# Forecasting of Heat Pump Power Consumption using Neural Networks

Gustav Warnström

Johan Fant

LUND
UNIVERSITY

Department of Automatic Control

Msc Thesis
TFRT-6191
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden


Printed in Sweden by Tryckeriet i E-huset
Lund 2023

# Contents

# Abstract

With the increase in electricity generation from renewable sources over recent years, the demand for ancillary services providing balancing support in the grid has risen. Demand side regulation can be performed by regulating consumption of some household devices, which can act as a virtual battery that can be loaded and unloaded into the grid. Among these devices are thermostatically controllable loads, e.g. heat pumps, AC:s and refrigerators. In order to create a virtual battery however, the nominal consumption of the load needs to be known, and in addition, market mechanisms requires this to be known ahead in time. This thesis has investigated predictive models using neural networks for thermostatically controllable loads in the form of heat pumps. The models were compared to two different naive predictors used as baselines. Results indicated that it is possible to beat these baselines, although it is difficult to fully evaluate the performance until put in real operation. Furthermore, it is unknown whether the presented models are optimal, and further work is likely required to find a more optimal model.

## Acknowledgements

We would like thank our supervisors Richard Pates at the Department of Control, Lund University and Daria Madjidian at Emulate Energy AB for the opportunity to do this interesting project, and for valuable discussions and help throughout.

Secondly, we would like to acknowledge restaurant Bryggan at the LTH Campus for terrific lunches during our years here at LTH. We will happily pay full prize at the restaurant from here on.

Lastly, we would also like to thank family and friends for their support along the way.

# Abbreviations

**LSTM** Long Short-Term Memory

**MAE** Mean Absolute Error

**MSE** Mean Squared Error

**NN** Neural Network

**ReLU** Rectified Linear Unit

**RNN** Recurrent Neural Network

**SARIMA** Seasonal Auto-Regressive Integrated Moving-Average

**TCL** Termostatically Controlled Load

**UAT** Universal Approximation Theorem

# 1 Introduction

This report presents work that has been conducted at Emulate Energy in Lund, Sweden during summer and fall 2022.

## 1.1 Background

With the increase in electricity generation from renewable sources over recent years, the demand for ancillary services providing balancing support in the grid has risen. As most sources of renewable energy production are intermittent, such as solar and wind, the supply of electricity has become more volatile as renewable energy sources such as these make up a prominent part of the electricity production. Providing balancing support by burning fossil fuels to produce electricity would reduce the environmental benefit from integrating renewables into the electricity grid. Therefore demand side regulation is deemed to play a significant role in ensuring deep integration of renewable energy sources in the electricity grid [4].

In Sweden the System operator Svenska Kraftnät (SVK) purchases ancillary services to maintain the system frequency in the grid to be 50 Hz. Among the ancillary services that SVK purchases are FCR-N (Frequency Containment Reserve - Normal), FCR-D Up and FCR-D Down (Frequency Containment Reserve - Disturbances), which are used to balance the frequency at normal operation (between 49.90 - 50.10 Hz) and under disturbances (Up: 49.9-49.5 Hz, Down: 50.1-50.5 Hz), respectively [8]

On the demand side, deferrable energy loads can be used to provide balancing support. These include, among others, thermostatically controlled loads, TCLs (heat pumps, ACs, waters heaters e.t.c.) and electric vehicle charging. However, in this thesis TCLs in the form of heat pumps will exclusively be considered. The flexibility of TCLs are due to that various power trajectories can meet user-specified temperature constraints [4]. Knowing the nominal power of a TCL, one can aggregate these into a virtual battery that can be loaded by or, unloaded into, the electricity grid. A model for the power consumption of a heat pump for some exogenous signals may enable a cluster of heat pumps to act as a virtual battery. Although, since electricity is traded one day ahead or more for delivery, anyone having "access" to the virtual battery needs to know its capacity, and therefore a predictive model for the nominal power consumption of the heat pumps is necessary.

## 1.2 Problem Formulation

This thesis aims to answer the following questions:

- Can a Recurrent Neural Network improve predictions of heat pump power consumption over current industry baseline methods?
- Given a model is created, how could it be improved further? And using what methods?

Most of the work for the thesis was conducted to create a framework for a data driven approach to modelling of power consumption with the Emulate API. It should be considered as an investigation in the feasibility of such an approach, meaning that all models might not be optimal. Possible ways of continuing the work and applications will be discussed in section 5

# 2 Theory

This section gives an introduction to research that provides a background to this thesis, as well give a brief go-through of the machine learning methods employed in the thesis.

## 2.1 Prior Work

The basis of this thesis is research conducted on demand side frequency regulation using generalized battery models. Battery models for demand-side regulation using thermostatically controlled loads has been studied by Hao et. al in [4]. Hao et. al. characterizes the battery model's power limits and energy capacity as well as proposes a control strategy to provide frequency regulation. This thesis does not intend to create a full battery model, instead the Hao paper serves as a motivation of the work conducted in this thesis project. In the conclusion, the Hao paper points out a few research issues that have to be addressed, whereas one is "estimating the overall hourly availability of TCLs using historic measurement data" and this is the issue to be addressed in this thesis. Effectively this means creating a reliable model of the heat pump, that outputs the power consumption under conditions where no external control is exerted on the pumps, i.e. the nominal power consumption. Ideally, as the majority of the power in the grid is traded a day ahead, in order to monetize on a battery model, one would want to be able to make reliable predictions of nominal consumption 36-48 hours ahead.

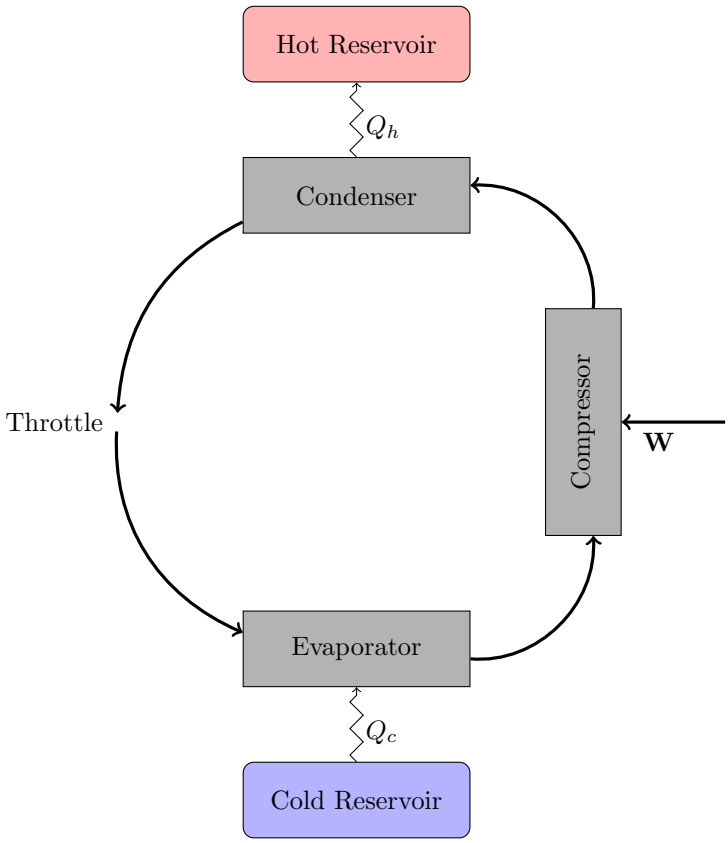
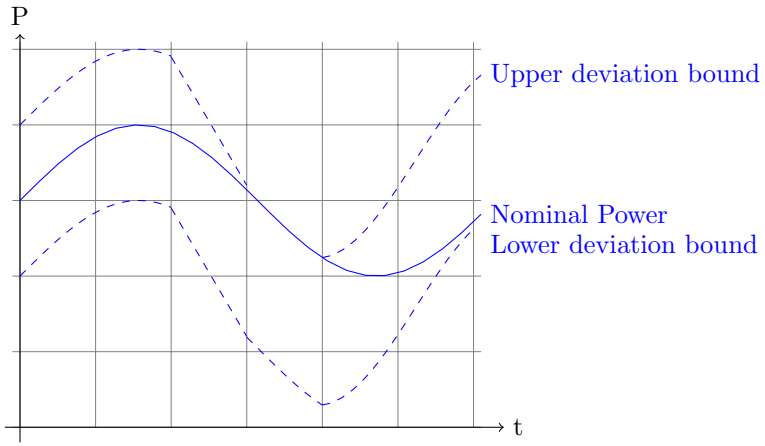



Figure 1: Overview of how a heat pump works. Inspiration to picture taken from [3, p.138].

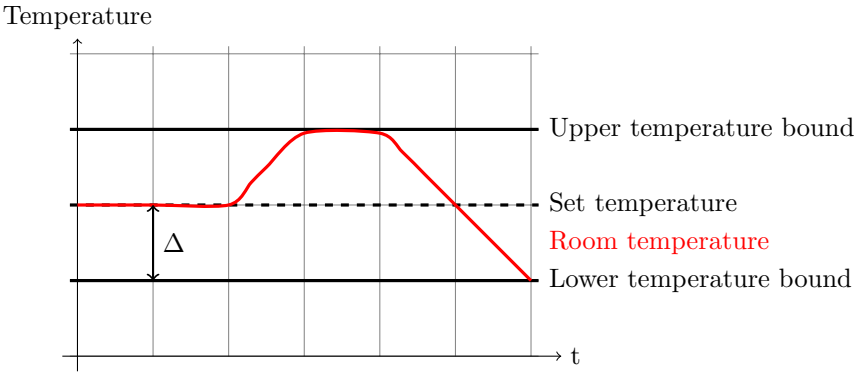
## 2.2 Heat Pumps

Heat pumps are heat engines operating in reverse [3]. That is, instead of absorbing heat and converting parts of it into work, work is added to the process and converts the work into heat. Figure 1 displays a schematic overview of how a heat pump works. In the closed loop, marked by the large arrows, a medium called refrigerant flows. Heat $Q_c$ is picked up from the cold reservoir, the outside of the house, such that the refrigerant evaporates[3]. Work $W$ is then added in the form of electrical energy that runs a compressor that raises the pressure and temperature of the gas. In the condenser the gas then gives up heat to the hot reservoir, the inside of the house, and the gas liquifies. At last the liquid passes a "throttling valve", where the pressure and temperature is further decreased such that the liquid once again can pick up heat from the cold reservoir, and the process can start over.

