses with 64 batches.

Batch size tests for the selected architecture were conducted next. Their results are presented in Figure 24. A large range of batch sizes performed equally well, both converging quickly and being stable across a longer training.

Finally, a ForecastNet was trained using 128 batches. Losses during training are visualised

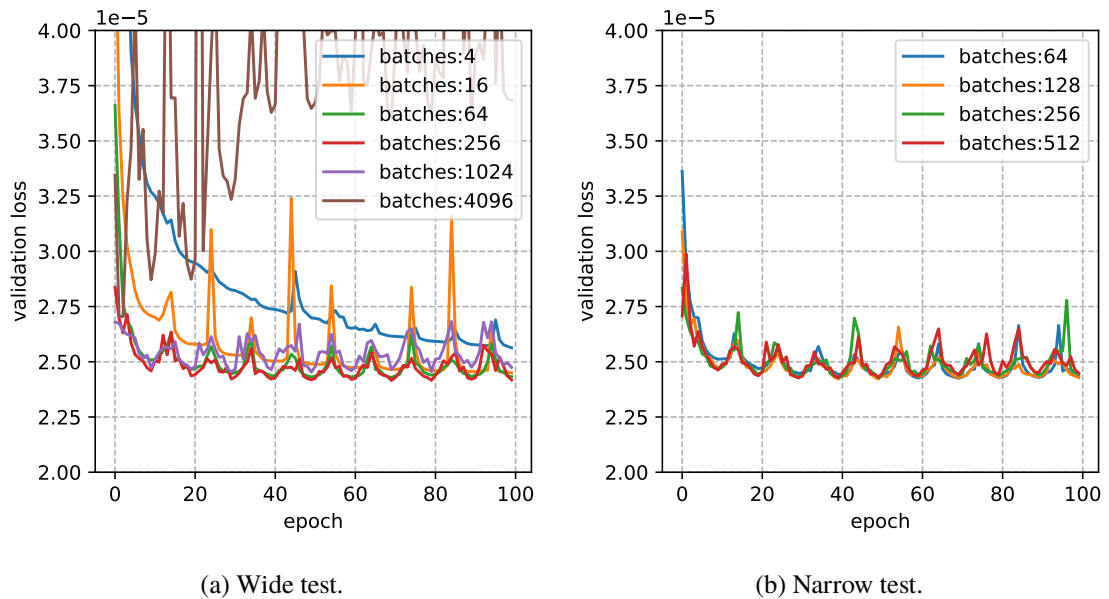


Figure 24. Validation losses of batch size tests.

in Figure 25. A good validation loss was achieved quickly. Although training loss kept decreasing, there was no sign of overfitting.

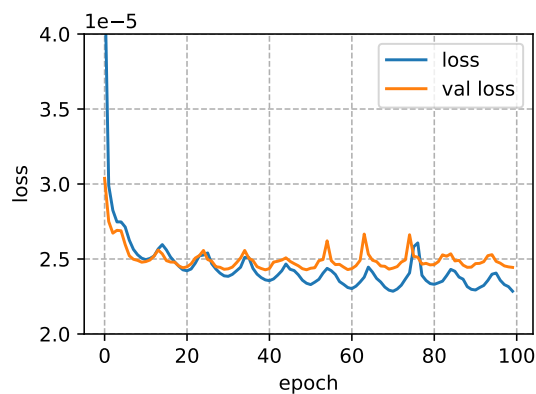


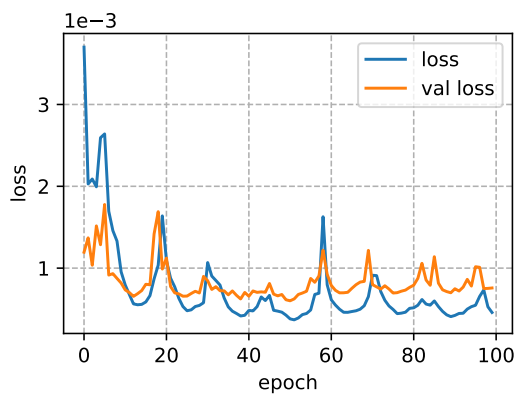
Figure 25. Losses of the selected ForecastNet.

6.5 Forecasting with clusters from dense autoencoder

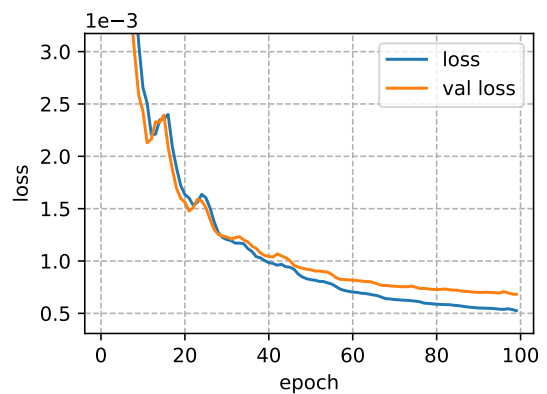
If the architecture that was selected for forecasting with the whole dataset cannot be improved, then improvements seen when training similar models for each cluster separately could indicate that such a division is beneficial for forecasting too.

However, there is reason to doubt that improvements would be made. Firstly, as Laurinec et al. noted in [16], deep networks benefit less from prior clustering when compared to shallow and statistical approaches, as they naturally are able to extract more information about the input. Secondly, the nature of the dataset in question, even after clustering, is such that no distinct groups were found.

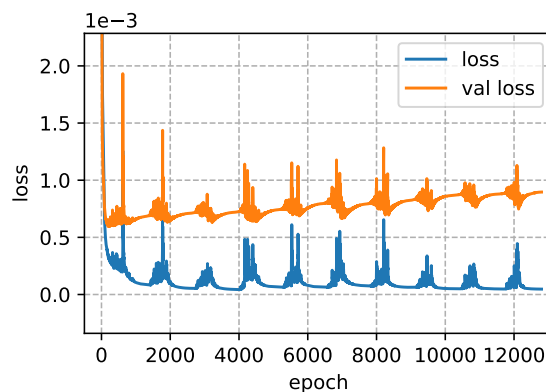
Training a similar network, but with a subset of the data required, a decision regarding batch sizes and the number of model updates. If the same batch size is used, the number of batches reduces dramatically for the smallest clusters. But if it is counterbalanced with increasing the number of epochs to have the same number of model updates, the network starts overfitting. Results of applying these approaches to the smallest cluster are shown in Figure 26 and for all clusters in Appendix 3. In order to avoid overfitting problems, a constant batch size was used while keeping the number of epochs at 100.



(a) Constant number of batches per epoch.



(b) Constant batch size.



(c) Constant batch size and number of model updates.

Figure 26. ForecastNets trained with the data belonging to the smallest cluster.

6.6 Clustering with joint autoencoding and cluster hardening

In order to have comparable results with the previous clustering, joint learning is performed with a similar model, but with the added clustering layer. It introduces a new hyperparameter, clustering loss weight. The appropriate number of batches per epoch and learning rates should also be selected again.

Parameters were selected as before, first performing a learning rate range test, then training networks with each combination in order to select a suitable set of parameters. As can be seen from Figure 27, validation losses that only include the reconstruction loss increase with the addition of clustering loss, which is expected because the learning objective changes.

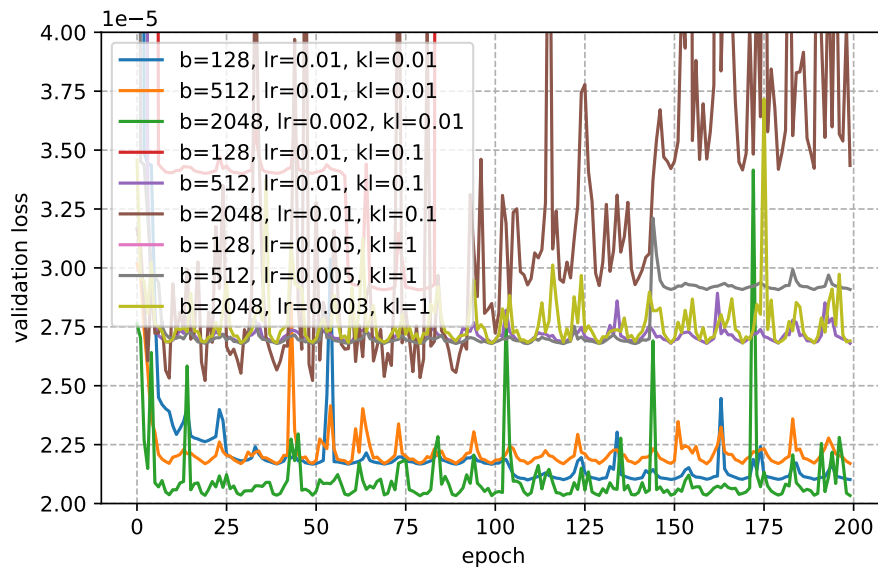


Figure 27. Validation reconstruction losses across different models.

However, training with the clustering loss led to unwanted results. Figure 28 visualises the encoded data with the learned cluster centers. All centers are extremely close to one another and outside the data. This prompted a more thorough examination of the clustering loss.

A minimal, synthetic data set was constructed to examine the loss and its components. It consists of three two-dimensional data points, $(0, 0)$, $(0, 1)$ and $(1, 0)$ and three cluster centers whose places are varied. Let us consider six different situations: centers on top of each data point, close to each data point, on top of each other on one data point, on top of



Figure 28. Unwanted clustering result.

each other in the middle of the data, on top of each other outside the data, and far from each other outside the data. For each situation the chosen centers are presented with the Q and P values that arise from cluster hardening, along with the final Kullback-Leibler divergence loss.