As mentioned in section 1.1, heat pumps can be used for the purpose of demand side flexibility due to that their power consumption can be modulated while still meeting temperature constraints set by the end user. This principle is depicted in figure 2, note that this is just a sketch of the principle. The filled line of figure 2a) represents the nominal power consumption $P_o$. The dashed lines represents the boundaries of how much the power trajectory can deviate from the nominal while the red curve representing the room temperature in figure 2b meets the temperature bounds. When the room temperature equals the set temperature, the power can be regulated to deviate both upwards or downwards from the nominal power. As the room temperature hits the upper bound, flexibility is only possible downwards, and the other way around as the room temperature equals the lower bound.




(a) Nominal power (filled line) and the upper and lower bounds on alternative power trajectories (dashed lines).




(b) Room temperature, set temperature and temperature constraints.

Figure 2: Outline to principle of TCLs used for demand side flexibility. Dashed lines in a) represents the boundaries of possible deviation from the nominal power consumption while the temperature is that of the red curve in b).

Using the notation of [4], each heat pump $k$ in a collection can accept perturbations $e^k(t)$ around its nominal power consumption. The flexibility of each heat pump $k$ is then defined by the set of power signals

$$\mathbb{E}^k = \left\{ e^k(t) \middle| \begin{array}{l} 0 \leq P_o^k + e^k(t) \leq P_m^k, \\ P_o^k + e^k(t) \text{ keeps } |\theta^k(t) - \theta_r^k| \leq \Delta^k \end{array} \right\}, \tag{1}$$

where $P_m^k$, $\theta_k(t)$ and $\theta_r^k$ are the maximum power consumption, inside temperature and set reference temperature for device $k$, respectively [4]. $\Delta^k$ is half the width of the temperature dead-band that has to be respected, as seen in figure 2b). The aggregate flexibility for a collection is then

$$\mathbb{U} = \sum_k \mathbb{E}^k. \tag{2}$$

## 2.3 Neural Networks

Artificial Neural Networks, ANNs for short, is a collection of simple processing units connected to each other in stacked layers [9]. An example of a simple feed-forward ANN to illustrate the idea behind this is displayed in figure 3.

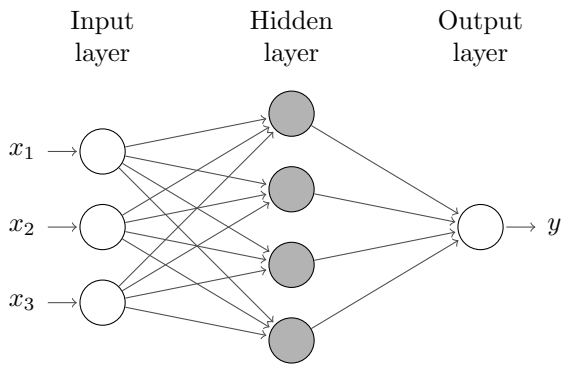



Figure 3: Illustration of a simple feed-forward ANN.

Each node performs a weighted summation of its inputs, passes this through an activation function which in turn computes the output of the node. Mathematically, this is formulated as

$$a = \sum_{k=1}^{K} w_k x_k + w_0 \tag{3}$$

$$y = \phi(a), \tag{4}$$

where $y$ is the output and $\phi$ the activation function. An example of a node with two inputs is displayed in figure 4 below. There is a large variety of

activation functions, and the most suitable often depends on the problem, the architecture and the problems that may arise when training the network. For regression however, common choices of activation functions for the output node are a simple linear function,

$$f(x) = x, \tag{5}$$

or a Rectified Linear Unit (ReLU) activation function,

$$f(x) = \max(0, x) \tag{6}$$

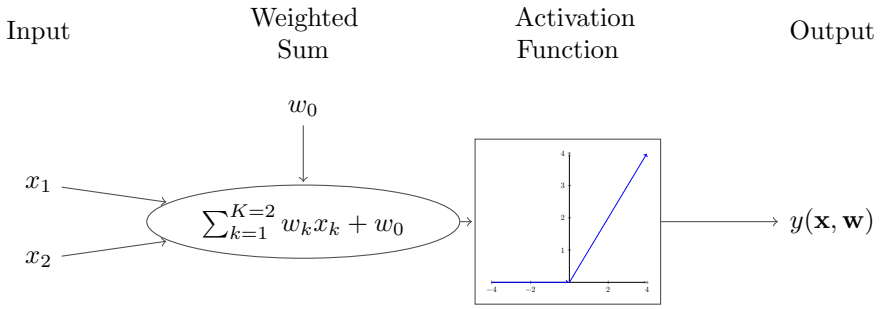



Figure 4: The basis of a Neural Network, a node, with a ReLU-activation function. The output is a function of the inputs $\mathbf{x}$ and weights $\mathbf{w}$.

### 2.3.1 Universal Approximation Theorem

The universal approximation theorem [1] states that a standard multilayer feed-forward network with as few as one hidden layer and a squashing function (e.g. logistic or tanh) as activation can approximate any Borel measurable function from a finite dimensional space to another with any desired non-zero error provided that the network is given sufficient amount of weights. Continous functions on closed and bounded subsets of $\mathbb{R}^n$ are Borel measurable [5] and that suffices for this thesis. The meaning of the theorem is thus that a network with enough weights can learn any nonlinear function. However, the theorem does not provide any guarantee that the sought function will be found during training, and there are two reasons for this [5]. Either, the optimization fails and does not find the correct parameter values for the desired function, or the training may cause overfitting resulting in the wrong function.

## 2.4 Loss Function and Stochastic Gradient Descent

In order to train an ANN from data, an objective function, in this domain often called loss function, of some sort is minimised. For regression problems it is natural to have a loss based either on the mean squared error, MSE for short,

$$E(\mathbf{w}) = \frac{1}{N} \sum_{k=1}^{N} (y_n(\mathbf{w}, \mathbf{x}) - d_n)^2 \tag{7}$$

or the mean absolute error

$$E(\mathbf{w}) = \frac{1}{N}\sum_{k=1}^{N}|y_n(\mathbf{w},\mathbf{x}) - d_n|, \tag{8}$$

where $d_n$ is target for sample n. While for classification problems, common loss functions used are instead the cross-entropy error

$$E(\mathbf{w}) = -\frac{1}{N}\sum_{n}^{N}\left(d_n \ln y_n(\mathbf{w},\mathbf{x}) + (1-d_n)\ln\left(1-y_n(\mathbf{w},\mathbf{x})\right)\right) \tag{9}$$

for binary classification, or

$$E(\mathbf{w}) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{c} d_{ni}\ln y_{ni} \tag{10}$$

for multi-class classification [9], with index $i$ denoting the element of a one-hot-encoded target vector $\mathbf{d_n}$ and $c$ are the number of classes.

The loss function is then minimzed w.r.t. the weights $\mathbf{w}$ using some optimisation technique. Usually some extension of the stochastic gradient descent optimisation algorithm is used [5], which employs the fact that a function decreases fastest along the direction of the negative gradient. For the vanilla gradient descent, using the MSE as loss funcition, the weights $\mathbf{w}$ are updated by iterations on the form

$$\mathbf{w}^{i+1} = \mathbf{w}^i - \eta\nabla_{\mathbf{w}}E = \mathbf{w}^i - \frac{\eta}{N}\sum_{k=1}^{N}\nabla_{\mathbf{w}}(y_n(\mathbf{w},\mathbf{x}) - d_n)^2. \tag{11}$$

As the computational cost of the gradient operation is $O(N)$, this becomes too computationally cumbersome as the the dataset grows and the time it takes to take a single gradient step makes this algorithm infeasible [5]. Stochastic Gradient Descent instead forms an estimate using a small set of samples. On each step of the algorithm a minibatch of $N'$ samples is drawn uniformly from the training set and the gradient is approximated as

$$\nabla_{\mathbf{w}}E \approx \frac{1}{N'}\nabla_{\mathbf{w}}\sum_{i=1}^{N'}E_n. \tag{12}$$

Numerically computing an analytical expression for the gradient of the loss function w.r.t. the weights, $\nabla_{\mathbf{w}}E$, can be computationally expensive, especially for a network with a large amount of weights [5]. Therefore, an algorithm named back-propagation is used to calculate the gradients for each input-output pair. Back-propagation uses the chain-rule for derivatives in a smart manner so that already computed derivatives can be re-used and thus avoids redundant

calculations. For a neural network with only one hidden layer, such as the one in figure 3, two sets of nodes need to be updated in order to take a step with gradient descent [9],

$$\text{input-to-hidden weights: } \Delta\tilde{w}_{jk} = -\eta\frac{\partial E}{\partial\tilde{w}_{jk}}$$

$$\text{input-to-output weights: } \Delta w_{ij} = -\eta\frac{\partial E}{\partial w_{ij}},$$

where the indices denotes input $k$, hidden node $j$ and output $i$, respectively. For example $w_{jk}$ denotes the weight between input $k$ and hidden node $j$. The arguments to each node for a specific data sample $n$ can then expressed as

$$a_{ni} = \sum_j w_{ij}h_{nj}$$

for output node i and

$$\tilde{a}_{nj} = \sum_k \tilde{w}_{jk}x_{nk}$$

for hidden node j. The partial derivatives of the loss function $E$ w.r.t. the weights $w_{ij}$ and $w_{jk}$ for a specific data sample $n$ are then calculated as

$$\frac{\partial E_n}{\partial w_{ij}} = \frac{\partial E}{\partial y_{ni}}\frac{\partial y_{ni}}{\partial w_{ij}} = \frac{\partial E}{\partial y_{ni}}\frac{\partial y_{ni}}{\partial a_{ni}}\frac{\partial a_{ni}}{\partial w_{ij}} = \frac{\partial E}{\partial y_{ni}}\phi_o'(a_{ni})h_{nj} \tag{13}$$

and

$$\begin{aligned}\frac{\partial E_n}{\partial\tilde{w}_{jk}} &= \sum_i \frac{\partial E_n}{\partial y_{ni}}\frac{\partial y_{ni}}{\partial\tilde{w}_{jk}} = \sum_i \frac{\partial E_n}{\partial y_{ni}}\frac{\partial y_{ni}}{\partial a_{ni}}\frac{\partial a_{ni}}{\partial\tilde{w}_{jk}} = \\ &\sum_i \frac{\partial E_n}{\partial y_{ni}}\frac{\partial y_{ni}}{\partial a_{ni}}\frac{\partial a_{ni}}{\partial h_{nj}}\frac{\partial h_{nj}}{\partial\tilde{w}_{jk}} = \sum_i \left(\frac{\partial E_n}{\partial y_{ni}}\phi_o'(a_{ni})w_{ij}\right)\phi_h'(\tilde{a}_{nj})x_{nk}.\end{aligned} \tag{14}$$

Note that the first two factors are the same, and this is what gives the algorithm its name, already computed derivatives are passed backwards through the network [9]. The weights updates are at last computed as

$$\Delta w_{ij} = -\eta\sum_n \frac{\partial E}{\partial y_{ni}}\phi_o'(a_{ni})h_{nj} \tag{15}$$

and

$$\Delta\tilde{w}_{jk} = -\eta\sum_n\sum_i \left(\frac{\partial E_n}{\partial y_{ni}}\phi_o'(a_{ni})w_{ij}\right)\phi_h'(\tilde{a}_{nj})x_{nk}. \tag{16}$$

## 2.5 Recurrent Neural Networks

Recurrent Neural Network is a special architecture of neural networks, designed to handle temporal dependencies in data [9]. These differ from ordinary neural networks in that they include feedback connections between the nodes. A recurrent neural network with one recurrent node as hidden layer is displayed in figure 5. The feedback connection doesn't necessarily have to be between hidden units, but can also be e.g. from output to hidden unit.