1. Centers on top of each data point: $(0, 0)$, $(0, 1)$ and $(1, 0)$.

$$Q = \begin{bmatrix} 0.5 & 0.25 & 0.25 \\ 0.27 & 0.55 & 0.18 \\ 0.27 & 0.18 & 0.55 \end{bmatrix}, \quad P = \begin{bmatrix} 0.65 & 0.17 & 0.17 \\ 0.17 & 0.74 & 0.08 \\ 0.17 & 0.08 & 0.74 \end{bmatrix}, \quad \text{KL} = 0.221$$

2. Centers close to each data point: $(0.2, 0.2)$, $(0.2, 0.8)$ and $(0.8, 0.2)$.

$$Q = \begin{bmatrix} 0.44 & 0.28 & 0.28 \\ 0.3 & 0.47 & 0.22 \\ 0.3 & 0.22 & 0.47 \end{bmatrix}, \quad P = \begin{bmatrix} 0.53 & 0.23 & 0.23 \\ 0.24 & 0.62 & 0.14 \\ 0.24 & 0.14 & 0.62 \end{bmatrix}, \quad \text{KL} = 0.111$$

3. Centers on top of each other on one data point: $(0, 0)$.

$$Q = \begin{bmatrix} 0.33 & 0.33 & 0.33 \\ 0.33 & 0.33 & 0.33 \\ 0.33 & 0.33 & 0.33 \end{bmatrix}, \quad P = \begin{bmatrix} 0.33 & 0.33 & 0.33 \\ 0.33 & 0.33 & 0.33 \\ 0.33 & 0.33 & 0.33 \end{bmatrix}, \quad \text{KL} = 0$$

4. Centers on top of each other in the middle of the data: $(0.5, 0.5)$.

$$Q = \begin{bmatrix} 0.33 & 0.33 & 0.33 \\ 0.33 & 0.33 & 0.33 \\ 0.33 & 0.33 & 0.33 \end{bmatrix}, \quad P = \begin{bmatrix} 0.33 & 0.33 & 0.33 \\ 0.33 & 0.33 & 0.33 \\ 0.33 & 0.33 & 0.33 \end{bmatrix}, \quad \text{KL} = 0$$

5. Centers on top of each other outside the data: $(10, 10)$.

$$Q = \begin{bmatrix} 0.33 & 0.33 & 0.33 \\ 0.33 & 0.33 & 0.33 \\ 0.33 & 0.33 & 0.33 \end{bmatrix}, \quad P = \begin{bmatrix} 0.33 & 0.33 & 0.33 \\ 0.33 & 0.33 & 0.33 \\ 0.33 & 0.33 & 0.33 \end{bmatrix}, \quad \text{KL} = 0$$

6. Centers far from each other outside the data: $(10, 10)$, $(10, -10)$ and $(-10, -10)$.

$$Q = \begin{bmatrix} 0.33 & 0.33 & 0.33 \\ 0.38 & 0.31 & 0.31 \\ 0.35 & 0.35 & 0.29 \end{bmatrix}, \quad P = \begin{bmatrix} 0.31 & 0.33 & 0.36 \\ 0.4 & 0.29 & 0.31 \\ 0.35 & 0.38 & 0.27 \end{bmatrix}, \quad \text{KL} = 0.004$$

These cases reveal that while P values work as intended and require stricter cluster assignments, the loss associated with the optimal solution (1) is non-zero, while arbitrarily many cases with a zero loss can be found. This likely leads to the network collapsing the clusters on top of one another to minimise the loss. Having cluster centers far away from each other as well as the data has the same effect, because little differences between great distances matter less. Thus, the cluster hardening loss, at least as it is defined with the Kullback-Leibler divergence, is ill-defined.

7 Results

The final forecasting error metrics, for each model and data split separately, can be found in Table 5. The table contains reference models: constant zero (Zero), last known value (Last), daily mean (Mean), previous day (Prev) and ARIMA, after which the unclustered ForecastNet (FNet single) and clustered ForecastNet (FNet many) models are presented. Individual metrics are calculated for each forecast separately and averaged, aggregated metrics measure the error of total consumption and its prediction as the sum of individual predictions. The best results in each column are written in bold. Due to the different proportions of data, aggregated MAE and RMSE are comparable across different models, but not across data splits. The results reveal both the viability of naïve methods when compared to ARIMA and the lack of improvement from per-cluster models.

Table 5. Forecasting results.

Model	Split	Individual			Aggregated			
		MAE	RMSE	MAAPE	MAE	RMSE	MAAPE	MASE
Zero	train	0.765	0.900	43.93	988.1	999.4	50.00	2965
	val	0.765	0.901	43.85	329.6	333.6	50.00	3029
	test	0.678	0.814	43.86	292.2	296.1	50.00	3225
Last	train	0.457	0.602	25.70	134.6	163.1	9.86	444
	val	0.461	0.607	25.67	48.4	58.1	10.68	465
	test	0.450	0.588	25.73	54.2	63.3	12.71	605
Mean	train	0.362	0.479	33.89	149.4	178.9	10.23	457
	val	0.362	0.479	33.85	51.3	61.4	10.52	452
	test	0.345	0.457	35.16	47.4	56.4	11.06	527
Prev	train	0.356	0.540	23.31	68.3	80.5	4.29	190
	val	0.355	0.540	23.29	26.5	31.6	5.13	208
	test	0.335	0.514	22.84	24.3	29.2	5.24	262
ARIMA	train	0.351	0.469	40.44	133.0	161.1	9.48	433
	val	0.351	0.469	40.60	45.9	55.4	9.80	420
	test	0.333	0.446	42.28	43.3	51.8	10.48	515
FNet (Single)	train	0.285	0.399	37.50	61.9	72.1	3.88	166
	val	0.289	0.405	37.59	23.2	27.4	4.45	197
	test	0.277	0.391	39.71	21.0	25.1	4.55	214
FNet (Many)	train	0.293	0.406	37.15	62.2	71.9	3.88	164
	val	0.304	0.422	37.32	24.5	28.6	4.62	213
	test	0.285	0.401	39.42	21.4	25.5	4.63	224

Forecasting errors of testing data are smaller than training or validation across individual

metrics, indicating that the split could contain less problematic cases. Interestingly, the aggregated metrics seem to suggest the opposite. Compared to the constant zero forecasts, using the last known value or a daily mean improves results significantly. However, the best performing reference model, particularly with aggregated forecasts, is the previous day model. It almost achieves the performance of ForecastNet models, which do clearly produce the best results. Clustered models perform slightly worse.

Let us also examine some series whose predictions were especially good or poor. Figures 29 and 30 contain the best and worst predictions on training data for the single and clustered models, ordered by RMSE and MAAPE respectively. Similar graphs for all data splits can be found in Appendix 4. As the figures suggest, using different metrics to order predictions reflects their definitions. As expected, RMSE is clearly dominated by the scale of the series, while with MAAPE all the worst results come from a zero forecast horizon, but a nonzero prediction.

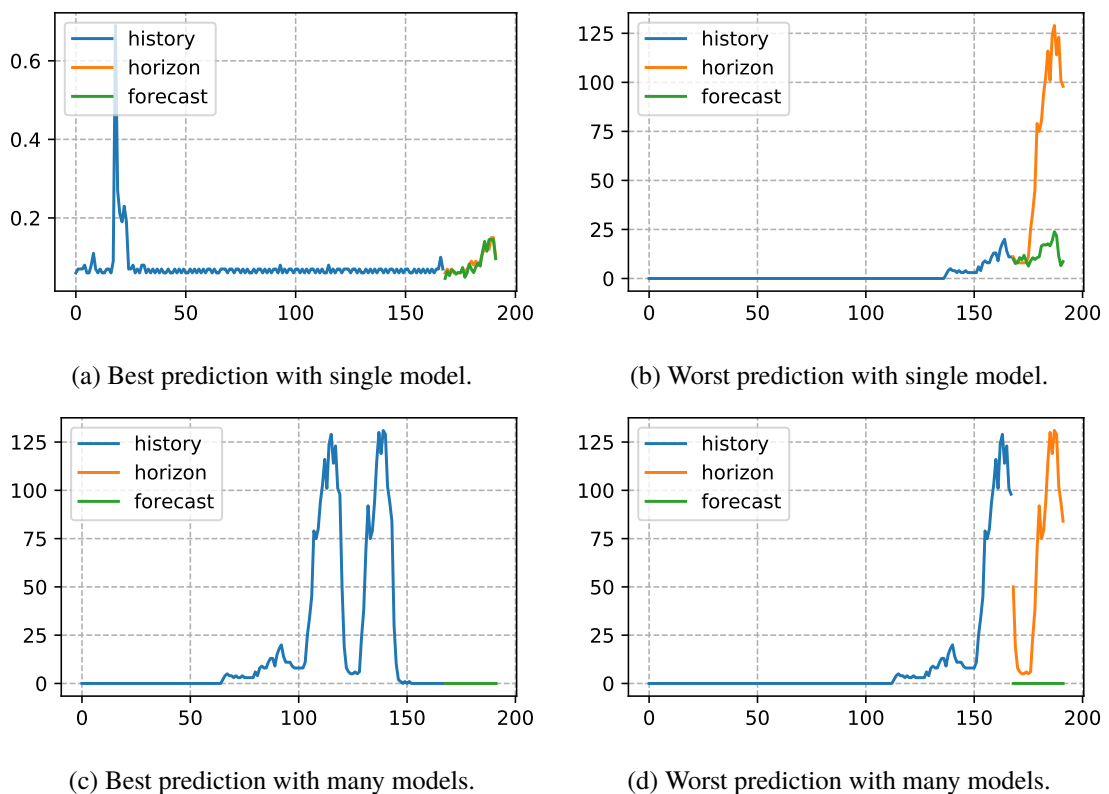
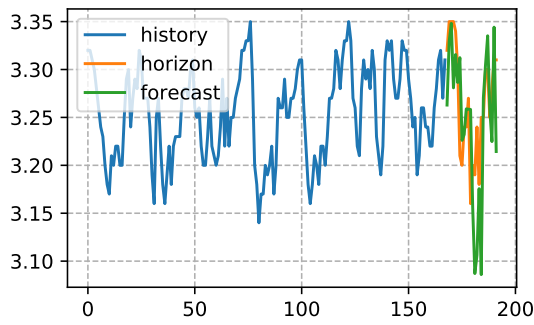
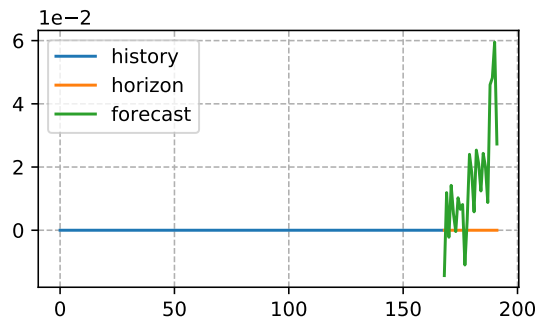


Figure 29. Example predictions on training data ordered with RMSE.

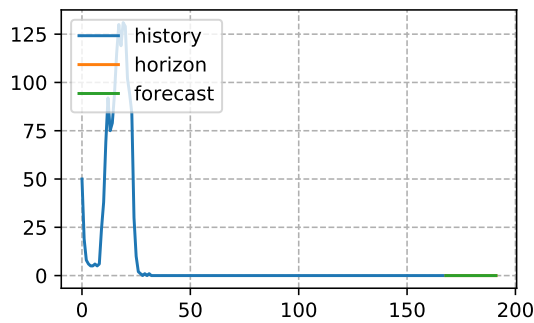
Another point of interest are error values of the forecast horizon by hour. Figure 31 visualises the errors by hour for both the unclustered and clustered models. The graphs have a similar structure, though the values of the clustered models are higher overall.



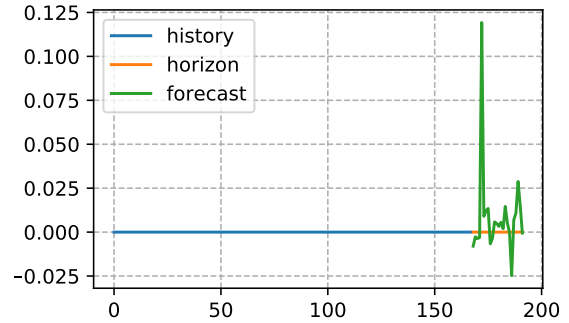
(a) Best prediction with single model.



(b) Worst prediction with single model.



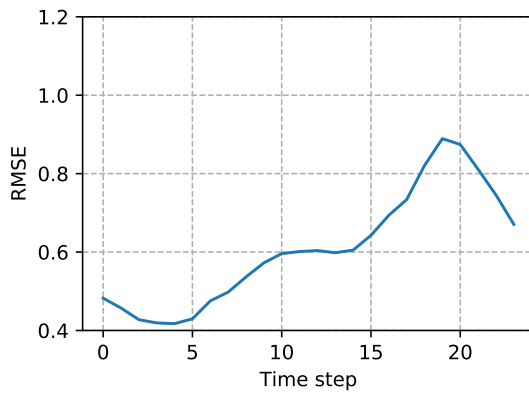
(c) Best prediction with many models.



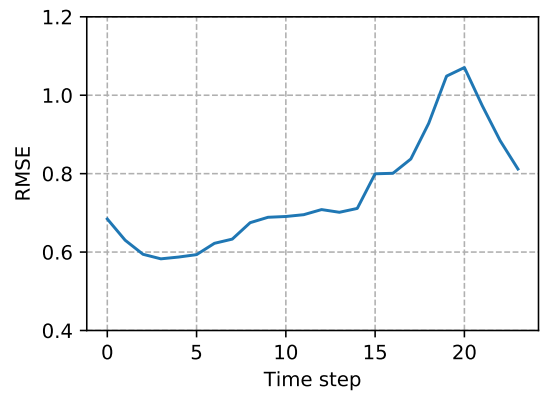
(d) Worst prediction with many models.

Figure 30. Example predictions on training data ordered with MAAPE.

Although there is an increasing trend, which is in line with the intuition that forecasts are more reliable temporally close to known values, the drops corresponding to early morning hours and late evening suggest that the daily cycle might also affect the difficulty of forecasting.



(a) Errors for single model.



(b) Errors for many models.

Figure 31. RMSE by forecast horizon hour.

8 Discussion

Data exploration revealed that the data set is likely best represented as a single cluster with a dense center and outliers. The position of consumers shifts over time, such that yearly cycles and patterns can be observed. Condensing the information of each series with an autoencoder led to a similar result, which suggests that there are no distinct clusters to be found in the data, and that the encoding may even be equivalent to PCA as the data was best represented with linear components. Nevertheless, applying k -means after determining the appropriate number of clusters led to a sensible result, although the number of samples in each cluster varied greatly.

The two most significant differences between consumers were overall consumption and night-time activity, as revealed by PCA. This indicates that consumer profiling based on simple factors is possible, despite the fact that distinct groups were not found. Some of the background data, like residence size or the number of residents, could therefore be directly used as a basis for clustering, because they are tightly associated with total consumption. With so much missing background data, it was not considered a reliable source for clustering in this project, but given relevant and reliable labels on enough consumers, background data alone could be used for clustering. However, in that case the effect of changing consumption patterns is not taken into account.

Forecasting results supported the findings of data exploration. A relatively small Forecast-Net achieved acceptable forecasting results, while using similar models for each cluster of consumers did not improve the result. This further solidified the notion that the data forms a single distinct cluster. Predictions for the smallest cluster seemed to fail completely, forecasting zero values while consumption was consistently high, which could indicate that the furthest cluster contains many problematic cases or that a few hundred samples is too little data. Results also showed that simple models, like the previous day model, are viable especially for aggregated load forecasting.

It was impractical to conduct experiments involving large neural networks, which is why conservative architectures were preferred. Even though more complex models showed no signs of significant gains, they could model the data in a new way. For example, the fact that autoencoding appeared to match PCA in dimensionality reduction could very well be the by-product of choosing architectures with lesser representation capabilities in the first place.

To account for the limited scope of this project, the major shortcomings of the thesis

are presented. They can be summarised with two related categories, arbitrary choices of methodology and lack of consideration for other options and standard practices. The shortcomings are as follows:

- The choice of maximum number of samples for data interpolation during preprocessing was arbitrary. Shorter spans of zeros can very well contain valid values, while longer ones may as well be invalid.
- Discarding all series that contained missing values was wasteful. Valid parts of the series could have been used because of short forecast histories.
- Weather information was not taken into account. Particularly temperature is a major factor in electricity consumption, and its effect should be considered.
- Different forecast history and horizon lengths were not considered. They surely affect the quality of forecasts, but both were chosen arbitrarily.

In addition to accounting for the limitations in this project, future work could focus on the following:

- In-depth examination of the cluster hardening loss and whether or not it has any value in its current formulation.
- Analysis of different data sets to further validate these results or provide new insights.
- Testing the effect of different forecast starting points.

Because both clustering and forecasting are essential tools in consumption analysis, this work could eventually be used as a basis for developing a single model that is capable of doing both. It could be valuable if the results are guaranteed to be appropriate for both aims.

9 Conclusion

As electric grids are modernised and gathering electricity consumption data becomes easier, new opportunities of leveraging that information arise. In this thesis, a method of clustering consumers and predicting their consumption was evaluated. Week-long, hourly series of each consumer were reduced using an autoencoder, and then clustered with k -means. The clustering was largely arbitrary, as no distinct groups were found. One-day-ahead forecasting was done for both the data set as a whole and for each cluster separately. Although the ForecastNet outperformed reference methods, prior clustering did not benefit forecasting. The clusters themselves may, however, be useful in consumer profiling. As simple tests revealed, cluster hardening, as presented by Aljalbout et al. in [8], is ill-defined and should be revised for further use.

REFERENCES

- [1] Stan M. Kaplan. Electric power transmission: background and policy issues. Library of Congress, Congressional Research Service, 2009.
- [2] Matteo Vasirani and Sascha Ossowski. Smart consumer load balancing: state of the art and an empirical evaluation in the Spanish electricity market. *Artificial Intelligence Review*, 39(1):81–95, 2013.
- [3] Fingrid. Open data on grid load and generation. <https://www.fingrid.fi/en/electricity-market/electricity-market-information/load-and-generation/>. Accessed: 2020-02-05.
- [4] H. Lopes Ferreira, A. Costescu, Angelo L’Abbate, Philip Minnebo, and Gianluca Fulli. Distributed generation and distribution market diversity in Europe. *Energy Policy*, 39(9):5561–5571, 2011.
- [5] Geoffrey K.F. Tso and Kelvin K.W. Yau. Predicting electricity energy consumption: A comparison of regression analysis, decision tree and neural networks. *Energy*, 32(9):1761–1768, 2007.
- [6] Xiaoyue Wang, Abdullah Mueen, Hui Ding, Goce Trajcevski, Peter Scheuermann, and Eamonn Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26(2):275–309, March 2013.
- [7] Martin Långkvist, Lars Karlsson, and Amy Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24, June 2014.
- [8] Elie Aljalbout, Vladimir Golkov, Yawar Siddiqui, Maximilian Strobel, and Daniel Cremers. Clustering with Deep Learning: Taxonomy and New Methods. *arXiv:1801.07648 [cs, stat]*, January 2018. arXiv: 1801.07648.
- [9] Abdulaziz Almalaq and George Edwards. A review of deep learning methods applied on load forecasting. In *2017 16th IEEE international conference on machine learning and applications (ICMLA)*, pages 511–516. IEEE, 2017.
- [10] Wan He. Load forecasting via deep neural networks. *Procedia Computer Science*, 122:308–314, 2017.
- [11] Mahmoud A. Hammad, Borut Jereb, Bojan Rosi, and Dejan Dragan. Methods and models for electric load forecasting: A comprehensive review. *Logistics & Sustainable Transport*, 11(1):51–76, 2020.