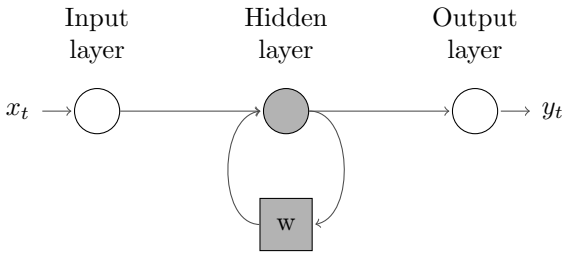



Figure 5: Recurrent neural network with one hidden layer.

Unfolded in time, the recurrent network in figure 5 takes the form of the graph shown in figure 6 below.

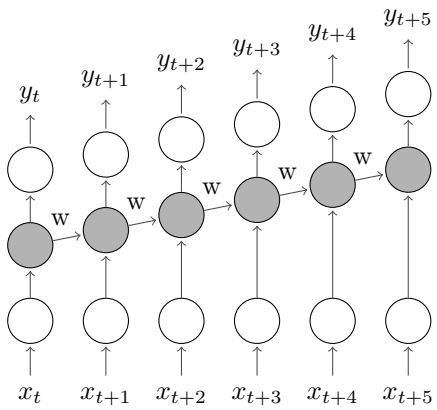



Figure 6: Recurrent neural network with one hidden layer unfolded in time.

### 2.5.1 Back-Propagation Through Time

With the computational graph of the recurrent network unfolded in time as in figure 6, the network takes the form of a regular neural network and can be trained in the same way using back-propagation.

A drawback with recurrent neural networks is that when performing back-propagation the passing of derivatives between the hidden nodes gives rise to

factors of the kind [9]

$$\frac{\partial h_{t+k}}{\partial h_t} = \frac{\partial h_t t + k}{\partial h_{t+k_1}} \frac{\partial h_{t+k-1}}{\partial h_{t+k-2}} \cdots \frac{\partial h_{t+1}}{\partial h_t}. \tag{17}$$

For a network with a long input sequence multiplication of many hidden node gradients will occur, and numerical issues arise as these factors are either small or large leading to the weight updates considered in the prior section to either vanish or explode.

### 2.5.2 Long Short-Term Memory Networks

Long Short-Term Memory (LSTM) Networks is a type of recurrent neural network architecture designed to overcome the issue with vanishing and exploding gradients (introduced in 1997 in [2]). Instead of a simple recurrent node as in figures 5 and 6, this is replaced by an LSTM unit, seen in figure 7.

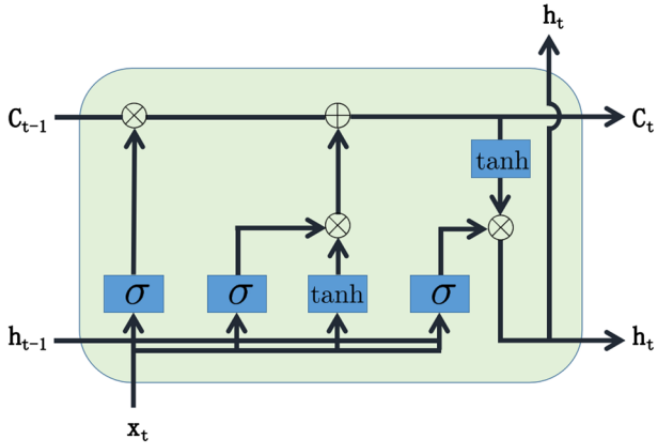



Figure 7: A LSTM unit as used in the hidden layers. $x_t$, $h_t$ and $c_t$ are input, output and hidden state, respectively. Taken from [7, p.32276]

Apart from the the input and output, $x_t$ and $h_t$, the network also contains the hidden state $c_t$ which acts as an internal memory of the node[9]. There are several gates inside the LSTM unit. The gates with the logistic function as activation are from left to right, the forget gate $f_t$, the input gate $i_t$ and the output gate $o_t$. The tanh gate to left in figure 7 is used to compute a candidate value for the hidden state, $\tilde{c}_t$. These gates are computed as

$$f_t = \sigma(x_t u^f + h_{t-1} u^f), \tag{18}$$

$$i_t = \sigma(x_t u^i + h_{t-1} w^i), \tag{19}$$

$$o_t = \sigma(x_t u^o + h_{t-1} w^o) \tag{20}$$

and

$$\tilde{c}_t = \tanh\left(x_t u^c + h_{t-1} w^c\right), \tag{21}$$

where $u$ and $w$ are the input and prior output weights. The new hidden state is computed as the combination of the prior hidden state $c_{t-1}$ and the new candidate hidden value $\tilde{c}_t$, each weighted by the forget gate value $f_t$ and the input gate value $i_t$,

$$c_t = c_{t-1} f_t + \tilde{c}_t i_t. \tag{22}$$

At last the output $h_t$ is computed by applying a tanh on the hidden value and then weighted with the value of the output gate $o_t$,

$$h_t = \tanh\left(c_t o_t\right). \tag{23}$$

The mathematical argument to why the gradient updates do not vanish or blow up is rather cumbersome and will be omitted here. The curious reader may resort to e.g. [6].

# 3 Methodology

The approach was to create LSTM network architectures to form predictions of the power consumption of individual heat pumps. This should allow the networks to catch any individual dynamics of the heat pumps, thus avoiding time consuming manual model order determination and parameter tuning, which may be the case if employing e.g. SARIMA-models. Remembering UAT, this is possible in theory. On-boarding of new devices to any aggregated model would be easy. However, because of the flexibility of neural networks, models may easily overfit resulting in poor generalisation.

To answer the first question in the problem formulations "Can a Recurrent Neural Network improve predictions of heat pump power consumption over current industry baseline methods?", a baseline and a RNN model is needed. We introduce two naive predictors in 3.5.2, 3.5.3, and three model architectures in 3.4.1, 3.4.1 and 3.4.3.

Separate models were trained for all pumps, these were then be aggregated to evaluate the total power consumption of the system, which was the quantity to be predicted. Creating separate models for each pump does impose some practical limitations however. Due to the behaviour of each pump not being identical, the optimal model architecture and parameters might not be the same for all pumps. This thesis will use a single architecture for all pumps, that works well in aggregate, to limit scope.

The naive predictor will be the power consumption at the same time 2 days before. Due to the 24 hour seasonality of the data this will be a decent predictor.

In general when tackling problems with neural networks, the methodology consists mostly of trial and error, as there are no physical interpretation of the models. An ansatz for a simple model is made which is then built upon. The performance metrics used is the mean squared error, MSE.

The following Python libraries were used for data handling and creation of the predictive models:

- keras
- pandas
- numpy

## 3.1 Data

The data was provided by Emulate Energy, and were recordings from real heat pumps connected to Emulate's API. Data between the 2022-01-04 and 2022-03-28 was used in the final models. This period of time was chosen due to that there is only a heating demand, and thus a demand for power flexibility during winter, and because there were a fair amount of devices (22) connected to the API during this time period.

The data was divided into three sets: train, validation, and test. The train set is the data on which the networks are optimised. During training the algorithm evaluates the performance of the network on the validation set using a given metric, the MSE. After training model performance can be evaluated on the test set, which contains values independent from those trained on. The test set consists of the last 20 % of the data set. Validation size is specified for each model in tables 12, 16, 14.

Data was sampled at two sampling rates, once every minute and once every five minutes. The attributes deemed to be of most interest in the data series is shown in table 1, see Appendix A for all attributes.

Table 1: Attributes of interest in data recordings.

| **Attribute** | **Description** |
|---|---|
| `added_power` | Thermal element power |
| `out_temp` | Outside ambient air temperature |
| `compr_freq` | Compressor frequency |
| `compressor_state` | Compressor state |
| `heat_offset` | Setting deciding heat medium temperature |
| `time` | Date-time in UTC |

The power consumption of the heat pumps consist of the electrical power consumed by the compressor, represented in either `compr_freq` or `compressor_state` and the power consumed by the thermal element that starts to heat when the compressor is not sufficient to meet the heating demand, represented in `added_power`. Whether `compr_freq` or `compressor_state` is present in the heat pump data depends of what `product`, i.e. model of heat pump, the heat pump is. Both `added_power` and `compr_freq` attributes are continous, while `compressor_state` is categorical and can belong to the categories {20 (= Stopped), 40 (= Starting), 60 (= Running), 40 (= Stopping)}. The compressor power `P_compr` for devices with `compressor_state` is then computed as

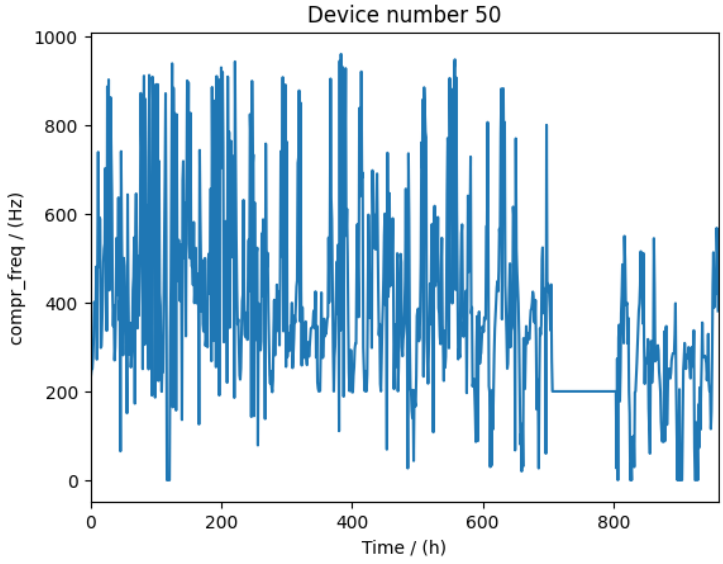
**Algorithm 1** `P_compr` for devices with `compressor_state`