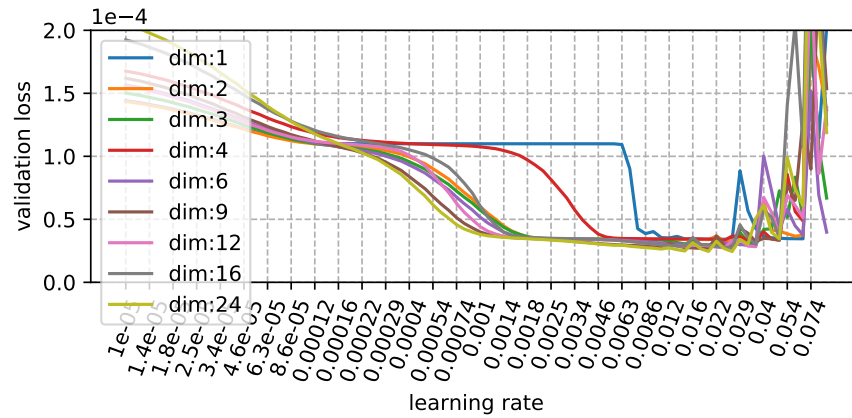
- [12] Ahmad S. Ahmad, Mohammad Y. Hassan, Md P. Abdullah, Hasimah A. Rahman, Faridah N. Hussin, Hayati Abdullah, and Rahman Saidur. A review on applications of ANN and SVM for building electrical energy consumption forecasting. *Renewable and Sustainable Energy Reviews*, 33:102–109, 2014.
- [13] Richard E. Edwards, Joshua New, and Lynne E. Parker. Predicting future hourly residential electrical consumption: A machine learning case study. *Energy and Buildings*, 49:591–603, 2012.
- [14] Mohammad A. Mat Daut, Mohammad Yusri Hassan, Hayati Abdullah, Hasimah A. Rahman, Md P. Abdullah, and Faridah Hussin. Building electrical energy consumption forecasting analysis using conventional and artificial intelligence methods: A review. *Renewable and Sustainable Energy Reviews*, 70:1108–1118, 2017.
- [15] Haiwen Chen, Shouxiang Wang, Shaomin Wang, and Ye Li. Day-ahead aggregated load forecasting based on two-terminal sparse coding and deep neural network fusion. *Electric Power Systems Research*, 177:105987, 2019.
- [16] Peter Laurinec, Marek Lóderer, Petra Vrablecová, Mária Lucká, Viera Rozinajová, and Anna Bou Ezzeddine. Adaptive time series forecasting of energy consumption using optimized cluster analysis. In *2016 IEEE 16th international conference on data mining workshops (ICDMW)*, pages 398–405. IEEE, 2016.
- [17] Zuhaina Zakaria, K.L. Lo, and Mohamad Hadi Sohod. Application of fuzzy clustering to determine electricity consumers' load profiles. In *2006 IEEE International Power and Energy Conference*, pages 99–103. IEEE, 2006.
- [18] Gianfranco Chicco, Roberto Napoli, and Federico Piglione. Comparisons among clustering techniques for electricity customer classification. *IEEE Transactions on Power Systems*, 21(2):933–940, 2006.
- [19] Runming Yao and Koen Steemers. A method of formulating energy load profile for domestic buildings in the UK. *Energy and Buildings*, 37(6):663–671, 2005.
- [20] Fintan McLoughlin, Aidan Duffy, and Michael Conlon. A clustering approach to domestic electricity load profile characterisation using smart metering data. *Applied Energy*, 141:190–199, 2015.
- [21] Ervin D. Varga, Sándor F. Beretka, Christian Noce, and Gianluca Sapienza. Robust real-time load profile encoding and classification framework for efficient power systems operation. *IEEE Transactions on Power Systems*, 30(4):1897–1904, 2014.

- [22] Ali Azadeh, Seyed F. Ghaderi, S. Tarverdian, and Morteza Saberi. Integration of artificial neural networks and genetic algorithm to predict electrical energy consumption. *Applied Mathematics and Computation*, 186(2):1731–1741, 2007.
- [23] Baran Yildiz, Jose I. Bilbao, and Alistair B. Sproul. A review and analysis of regression and machine learning models on commercial building electricity load forecasting. *Renewable and Sustainable Energy Reviews*, 73:1104–1122, 2017.
- [24] Chirag Deb, Fan Zhang, Junjing Yang, Siew Eang Lee, and Kwok Wei Shah. A review on time series forecasting techniques for building energy consumption. *Renewable and Sustainable Energy Reviews*, 74:902–924, 2017.
- [25] Allan Pinkus. Approximation theory of the MLP model in neural networks. *Acta numerica*, 8(1):143–195, 1999.
- [26] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.
- [27] Boris Hanin. Which neural net architectures give rise to exploding and vanishing gradients? In *Advances in Neural Information Processing Systems*, pages 582–591, 2018.
- [28] T. Warren Liao. Clustering of time series data—a survey. *Pattern Recognition*, 38(11):1857–1874, November 2005.
- [29] Naveen Sai Madiraju, Seid M. Sadat, Dimitry Fisher, and Homa Karimabadi. Deep Temporal Clustering: Fully Unsupervised Learning of Time-Domain Features. *arXiv:1802.01059 [cs, stat]*, February 2018. arXiv: 1802.01059.
- [30] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. Time-series clustering—a decade review. *Information Systems*, 53:16–38, 2015.
- [31] Peter J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.
- [32] David L. Davies and Donald W. Bouldin. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227, 1979.
- [33] Javier Contreras, Rosario Espinola, Francisco J. Nogales, and Antonio J. Conejo. ARIMA models to predict next-day electricity prices. *IEEE transactions on power systems*, 18(3):1014–1020, 2003.

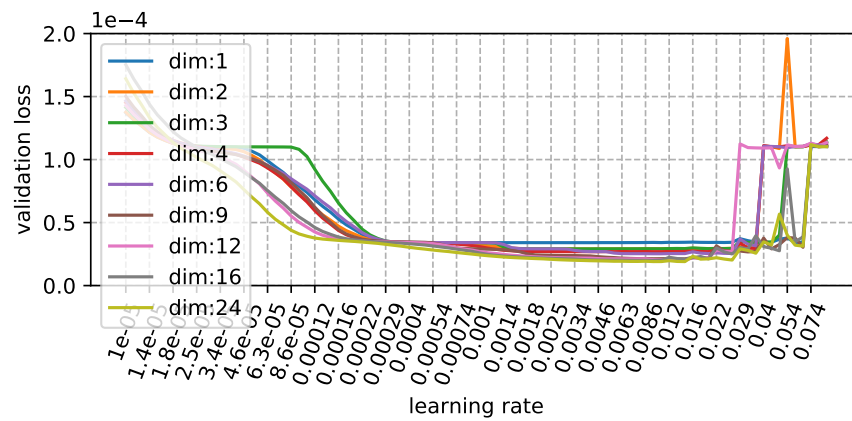
- [34] Jerome T. Connor, R. Douglas Martin, and Les E. Atlas. Recurrent neural networks and robust time series prediction. *IEEE Transactions on Neural Networks*, 5(2):240–254, 1994.
- [35] Zachary C. Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.
- [36] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
- [37] Andrew Karpathy. The unreasonable effectiveness of recurrent neural networks. <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>. Accessed: 2020-01-28.
- [38] Joel Janek Dabrowski, YiFan Zhang, and Ashfaqur Rahman. Forecastnet: A time-variant deep feed-forward neural network architecture for multi-step-ahead time-series forecasting. *arXiv*, pages arXiv–2002, 2020.
- [39] Rob J. Hyndman and Anne B. Koehler. Another look at measures of forecast accuracy. *International journal of forecasting*, 22(4):679–688, 2006.
- [40] Tianfeng Chai and Roland R. Draxler. Root mean square error (RMSE) or mean absolute error (MAE)?—Arguments against avoiding RMSE in the literature. *Geoscientific model development*, 7(3):1247–1250, 2014.
- [41] Philip Hans Franses. A note on the mean absolute scaled error. *International Journal of Forecasting*, 32(1):20–22, 2016.
- [42] Sungil Kim and Heeyoung Kim. A new metric of absolute percentage error for intermittent demand forecasts. *International Journal of Forecasting*, 32(3):669–679, 2016.
- [43] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.
- [44] Anil K. Jain. Data clustering: 50 years beyond K-means. *Pattern recognition letters*, 31(8):651–666, 2010.
- [45] Alex M. Lamb, Anirudh Goyal Alias Parth Goyal, Ying Zhang, Saizheng Zhang, Aaron C. Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. In *Advances in neural information processing systems*, pages 4601–4609, 2016.

- [46] Denis Kwiatkowski, Peter C.B. Phillips, Peter Schmidt, Yongcheol Shin, et al. Testing the null hypothesis of stationarity against the alternative of a unit root. *Journal of econometrics*, 54(1-3):159–178, 1992.
- [47] David Posada and Thomas R. Buckley. Model selection and model averaging in phylogenetics: advantages of Akaike information criterion and Bayesian approaches over likelihood ratio tests. *Systematic biology*, 53(5):793–808, 2004.
- [48] Leslie N. Smith. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv*, pages arXiv–1803, 2018.

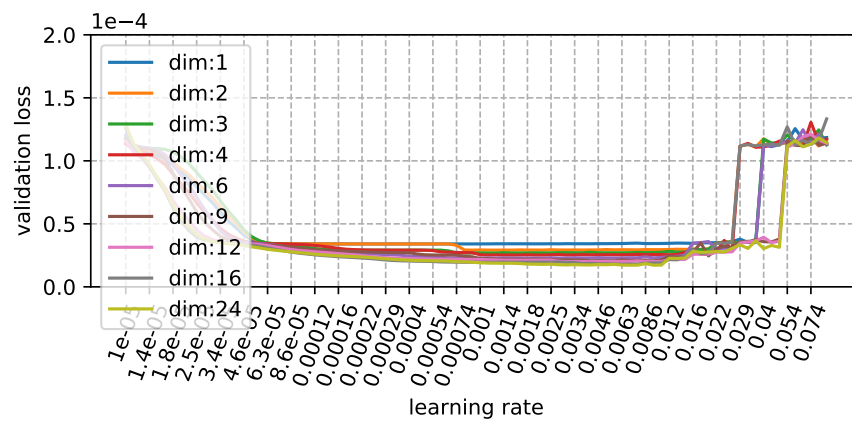
Appendix 1. Clustering with shallow autoencoder figures



(a) Test for 8 batches.



(b) Test for 64 batches.

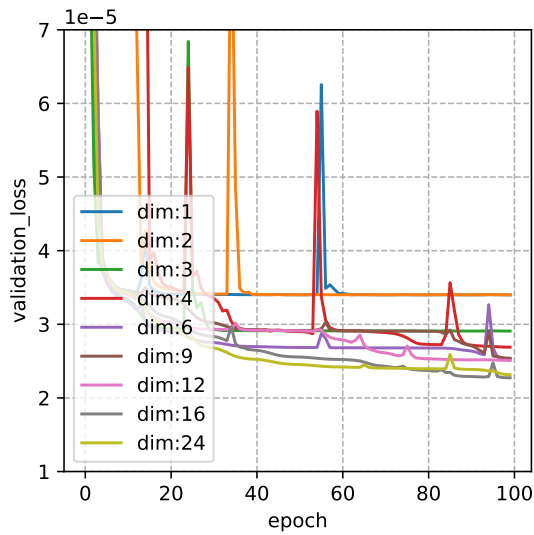


(c) Test for 512 batches.

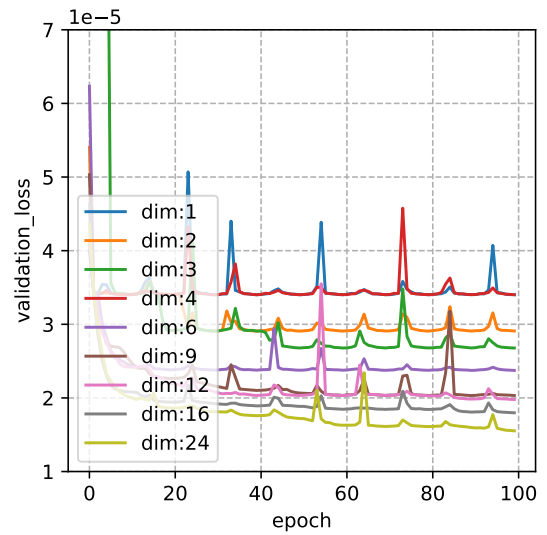
Figure A1.1. Learning rate range tests.

(continues)

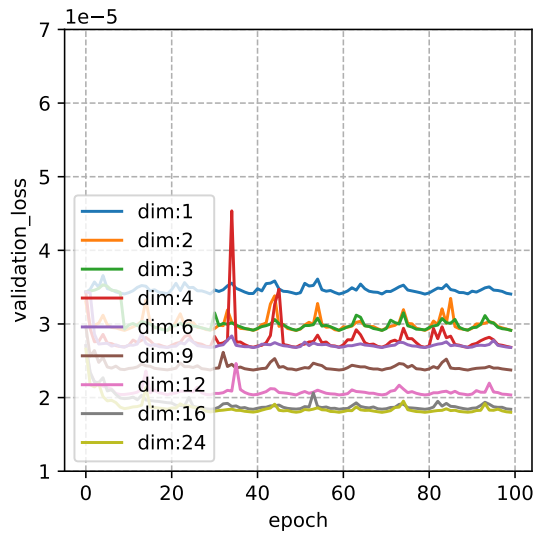
Appendix 1. (continued)



(a) Losses for 8 batches.



(b) Losses for 64 batches.

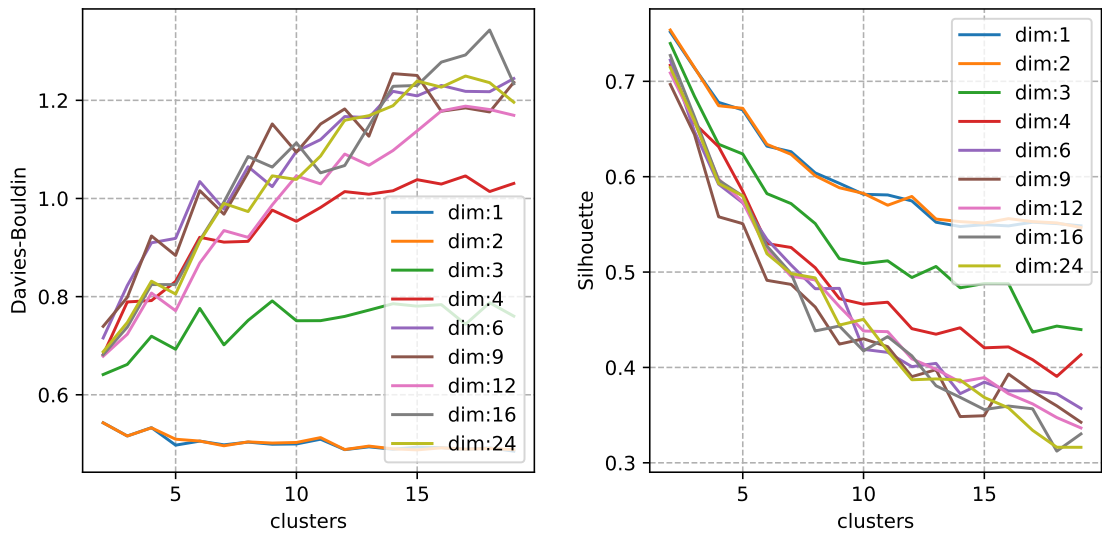


(c) Losses for 512 batches.

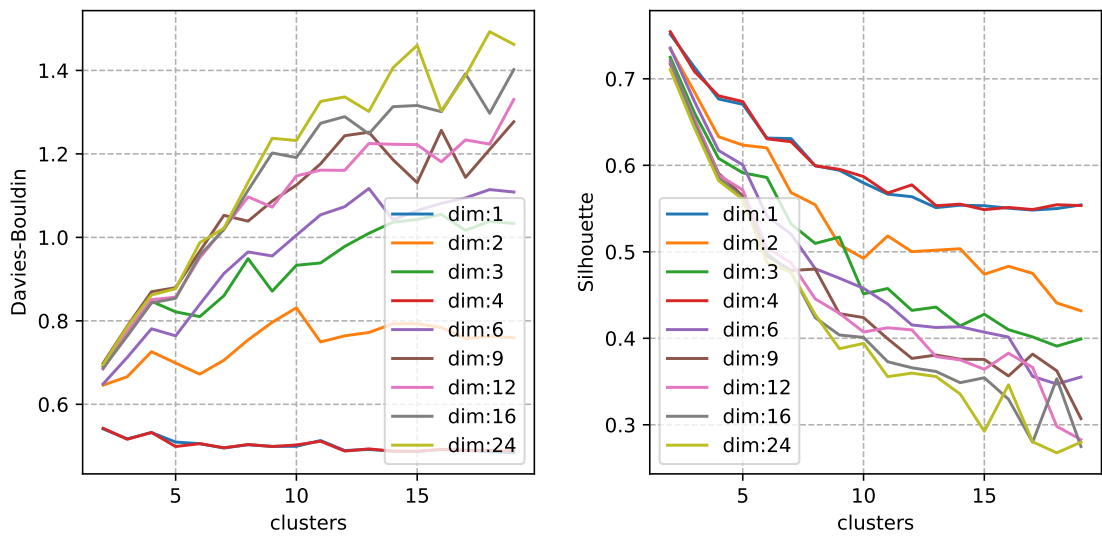
Figure A1.2. Validation losses.

(continues)

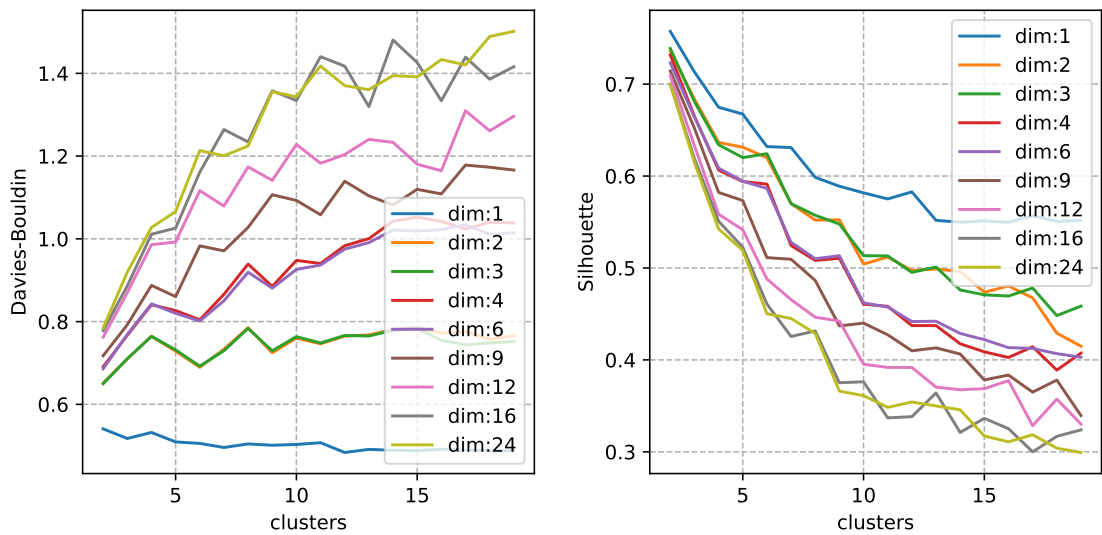
Appendix 1. (continued)



(a) Metrics with 8 batches.



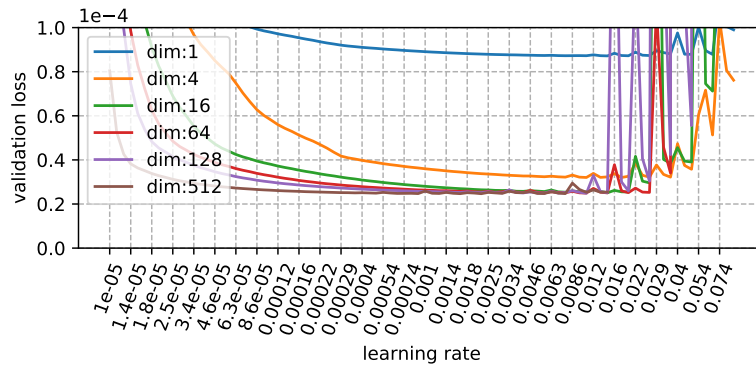
(b) Metrics with 64 batches.



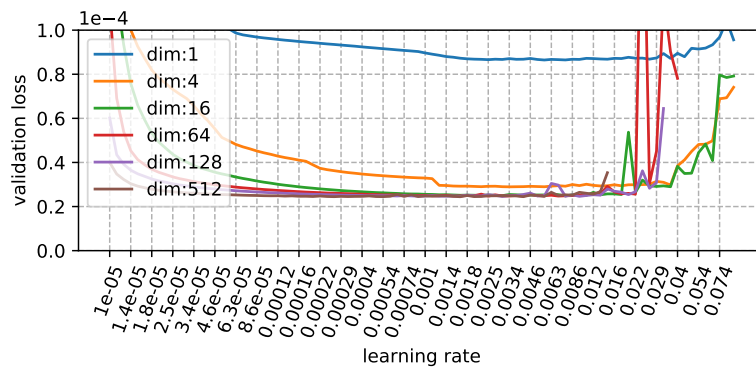
(c) Metrics with 512 batches.

Figure A1.3. Clustering metrics.