**if** `compressor_state` $= 60$ **then**
  `P_compr` $=$ `P_rated` / `COP`
**else if** `compressor_state` $\in [40, 100]$ **then**
  `P_compr` $=$ `P_rated` / `COP` $/ 2$
**else**
  `P_compr` $= 0$
**end if**

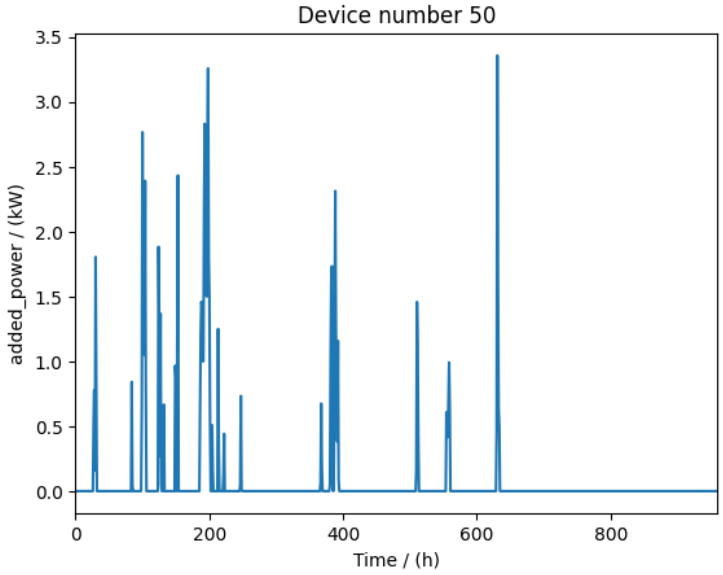
where the device parameters `P_rated` and `COP` are the rated heat of the compressor and coefficient of performance, respectively. `P_compr` for devices with `compr_freq` is simply

$$\texttt{P\_compr} = \texttt{compr\_freq} \,/\, \texttt{max\_freq} \cdot \texttt{P\_rated} \,/\, \texttt{COP} \tag{24}$$

Figure 8 and figure 9 shows the `added_power` and `compr_freq` or `compressor_state` for two different devices. The `compr_freq` attribute is deemed to be rather stationary, as can be seen in figure 8, and Augmented Dickey-Fuller tests also confirmed this. `compr_freq` and `compressor_state` are somewhat similar throughout the different devices. `added_power` however, has a more erratic behaviour, which also varies very much between different devices.

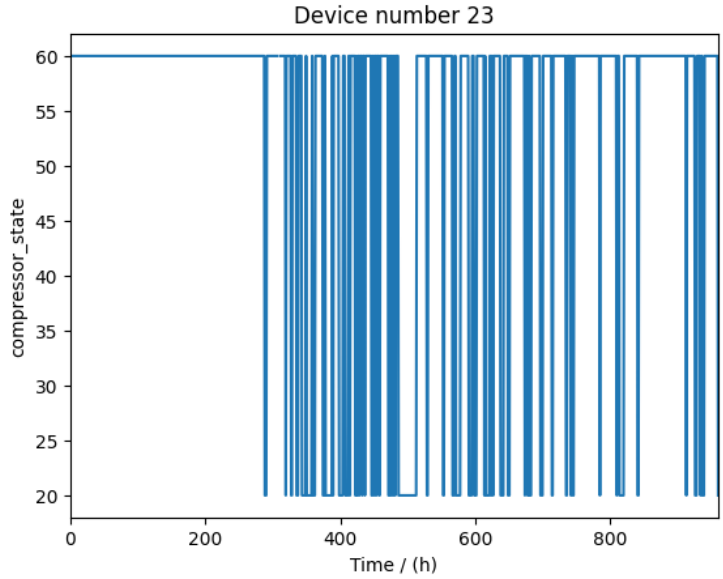



(a) `compr_freq`

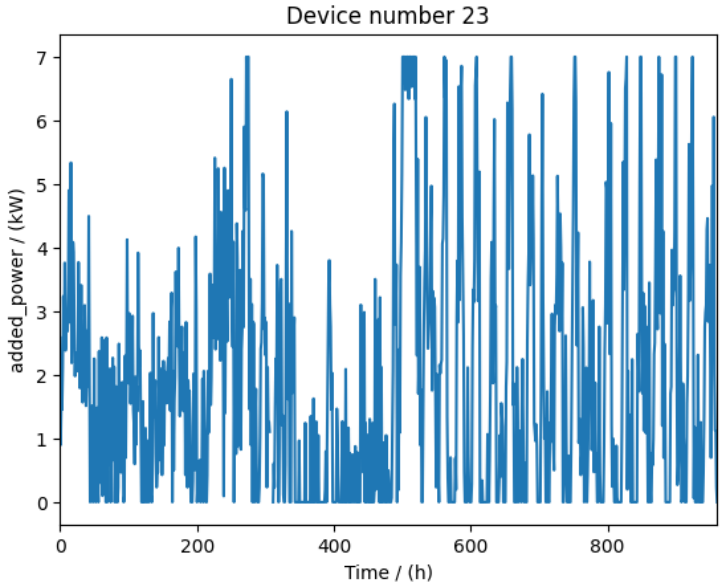



(b) `added_power`

Figure 8: Samples of attributes `compr_freq` and `added_power` from device with id 50 during a time period of ∼40 days

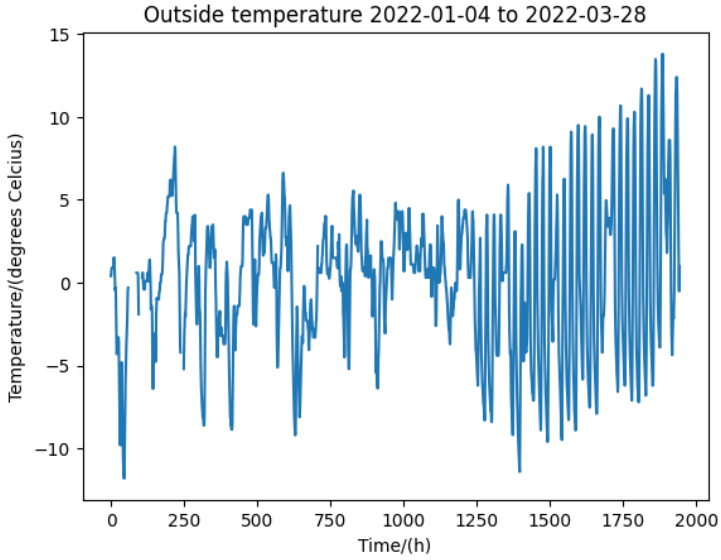



(a) `compressor_state`

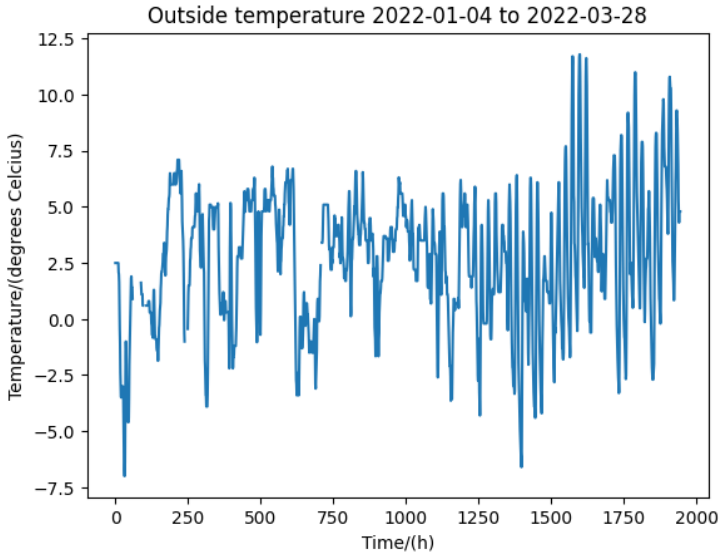



(b) `added_power`

Figure 9: Samples of attributes `compressor_state` and `added_power` from device with id 23 during a time perdod of $\sim$ 40 days.




(a) Device number 16.




(b) Device number 36.

Figure 10: `out_temp` recordings for two different devices

A sample of recorded `out_temp` for the time period considered is seen in figure 10. The temperature data exhibits only a small trend at the end of the dataset.

Using the attribute `time`, which was only a timestamp in the original data, the additional time attributes `sinMonth`, `cosMonth`, `sinDay`, `cosDay`, `sinHour` and `cosHouR` were created to use for inputs to the models. These were calculated as

$$\texttt{sinMonth} = \sin\left(2\pi\frac{\texttt{month}}{12}\right), \tag{25}$$

$$\texttt{sinDay} = \sin\left(2\pi\frac{\texttt{weekday}}{7}\right) \tag{26}$$

and

$$\texttt{sinHour} = \sin\left(2\pi\frac{\texttt{hour}}{24}\right) \tag{27}$$

The cosine attributes were calculated similarly. These were originally created to be used for any possible machine learning algorithm using distances between samples, which LSTMs don't, but were then kept throughout the project.

When training the models, `out_temp` and `heat_offset`, were assumed to be known up until the time of the prediction, e.g. $t_0 + 36$, where $t_0$ is the present time. The reason for this was that if the models were to be put to use in real time, the idea is to input forecasts of the temperature, while the `heat_offset` could just be kept to the desired setting. The other attributes were not assumed to be known ahead in time, i.e. only up until $t_0$.

## 3.2 Pre-processing

The recorded data first had to processed from the raw data output of the Emulate API into a useful form for a data-driven model. Some important points here were:

- The data needed to be consistent time-wise, i.e. missing data points had to be treated. In the final models, this was simply handed by omitting sequences containing missing data. See algorithm 2.
- For the purpose of aggregating predictions for all pumps, the time-series had to be synchronised in time.

### 3.2.1 Data set generation

After averaging features into hourly chunks, time series for each feature at an hourly resolution were available. These then needed to be divided into the tensor form required for an LSTM: (`no_samples`, `no_timesteps`, `no_features`). Algorithm 2 was ran.
Note that `feature_series` has dimensions (`no_samples`, `no_features`). Algorithm 2 results in a data set where historical data for `no_timesteps` hours past is provided, additional future data in the form of `out_temp` is provided. This will simulate weather forecasts. The final dimensions of `X` will thus be `(no_samples`$_2$`, no_timesteps, no_features+1)`. Where `no_samples`$_2$ is some integer smaller than `no_samples`, as sequences containing `NaN`:s are ignored.

**Algorithm 2** Creation of data set `X`

```
for idx in 0:(len(feature_series) - no_timesteps do
  sub_series = feature_series[idx:idx+2*no_timesteps,:]
  if sub_series contains NaN then
    continue
  else
    X.add(concat(sub_series[0:no_timesteps, :],
sub_series[no_timesteps:end, out_temp]))
  end if
end for
```

### 3.2.2 Scaling

To train an LSTM Network it is neccessary to scale the inputs such that they are somewhat in the same value range. This is to ensure that the model trains properly and to speed up training. Two forms of scaling were used: standard scaling and min-max scaling. Min-max scaling takes the values in a vector and applies the transformation

$$\tilde{x}_i = -1 + \frac{2(x_i - \min(\mathbf{x}))}{\max(\mathbf{x}) - \min(\mathbf{x})}, \tag{28}$$

which results in a feature range of $[-1, 1]$. Standard scaling is done using

$$\tilde{x}_i = \frac{x_i - \bar{x}}{s}, \tag{29}$$

where $\bar{x}$ is the sample mean, and $s$ is the sample standard deviation. In general standard scaling is preferred as it retains sensitivity to outlier values. It does however assume that the the samples are reasonably Gaussian distributed. Min-max scaling is applied to features with discrete values, such as the `heat_offset` and `compressor_state` values.

### 3.2.3 Feature selection

First, an initial set of features must be decided. Using domain knowledge about heat pumps four initial features were chosen

- `out_temp`
- `heat_offset`
- `sinHour` and `cosHour`

## 3.3 Loss functions

The loss function used to train the models is the MSE for all models, except the sub-model in Model 3 (see section 3.4.2) that makes a categorical output for the attribute `compressor_state`. The MSE is usually used in prediction problems as it punishes large deviations more than the MAE. The categorical output sub-model instead uses the categorical cross-entropy as the loss function.

## 3.4 Model architectures

Three different models, described below, were created and compared. The initial idea behind these were to create different models for the power that stems from the compressor and heating element, respectively, since the power of these exhibited quite different behaviour (see figures 8 and 9). In the model overviews in sections 3.4.1-3.4.2, what is referred to as a LSTM model is a LSTM Network similar to that of figure 11. The inputs to theses networks are tensors with the dimension (Number of samples, Time steps, Number of Attributes). The outputs from the output layer is a scalar.

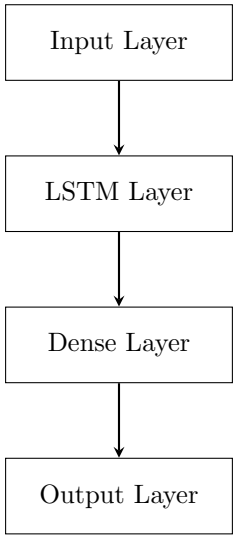



Figure 11: LSTM Network architecture used for the predictive models. The dense layer is a regular feed-forward layer as described in section 2.3.

### 3.4.1 Model 1

Combination of two separate models trained to predict power contribution from compressor and added power, respectively. Schematic overview displayed in figure 12.

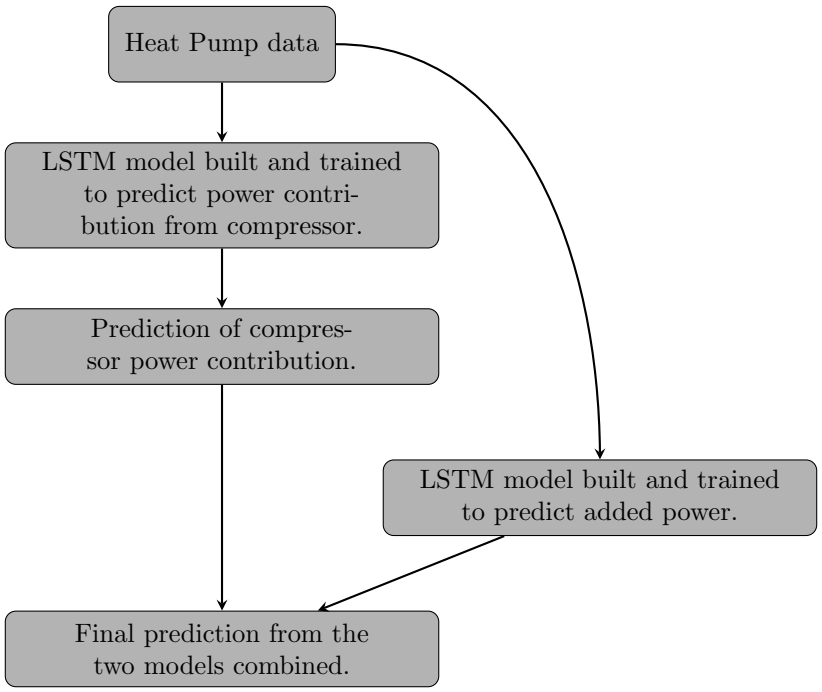



Figure 12: Model 1 architecture.

### 3.4.2 Model 2

Combination of two separate models where heat pump models containing compressor frequency were trained as model 1, and a classifier was created for heat pump models containing compressor state predicting the categorical compressor state. Added power was predicted as before. Schematic overview is seen in figure 13. This model differs from the other two as it is only configured to output scalars, due to the categorical output.

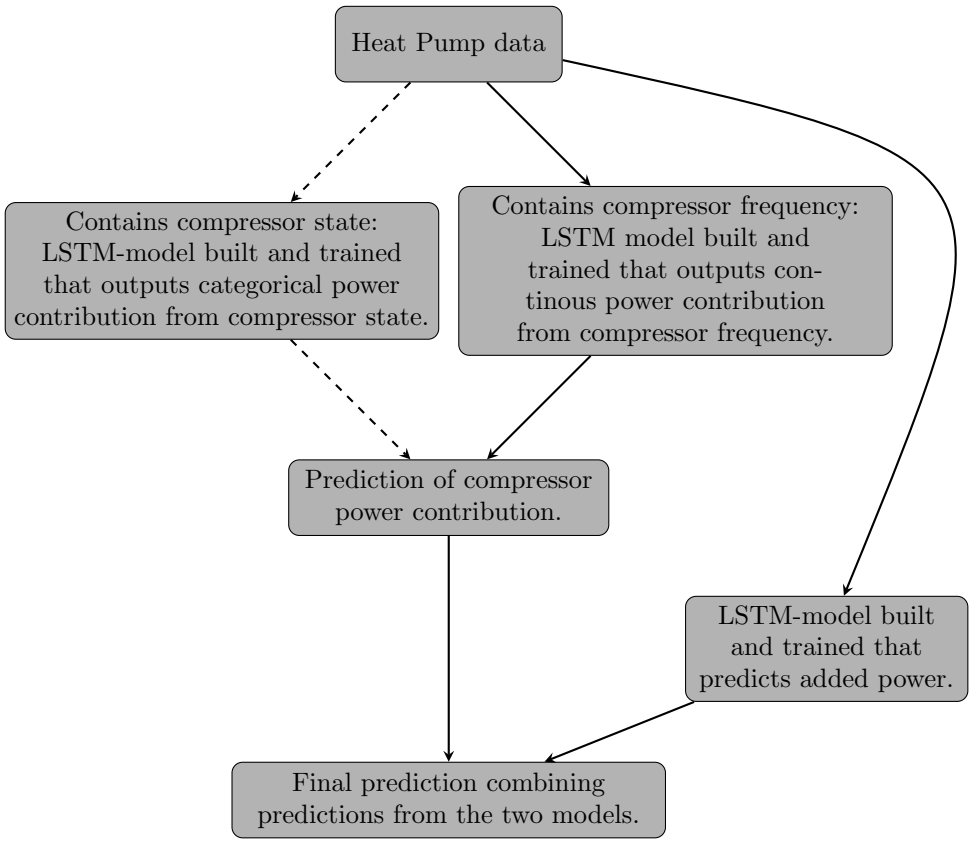



Figure 13: Schematic overview of model architecture 2.

### 3.4.3 Model 3

LSTM-model simply trained on the total electrical power for each heat pump as shown in figure 14

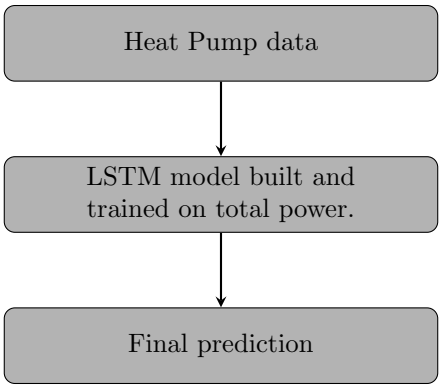



Figure 14: Model 3 architecture.

## 3.5 Aggregation of Models

The model performance of interest is not on individual level, but rather for all models in aggregate. Therefore, before performance evaluation all predictions were summed up and compared to the real power consumed by all pumps, as displayed in figure 15

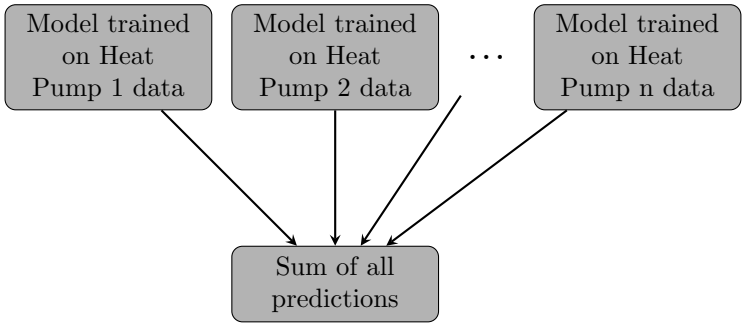



Figure 15: Aggregation of individual forecast models.

### 3.5.1 Model evaluation and validation

For evaluating the models some baseline was needed. Currently, an average of past power consumption is often used as an indicator for future events, and therefore it is of interest to compare the created models in this thesis to these baselines. Two different naive predictors were used.