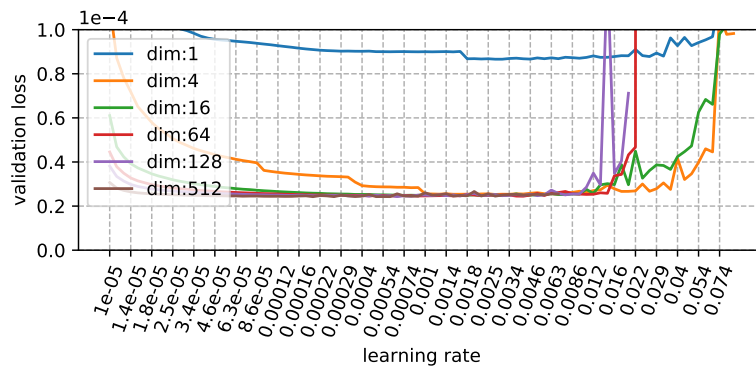
Appendix 2. Unclustered forecasting figures



(a) Test with 64 batches.



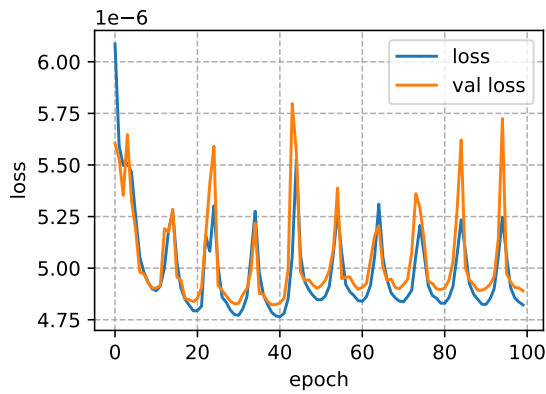
(b) Test with 128 batches.



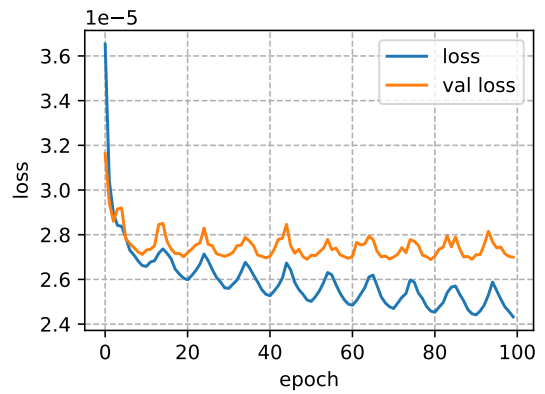
(c) Test with 512 batches.

Figure A2.1. Learning rate range tests.

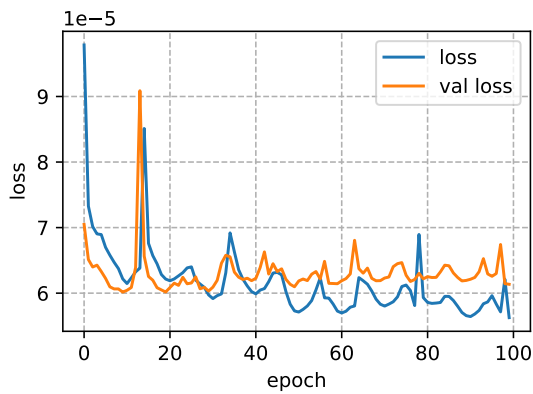
Appendix 3. Forecasting with clusters from shallow autoencoder figures



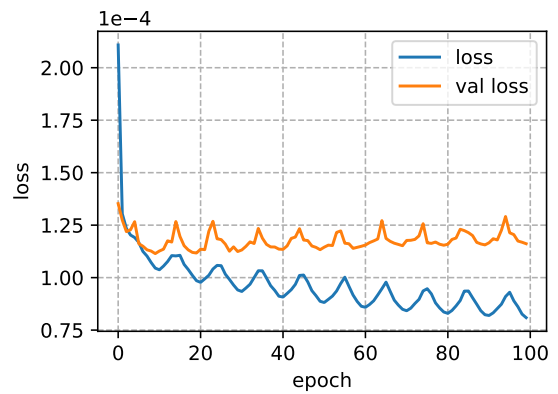
(a) Losses for cluster 0 (280 000 samples).



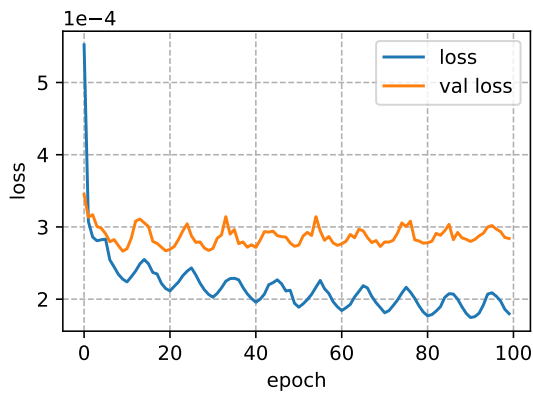
(b) Losses for cluster 1 (100 000 samples).



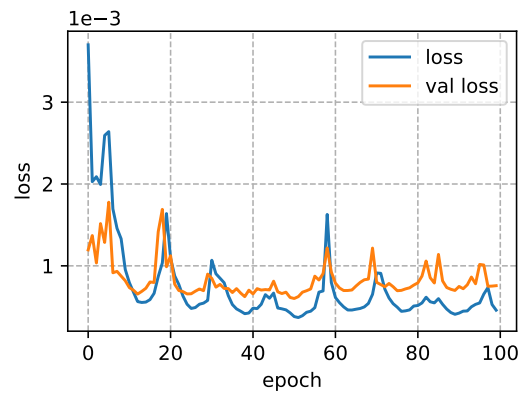
(c) Losses for cluster 2 (50 000 samples).



(d) Losses for cluster 3 (20 000 samples).



(e) Losses for cluster 4 (5 000 samples).

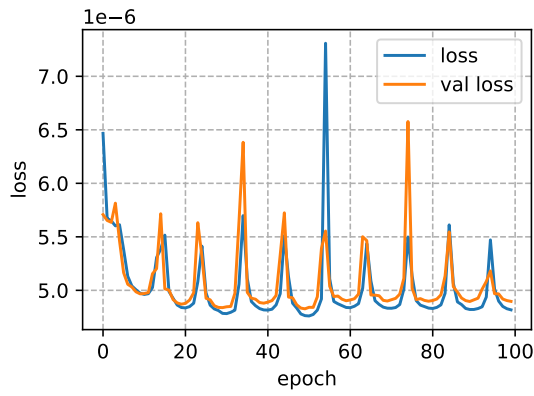


(f) Losses for cluster 5 (300 samples).

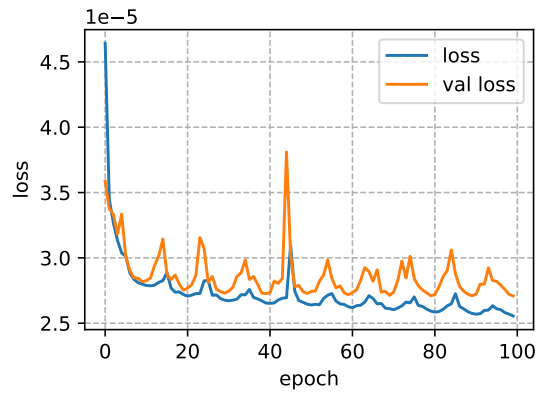
Figure A3.1. Clustered ForecastNets with constant number of batches.

(continues)

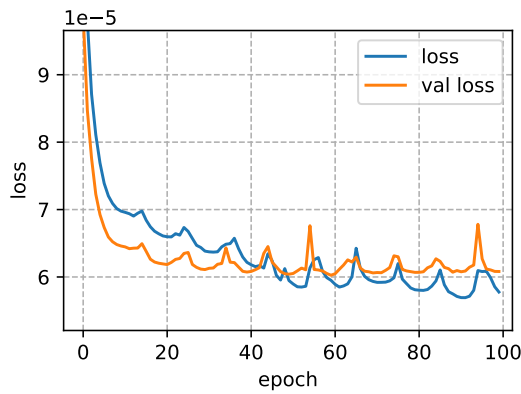
Appendix 3. (continued)



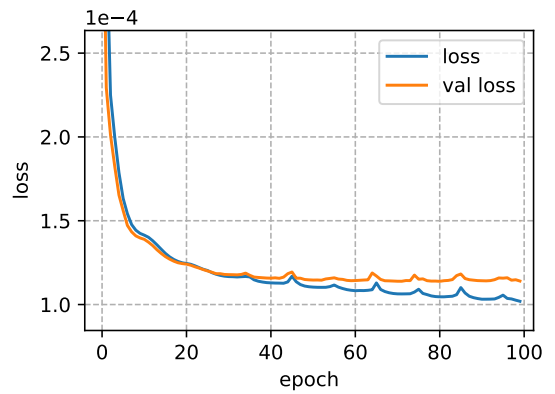
(a) Losses for cluster 0 (280 000 samples).



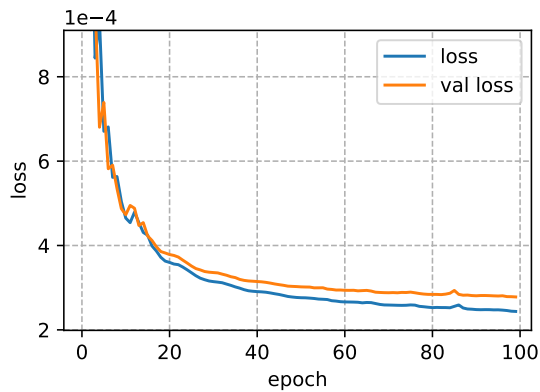
(b) Losses for cluster 1 (100 000 samples).



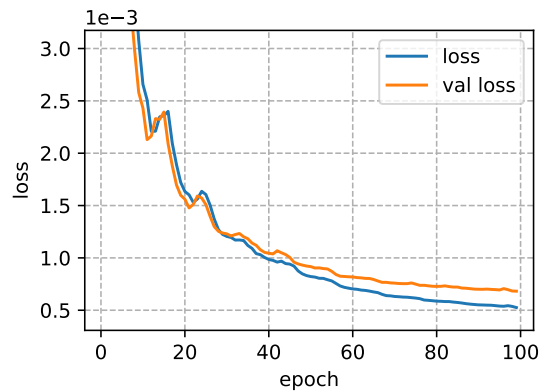
(c) Losses for cluster 2 (50 000 samples).



(d) Losses for cluster 3 (20 000 samples).



(e) Losses for cluster 4 (5 000 samples).