### 3.5.2 48-hour Naive Predictor

The 48-hour naive predictors simply use the power consumption from 24 hours or 48 hours earlier as a prediction, namely,

$$\hat{y}_{48}(t) = y(t - 48). \tag{30}$$

### 3.5.3 3-day average predictor

A choice for a less noisy sensitive naive predictor is taking the hourly average over three days i.e.

$$\hat{y}(t) = \frac{1}{3}(y(t - 96) + y(t - 72) + y(t - 48)) \tag{31}$$

## 3.6 Hyperparameter tuning

When investigating model parameters a rough grid search sweep is conducted first. The performance is then evaluated on the mse on the test set. A new grid of parameters is then chosen depending on what parameters alter model performance in a significant way. I.e. if altering the batch size does not change the MSE in a significant way, then the range of that parameter is reduced, and vice versa. These parameter spaces will be presented in appendix D.

An early stopping policy was used. As several networks were trained for each model this allows some additional flexibility. E.g. the max training epochs is set to 1000 for all pumps. Some pumps may require more epochs to converge while others require fewer. With an early stopping policy the network will stop training after $n$ epochs offer no improvement on the MSE of the validation set. This also reduces overfitting.

# 4 Results

An overview of model performances is presented first, followed by individual performance for each model.

## 4.1 Model comparison

The final model performances are presented in table 2. Performance metrics for the neural network models created seen in 2 are means of the MSE of predictions 36-47 hours ahead. See sections 4.3-4.5 for more details.

Table 2: Error metrics all models.

| **Model** | **MSE** |
|---|---|
| Model 1 | 24.18 (mean) |
| Model 2 | 20.67 (mean) |
| Model 3 | 22.31 (mean) |
| Naive 48 | 34.02 |
| Naive 3avg | 31.32 |

## 4.2 Model Parameters

As described in 3.4, the inputs are on the form (`no_samples`, `no_timesteps`, `no_features`). To simplify the construction of data sets `no_timesteps` was set to the prediction horizon, i.e. if a prediction was to be made for 36 hours in the future `no_timesteps` would be set to that value. Through trial and error the model was found to be quite insensitive to the size of the dense layer, and the number of nodes in the LSTM layer. Adding more layers, dense or LSTMs, also did not seem yield any significant increase in performance. The chosen values for these are presented for each model in the following sections.

## 4.3 Model 1

Performance metrics of Model 1 are shown in table 3, which displays the MSE for predictions 36-47 hours ahead. The two columns to the left, MSE Compressor power and MSE Added power, presents the MSEs for each of the two sub-models of this model. MSE of the predictions for the sum of the prior two are displayed in the column MSE Total power. The parameters of the two sub-models, that predict compressor power and added power, respectively, are shown in table 12.

Plots of predictions 36 hours ahead are seen in figures 16 below. In 16a) and 16c) the predictions for the compressor power and added power, respectively, 36 hours ahead on the test set are seen. In figure 16e), the prediction of the total power is seen.

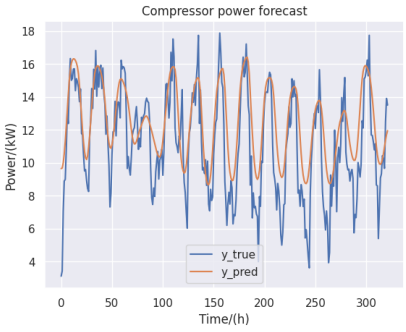
The right-hand side of figure 16 displays histograms of errors to each of the predictions on the left side. The errors are evaluated separately above and below the mean of the power consumption, this in order to be able to distinguish any difference in model performance when the nominal power is high or low.

Table 3: Mean squared errors for forecasts on test set with prediction horizons of 36-47 hours.

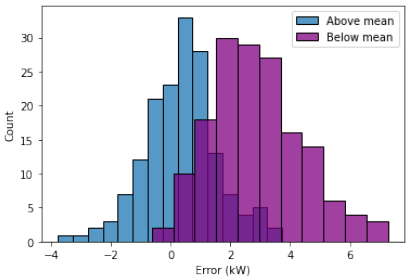
| **Hour** | **MSE Total power** | **MSE Compressor power** | **MSE Added power** |
|---|---|---|---|
| 36 | 29.82 | 5.55 | 16.47 |
| 37 | 29.88 | 6.06 | 15.95 |
| 38 | 27.14 | 4.85 | 14.82 |
| 39 | 26.49 | 4.61 | 15.24 |
| 40 | 24.00 | 3.99 | 13.88 |
| 41 | 21.38 | 3.96 | 12.85 |
| 42 | 21.57 | 4.00 | 12.11 |
| 43 | 23.44 | 3.86 | 13.57 |
| 44 | 21.23 | 3.73 | 12.71 |
| 45 | 21.60 | 3.97 | 12.35 |
| 46 | 21.18 | 4.03 | 12.13 |
| 47 | 22.37 | 4.59 | 11.85 |
| **Mean** | **24.18** | **4.43** | **13.66** |

Table 4: Model parameters used for model 1. See section 10.4 for input abbreviations.

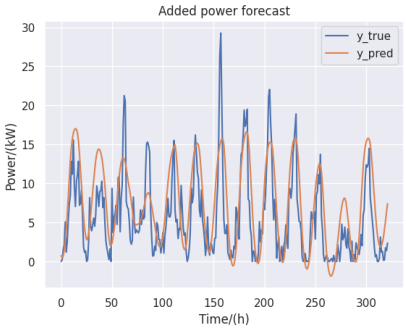
| | `compr_freq` **LSTM** | `added_power` **LSTM** |
|---|---|---|
| Dense Layers | 1 | 1 |
| LSTM Layers | 1 | 1 |
| Dense Nodes | 16 | 64 |
| LSTM Nodes | 64 | 64 |
| Hidden Activation | tanh | tanh |
| Validation split | 0.1 | 0.1 |
| Dropout rate | none | none |
| EarlyStopping Patience | 3 | 3 |
| Epochs | 800 | 800 |
| Batch size | 32 | 32 |
| LSTM Activation | tanh | tanh |
| Inputs | o, c | o, a |



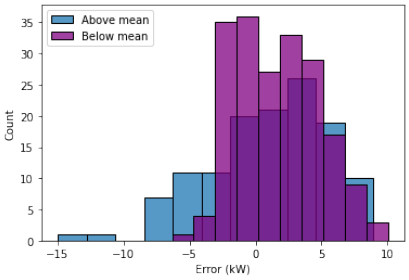

(a) 36 hours ahead forecast of compressor power on test set.



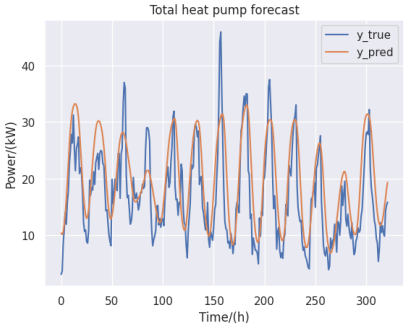

(b) Histogram of residuals for forecasts where the true values are above and below the true value mean. For compressor power consumption.



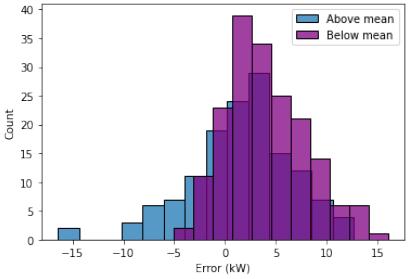

(c) 36 hours ahead forecast of added power on test set.




(d) Histogram of residuals for forecasts where the true values are above and below the true value mean. For thermal element power consumption.




(e) 36 hours ahead forecast of total power on test set.




(f) Histogram of residuals for forecasts where the true values are above and below the true value mean. For total consumption.

Figure 16: Model 1 predictions 36 hours ahead on test set.

## 4.4 Model 2

The resulting error metrics for model 2, for different prediction horizons are seen in table 15 below and the model parameters are shown in table 16, note here that there are instead of one sub-model predicting compressor power, there are two different kinds predicting either `compressor_state` or `compr_freq` depending what model the specific heat pump is.

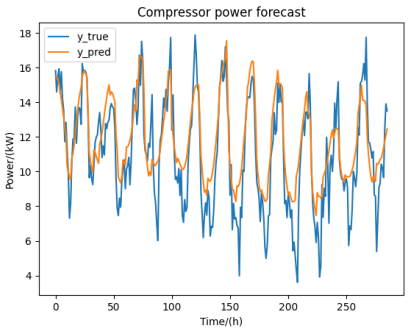
Plots of the 36-hour ahead predictions of the model on the test set and histograms of errors are seen in figure 17.

Table 5: Model 2 prediction results 36-47 hours ahead.

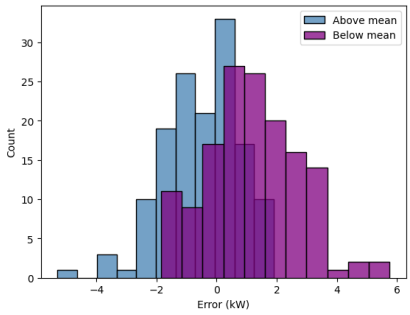
| **Hour** | **MSE Total power** | **MSE Compressor power** | **MSE Added power** |
|---|---|---|---|
| 36 | 26.35 | 2.83 | 21.51 |
| 37 | 25.67 | 2.88 | 20.62 |
| 38 | 26.02 | 3.24 | 20.64 |
| 39 | 30.14 | 3.19 | 22.27 |
| 40 | 30.31 | 3.28 | 22.28 |
| 41 | 24.30 | 2.53 | 21.07 |
| 42 | 29.79 | 2.91 | 24.89 |
| 43 | 29.70 | 2.91 | 24.48 |
| 44 | 33.48 | 3.21 | 26.29 |
| 45 | 30.56 | 2.85 | 24.39 |
| 46 | 33.82 | 3.08 | 27.21 |
| 47 | 36.84 | 5.18 | 25.80 |
| **Mean** | **29.75** | **3.17** | **23.45** |

Table 6: Model parameters used for model 2.