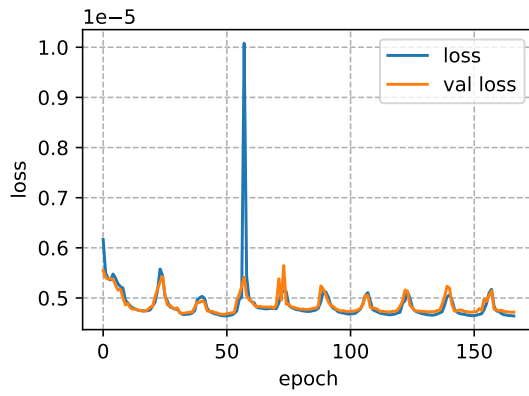


(f) Losses for cluster 5 (300 samples).

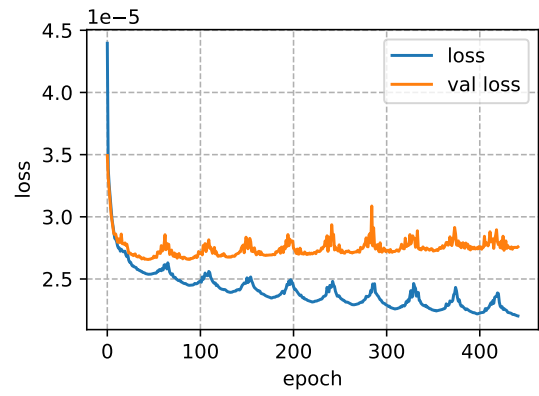
Figure A3.2. Clustered ForecastNets with constant batch size.

(continues)

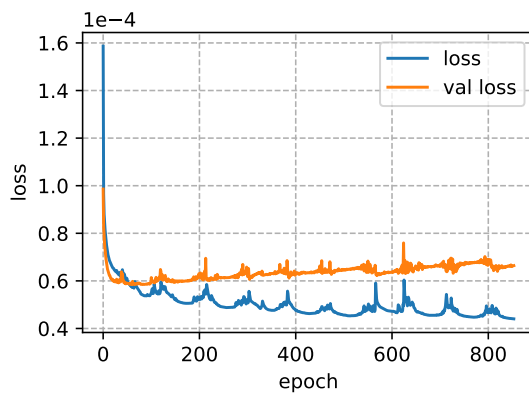
Appendix 3. (continued)



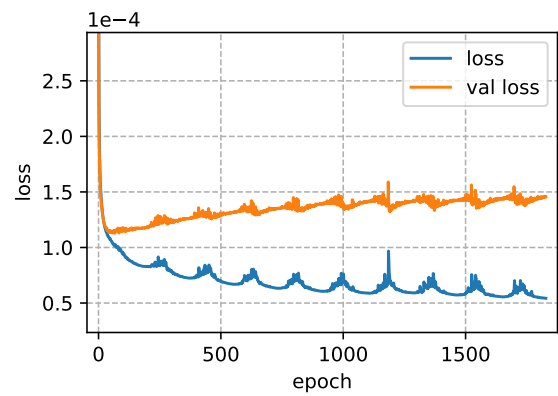
(a) Losses for cluster 0 (280 000 samples).



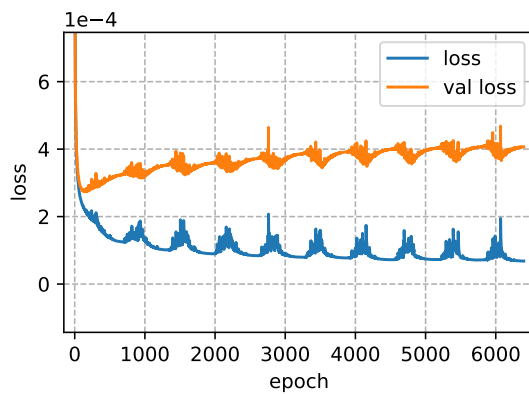
(b) Losses for cluster 1 (100 000 samples).



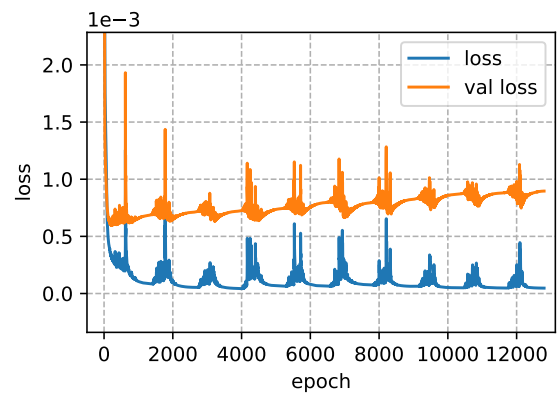
(c) Losses for cluster 2 (50 000 samples).



(d) Losses for cluster 3 (20 000 samples).



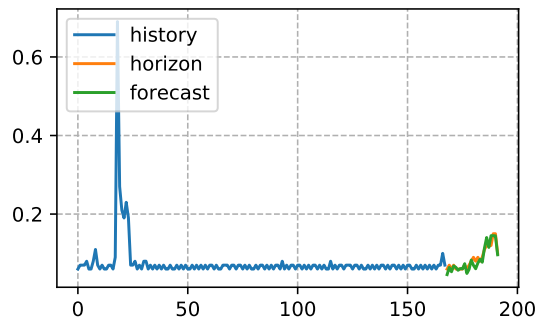
(e) Losses for cluster 4 (5 000 samples).



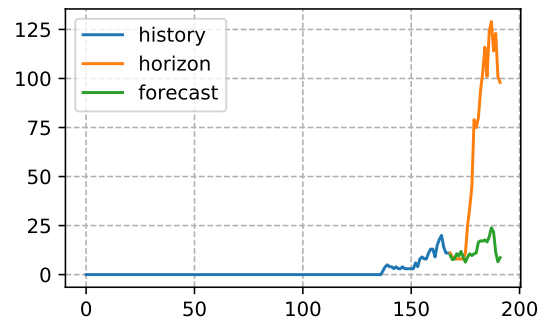
(f) Losses for cluster 5 (300 samples).

Figure A3.3. Clustered ForecastNets with constant batch size and number of model updates.

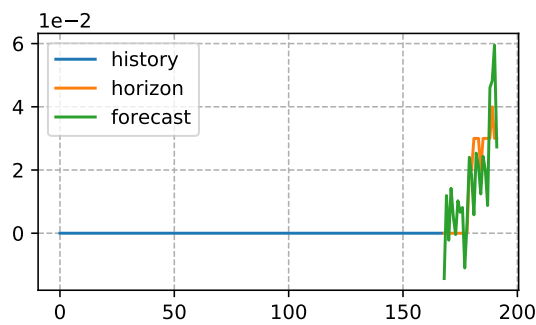
Appendix 4. Best and worst predictions of unclustered and clustered models



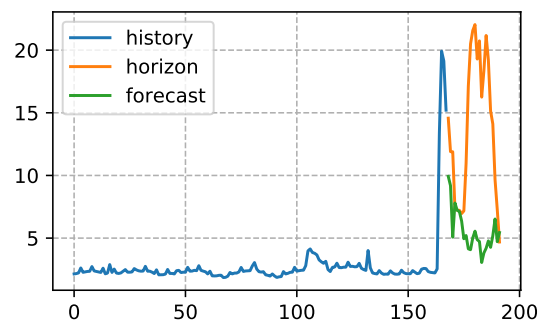
(a) Best prediction with training data.



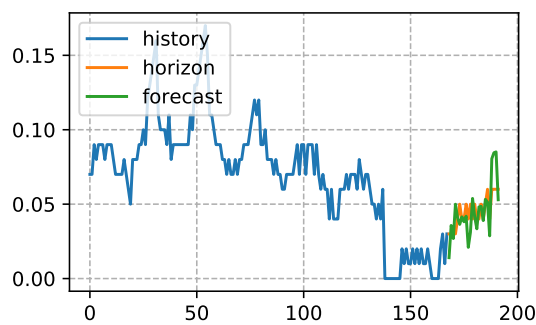
(b) Worst prediction with training data.



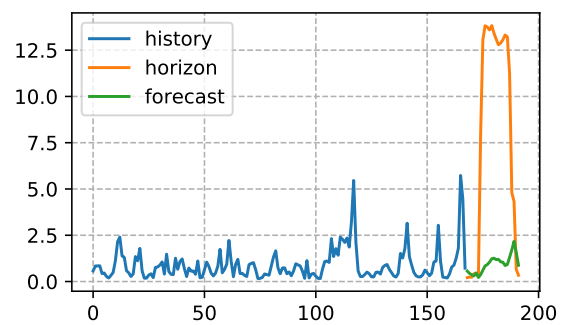
(c) Best prediction with validation data.



(d) Worst prediction with validation data.



(e) Best prediction with testing data.

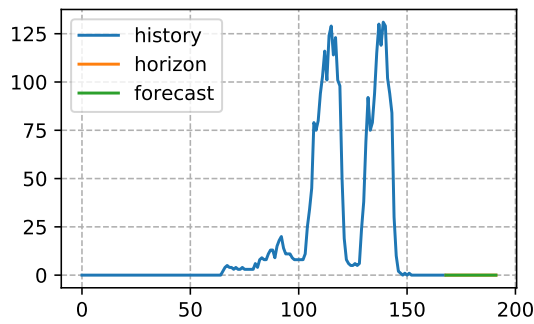


(f) Worst prediction with testing data.

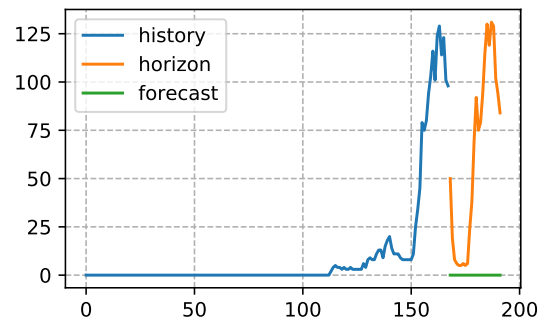
Figure A4.1. Example predictions with the single ForecastNet model ordered with RMSE.