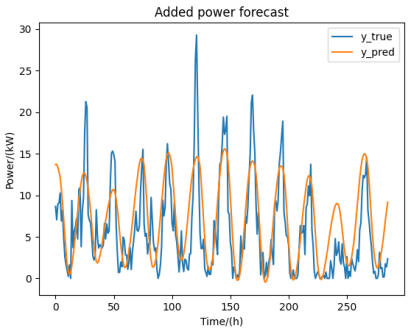
| | `compressor_state` **LSTM** | `compr_freq` **LSTM** | `added_power` **LSTM** |
|---|---|---|---|
| Dense Layers | 1 | 1 | 1 |
| LSTM Layers | 1 | 1 | 1 |
| Dense Nodes | 16 | 16 | 16 |
| LSTM Nodes | 16 | 16 | 16 |
| Hidden Activation | 16 | 16 | 16 |
| LSTM Activation | tanh | tanh | tanh |
| Validation split | 0.2 | 0.2 | 0.2 |
| Epochs | 2000 | 2000 | 2000 |
| Batch size | 64 | 64 | 64 |
| Dropout rate | 0.05 | 0.05 | 0.05 |
| EarlyStopping Patience | 30 | 30 | 30 |
| Inputs | o, c, h_o sinH,cosH | o,c,h_o sinH,cosH | o,a,h_o sinH,cosH |



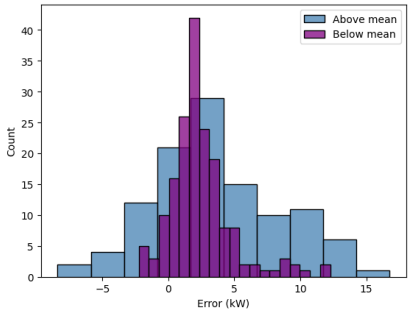

(a) 36 hours ahead forecast of compressor power on test set.



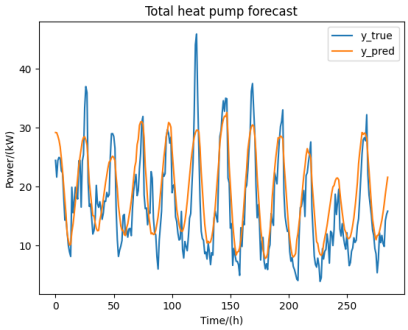

(b) Histogram of residuals for forecasts where the true values are above and below the true value mean. For compressor power consumption.



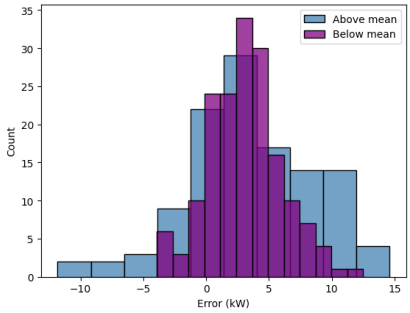

(c) 36 hours ahead forecast of added power on test set.




(d) Histogram of residuals for forecasts where the true values are above and below the true value mean. For thermal element power consumption.




(e) 36 hours ahead forecast of total power on test set.




(f) Histogram of residuals for forecasts where the true values are above and below the true value mean. For total consumption.

Figure 17: Model 2 predictions 36 hours ahead on test set.

## 4.5 Model 3

Results for model 3 are seen in table 7, and its parameters in table 7. Since in this model a LSTM Network was trained to predict the total power, table 14 contains only one column.

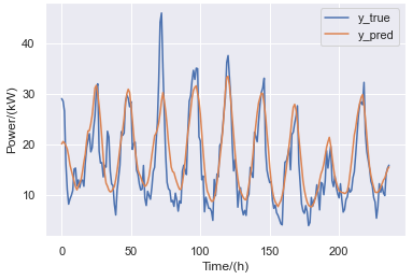
Table 7: Mean squared errors for forecasts on test set with prediction horizons of 36-47 hours.

| **Hour** | **MSE Total Power** |
|---|---|
| 36 | 31.08 |
| 37 | 26.91 |
| 38 | 25.72 |
| 39 | 24.49 |
| 40 | 21.51 |
| 41 | 21.38 |
| 42 | 21.15 |
| 43 | 18.73 |
| 44 | 19.68 |
| 45 | 18.85 |
| 46 | 19.50 |
| 47 | 18.75 |
| **Mean** | **22.31** |

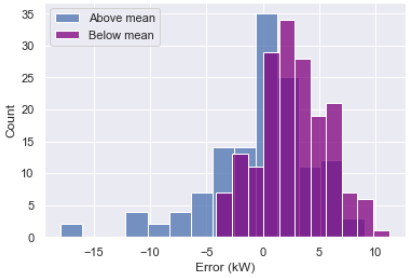
Table 8: Model parameters used for model 3.

| | `total_power` **LSTM** |
|---|---|
| Dense Layers | 1 |
| LSTM Layers | 1 |
| Dense Nodes | 64 |
| LSTM Nodes | 64 |
| Hidden Activation | tanh |
| LSTM Activation | tanh |
| Validation split | 0.1 |
| Epochs | 800 |
| Batch size | 32 |
| Inputs | o, t |

The plots of the 36 hour ahead predictions and the error histograms are seen in figure 18 below.




(a) Model 3 predictions on the test set.




(b) Histogram of total power residuals above and below mean of test set.

Figure 18: Model 3 predictions 36 hours ahead on test set. For total consumption.

## 4.6 Naive predictor

The performance metrics for the naive predictors are seen in table 9 below. The naive 48 hour predictor yielded an MSE of 34.02 $(kW)^2$ on the test data set, while the 3-day Average Naive Predictor yielded a MSE of 31.32 $(kW)^2$. Figure 19 displays the 3-day Average predictions on the test set.

Table 9: Performance metrics of naive predictors.

| **Predictor** | **MSE** |
|---|---|
| 48 Hour | 34.02 |
| 3-day Average | 31.32 |

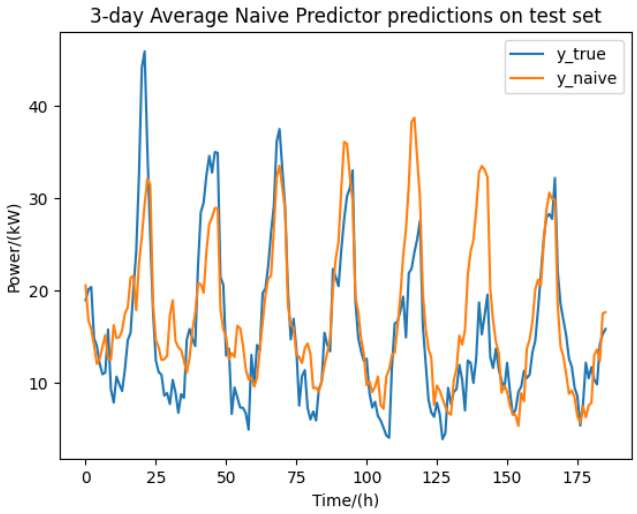



Figure 19: 3-day Average Naive Predictor predictions on the test set.

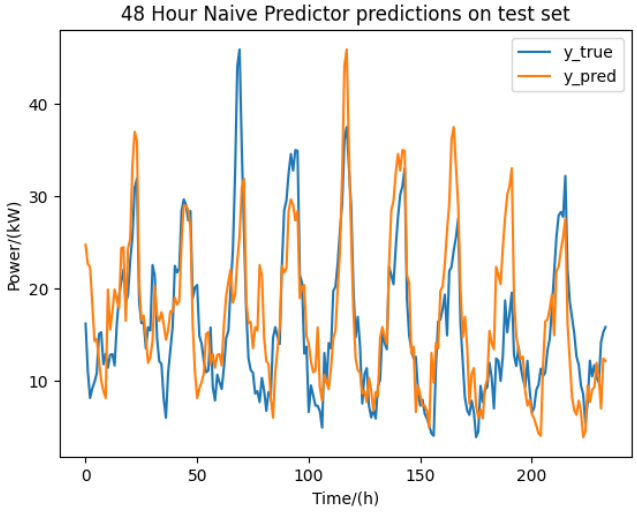



Figure 20: 48 Hour Naive Predictor predictions on the test set.

# 5 Discussion

The results indicate that a data driven machine learning approach can improve upon simple models based on historical data. In theory this should be expected, as providing a network that is well designed and given the same data, it should be able to be trained to perform the same operations as the naive predictor if those were optimal. Given that the problem formulation is specific to heat pumps it is also not unexpected that providing future temperature data will improve upon a pure history based model.

As tables 3 and 15 indicates, the error is larger for the power contribution from the `added_power`, compared to the error from the `compressor` and it is reasonable to believe that the erratic and varying behaviour of this part is more difficult to model. Therefore, an overall improvement could probably be made from improving the predictions for this part, either by further tuning hyperparameters and exploring size for a NN, or finding a more appropriate model of any other kind.

An aspect worth considering is that a deployed model would only use forecasts of outside temperature, which would imply further uncertainty in the predicting model. Therefore to fully be able to evaluate how well a predictive model performs, it probably has to be evaluated and compared to industry standards using actual forecasts of the outside temperature.

It was deemed unneccessary to handle non-stationarities or trend in the data since the time period was restricted to only one season. However, since the results are not comfortably beating the baselines, it could be interesting to explore if the model would improve if any possible trends were accounted for or to re-train and evaluate the models on smaller parts of the data.

Another approach could be to first aggregate the data from all pumps and then create a model. But since the pumps are in different locations with different outside temperatures, some way of aggregating the inputs would also be needed which would lead to a loss of information. Additionally the model would be less flexible in regard to adding or removing pumps from the system.

The amount of data was limited, which may limit the model performance and the certainty of possible conclusions. As more data over time periods of interest become available, larger training, validation and test sets can be obtained, and may enable more accurate models.

Given the scope of the thesis, enough model optimisation was not conducted to conclude that the given architectures and parameters were near optimal. It was also found that the behaviour of individual heat pumps could differ greatly. It is thus not unlikely that separate architectures for separate pumps could yield better results, but then the idea of not having to tune model parameters fails. However, the models were found to be quite insensitive to the parameters examined so the effectiveness of such an approach would have to be investigated.

Figures 16, 17, 18 show the appearance and error distributions of forecasts on the test set. The histograms suggest a bias in the forecasts since the errors don't seem to be distributed around zero, and this is the case for the errors both below and above the mean power consumption. This is especially significant for errors below the mean of the test set. Having a bias in the power predictions is undesirable since this may result in estimations of the flexiblity of a collection of heat pumps to be off. As models are trained to minimise MSE, the error of forecasts should be unbiased. However, considering the models are a sum of networks, bias could compound as models are summarised. Regardless, the bias seems to be systemic. Perhaps de-seasonalising the data could combat this.

# 6 Further Developments

We suggest three main ways of improving our model: feature engineering, alter model architecture, and hyperparameter tuning.

In theory, as the UAT states, our networks should be able to approximate an optimal function. But in order to help training convergence, feature engineering and additional pre-processing can prove valuable, especially with limited data. Concretely this could mean investigating de-trending, and de-seasonalising techniques.

Altering network architecture is another option. However, it was found during the course of this thesis that a small network with one hidden layer and one LSTM layer, such as those implemented in our final models yielded a lower MSE compared to architectures of larger size. With a change of input format and values as a result of feature engineering, this could change however.

Finally, as always with machine learning models more data is better. To quote Yngwie Malmsteen: "How can this be? How can less be more? More is More!". In essence, more data leads to a higher degree of certainty in what the network is trying to learn.

# 7 Conclusion

Remembering our initial problem formulation:

- Can a Recurrent Neural Network improve predictions of heat pump power consumption over current industry baseline methods?
- Given a model is created, how could it be improved further? And using what methods?

Answering the first question, the results indicate that this is the case for the chosen performance metric MSE.

It is difficult to reject or confirm whether an LSTM model could improve upon a naive predictor in the MSE metric, based on the results of the thesis. Worth noting, is that theoretically, an LSTM neural network should be able to perform the same linear operations as the naive predictor, and should with the additional input information yield results at least as good.

As the error distributions and test set prediction plots indicate, there seems to be a systemic overestimation during periods of low power consumption. More training data and further model optimisation would be a first step to improve the model. The models should also be tested with weather forecasts, as now they are trained with temperature observations. Testing the models with actual weather forecasts could introduce additional uncertainty.

# References


[1] Hornik K. and White H. "Multilayer Feedforward Networks are Universal Approximators". In: *Neural Networks* 2 (1989), p. 359–366. DOI: https://doi.org/10.1016/0893-6080(89)90020-8.

[2] Hochreiter S. and Schmidhuber J. "Long Short-Time Memory". In: *Neural Computation* 9.8 (1997), p. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[3] Schroeder D.V. *An Introduction to Thermal Physics*. Pearson, 2014. ISBN: 978-1-78434-339-2.

[4] Hao H. "Aggregate Flexibility of Thermostatically Controlled Loads". In: *IEEE Transactions on Power Systems* 30.1 (2015), p. 189–198. DOI: 10.1109/TPWRS.2014.2328865.

[5] Bengio I. Goodfellow Y. and Courville A. *Deep Learning*. Adaptive computation and machine learning series. MIT Press, 2017. ISBN: 9780262035613.

[6] Arbel N. *How LSTM Solve the Problem of Vanishing Gradients*. Last accessed 8 October 2022. 2018. URL: https://medium.datadriveninvestor.com/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577.

[7] Ma Z. and Sun Z. "Time-varying LSTM Networks for Action Recognition". In: *Multimedia Tools and Applications* 77 (2018), p. 32275–32285. DOI: https://doi.org/10.1007/s11042-018-6260-6.

[8] Svenska Kraftnät. *Information about Ancillary Services*. Last accessed 8 Spetember 2022. 2020. URL: https://www.svk.se/en/stakeholder-portal/electricity-market/information-about-ancillary-services/.

[9] Ohlsson M. and Edén P. *Lecture Notes on Introduction to Artificial Neural Networks and Deep Learning*. Faculty of Science, Lund Univeristy, 2020.

# 8 Appendix A - Data Attributes

Table 10: Attributes in data recordings.

| **Attribute** | **Description** |
|---|---|
| compr_freq | Compressor frequency |
| added_power | Thermal element power |
| out_temp | Outside ambient air temperature |
| compressor_state | Compressor state |
| heat_offset | Setting deciding heat medium temperature |
| time | Date-time in UTC |
| pump_speed_heat | Supply pump speed in % |
| condenser_out_temp | Condensor out temperature |
| return_temp | Return water temperature |
| dT_set_value | Set point delta T for heat medium flow |
| dT_current_value | Current value of the delta T for the heat medium flow |
| heat_medium_flow | Supply Temperature |
| room_temp_set_value | Set value room temperature |
| temporary_lux | Hours to set temporary lux |
| room_temp | Room temperature |
| heating_consumed_energy | Heat pump consumed energy due to due heating |
| hot_water_consumed_energy | Heat pump consumed energy due to hot water |
| ventilation_consumed_energy | Heat pump consumed energy due to ventilation |
| hot_water_comf_mode | Setting in heat pump menu |
| point_offset_out_temp | Outdoor temperature where heat curve offsets |
| point_offset | Amount of offset at the point offset temperature |
| min_supply_temp | Minimum supply temperature |
| max_supply_temp | Maximum supply temperature |

# 9 Appendix B - Device parameters

Table 11: Device parameters of the heat pumps.

| **Device Parameter** | **Description** |
|---|---|
| `product` | Model of heat pump |
| `home_location` | Coordinates to heat pump location |
| `P_rated` | Rated heat of compressor |
| `COP` | Coefficient of Power |
| `max_freq` | Maximum compressor frequency |

# 10 Appendix C - Model Architectures

## 10.1 Model 1

Table 12: Model parameters used for model 1. See section 10.4 for input abbreviations.

| | `compr_freq` **LSTM** | `added_power` **LSTM** |
|---|---|---|
| Dense Layers | 1 | 1 |
| LSTM Layers | 1 | 1 |
| Dense Nodes | 16 | 64 |
| LSTM Nodes | 64 | 64 |
| Hidden Activation | tanh | tanh |
| Validation split | 0.1 | 0.1 |
| Dropout rate | none | none |
| EarlyStopping Patience | 3 | 3 |
| Epochs | 800 | 800 |
| Batch size | 32 | 32 |
| LSTM Activation | tanh | tanh |
| Inputs | o, c | o, a |

## 10.2 Model 2

Table 13: Model parameters used for model 2.

| | `compressor_state` **LSTM** | `compr_freq` **LSTM** | `added_power` **LSTM** |
|---|---|---|---|
| Dense Layers | 1 | 1 | 1 |
| LSTM Layers | 1 | 1 | 1 |
| Dense Nodes | 16 | 16 | 16 |
| LSTM Nodes | 16 | 16 | 16 |
| Hidden Activation | 16 | 16 | 16 |
| LSTM Activation | tanh | tanh | tanh |
| Validation split | 0.2 | 0.2 | 0.2 |
| Epochs | 2000 | 2000 | 2000 |
| Batch size | 64 | 64 | 64 |
| Dropout rate | 0.05 | 0.05 | 0.05 |
| EarlyStopping Patience | 30 | 30 | 30 |
| Inputs | o, c, h_o sinH,cosH | o,c,h_o sinH,cosH | o,a,h_o sinH,cosH |

## 10.3 Model 3

Table 14: Model parameters used for model 3.

| | `total_power` **LSTM** |
|---|---|
| Dense Layers | 1 |
| LSTM Layers | 1 |
| Dense Nodes | 64 |
| LSTM Nodes | 64 |
| Hidden Activation | tanh |
| LSTM Activation | tanh |
| Validation split | 0.1 |
| Epochs | 800 |
| Batch size | 32 |
| Inputs | o, t |

## 10.4 Input Abbreviations

- o - `out_temp`
- t - `total_power`
- c - `compr_freq` or `compressor_state`
- sinH - `sinHour`
- cosH - `cosHour`
- h_o - `heat_offset`

# 11 Appendix D - Other Simulation Runs

Table 15: Model 2 prediction results 36-47 hours ahead.

| **Hour** | **MSE Total power** | **MSE Compressor power** | **MSE Added power** |
|---|---|---|---|
| 36 | 23.66 | 3.39 | 16.68 |
| 37 | 22.21 | 2.94 | 15.66 |
| 38 | 23.31 | 3.47 | 15.37 |
| 39 | 22.92 | 3.44 | 15.63 |
| 40 | 20.26 | 2.90 | 14.55 |
| 41 | 18.87 | 2.83 | 13.50 |
| 42 | 19.93 | 2.95 | 13.62 |
| 43 | 19.88 | 3.45 | 13.19 |
| 44 | 20.14 | 3.74 | 12.85 |
| 45 | 19.61 | 3.61 | 12.65 |
| 46 | 18.79 | 3.14 | 12.27 |
| 47 | 18.56 | 2.92 | 12.45 |
| **Mean** | **20.67** | **3.23** | **14.04** |

Table 16: Model parameters used for model 2.

| | `compressor_state` **LSTM** | `compr_freq` **LSTM** | `added_power` **LSTM** |
|---|---|---|---|
| Dense Layers | 1 | 1 | 1 |
| LSTM Layers | 1 | 1 | 1 |
| Dense Nodes | 16 | 16 | 64 |
| LSTM Nodes | 16 | 16 | 64 |
| Dense Activation | tanh | tanh | tanh |
| LSTM Activation | tanh | tanh | tanh |
| Validation split | 0.2 | 0.2 | 0.2 |
| Epochs | 2000 | 2000 | 800 |
| Batch size | 64 | 64 | 32 |
| Dropout rate | 0.1 | 0.1 | 0.1 |
| EarlyStopping Patience | 30 | 30 | 3 |
| Inputs | o, c,h_o sinH,cosH | o,c,h_o sinH,cosH | o,a |

**Lund University**
**Department of Automatic Control**
**Box 118**
**SE-221 00 Lund Sweden**

*Document name*
MASTER'S THESIS

*Date of issue*
February 2023

*Document Number*
TFRT-6191

*Author(s)*
Gustav Warnström
Johan Fant

*Supervisor*
Daria Madjidian, Emulate Energy AB, Sweden
Richard Pates, Dept. of Automatic Control, Lund University, Sweden
Emma Tegling, Dept. of Automatic Control, Lund University, Sweden (examiner)

*Title and subtitle*
Forecasting of Heat Pump Power Consumption using Neural Networks

*Abstract*

With the increase in electricity generation from renewable sources over recent years, the demand for ancillary services providing balancing support in the grid has risen. Demand side regulation can be performed by regulating consumption of some household devices, which can act as a virtual battery that can be loaded and unloaded into the grid. Among these devices are thermostatically controllable loads, e.g. heat pumps, AC:s and refrigerators. In order to create a virtual battery however, the nominal consumption of the load needs to be known, and in addition, market mechanisms requires this to be known ahead in time. This thesis has investigated predictive models using neural networks for thermostatically controllable loads in the form of heat pumps. The models were compared to two different naive predictors used as baselines. Results indicated that it is possible to beat these baselines, although it is difficult to fully evaluate the performance until put in real operation. Furthermore, it is unknown whether the presented models are optimal, and further work is likely required to find a more optimal model.

*Keywords*

*Classification system and/or index terms (if any)*

*Supplementary bibliographical information*

*ISSN and key title*
0280-5316

*ISBN*

*Language*
English

*Number of pages*
1-49

*Recipient's notes*

*Security classification*

http://www.control.lth.se/publications/