(continues)

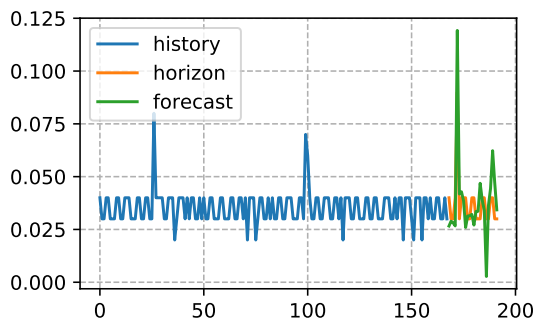
Appendix 4. (continued)



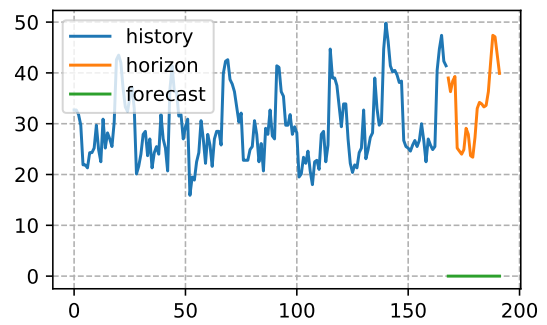
(a) Best prediction with training data.



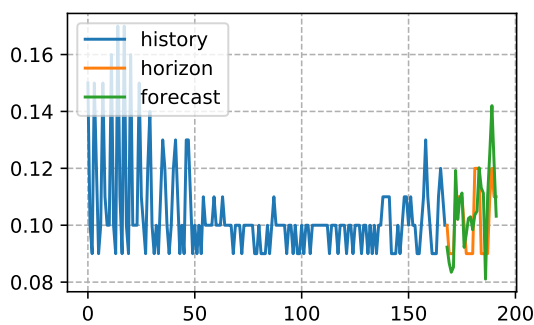
(b) Worst prediction with training data.



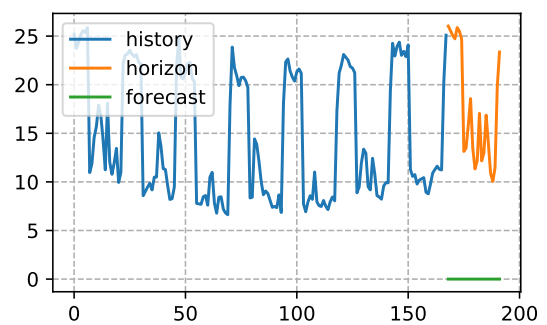
(c) Best prediction with validation data.



(d) Worst prediction with validation data.



(e) Best prediction with testing data.

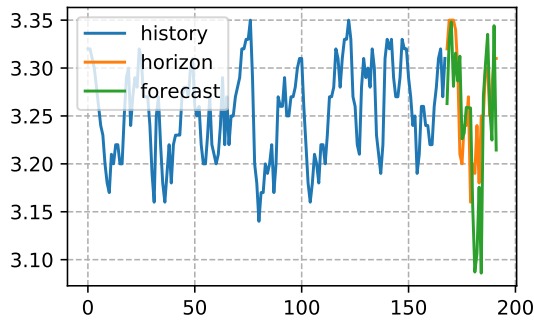


(f) Worst prediction with testing data.

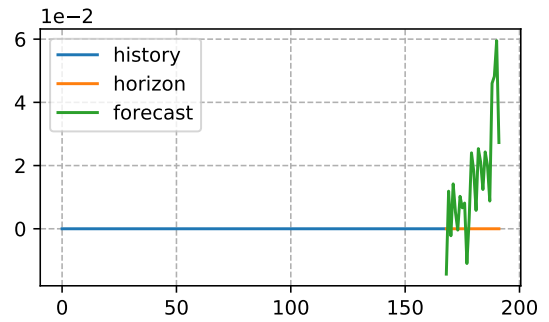
Figure A4.2. Example predictions with the clustered ForecastNet model ordered by RMSE.

(continues)

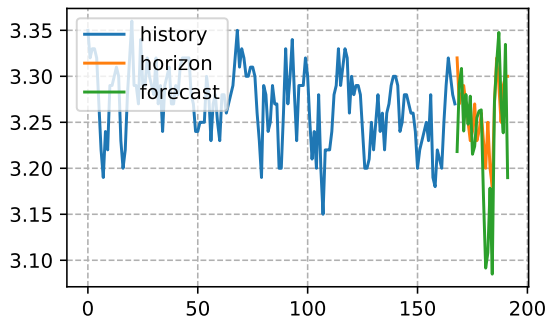
Appendix 4. (continued)



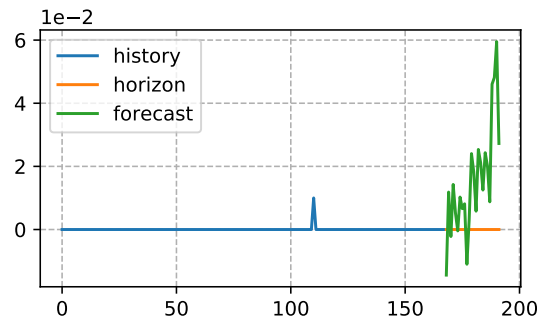
(a) Best prediction with training data.



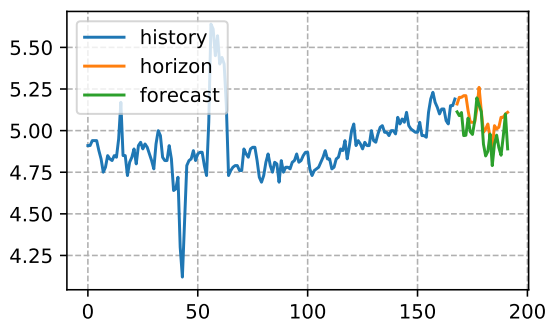
(b) Worst prediction with training data.



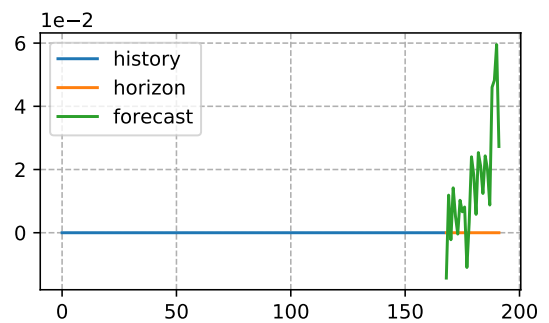
(c) Best prediction with validation data.



(d) Worst prediction with validation data.



(e) Best prediction with testing data.

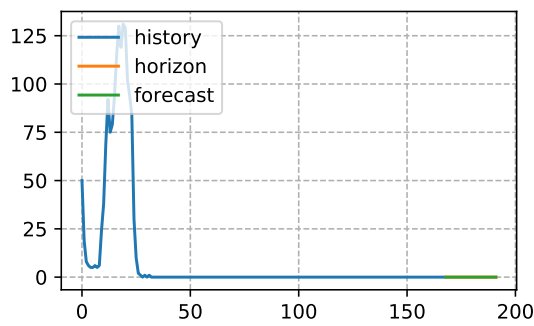


(f) Worst prediction with testing data.

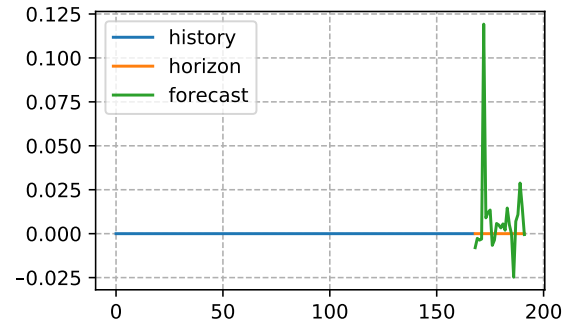
Figure A4.3. Example predictions with the single ForecastNet model ordered by MAAPE.

(continues)

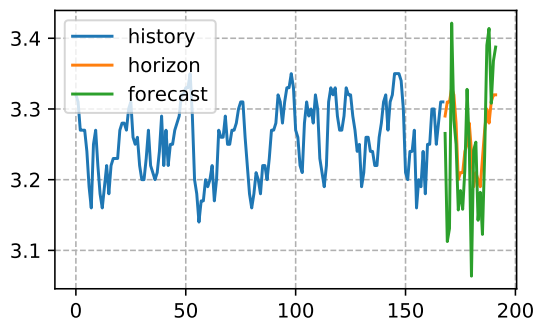
Appendix 4. (continued)



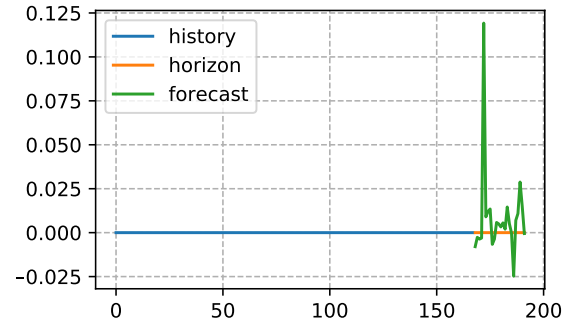
(a) Best prediction with training data.



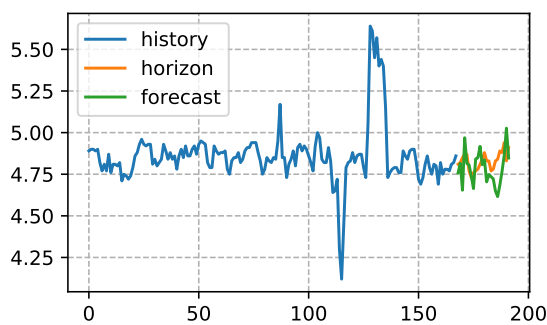
(b) Worst prediction with training data.



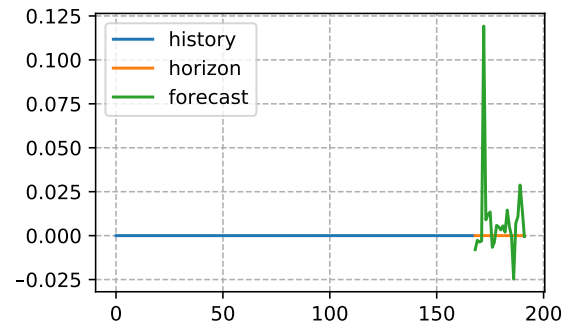
(c) Best prediction with validation data.



(d) Worst prediction with validation data.



(e) Best prediction with testing data.



(f) Worst prediction with testing data.

Figure A4.4. Example predictions with the clustered ForecastNet model ordered by MAAPE.