Master's Thesis

Detection Of High Energy Consuming Appliances' Load Profiles Using Non-Intrusive Load Monitoring

Rohit Kerekoppa Ramesha

Examiners: Prof. Dr. Hannah Bast, Prof. Dr.-Ing. Christof Wittwer

Advisers: Matthias Hertel, Benedikt Köpfer

University of Freiburg
Faculty of Engineering
Department of Computer Science
Chair for Algorithms and Data Structures

Fraunhofer ISE
Fraunhofer-Institut für Solare Energiesysteme
Freiburg

November 07th, 2022

Writing Period

02. 06. 2022 – 07. 11. 2022

Examiner

Prof. Dr. Hannah Bast

Second Examiner

Prof. Dr.-Ing. Christof Wittwer

Advisers

Matthias Hertel, Benedikt Köpfer

Declaration

I hereby declare that I am the sole author and composer of my thesis and that no other sources or learning aids, other than those listed, have been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work.
I hereby also declare that my Thesis has not been prepared for another examination or assignment, either wholly or excerpts thereof.

Place, Date	Signature

Abstract

Non-intrusive load monitoring (NILM) is the process of using the energy consumption of a house as a time-series, which is the sum of the consumptions of the individual appliances to predict the individual appliance's consumption time-series. The goal of this thesis is to predict the load profiles of high energy consuming appliances such as electric vehicle chargers and heat-pumps using the overall energy consumption time-series of a house. NILMTK is an open-source toolkit for comparative analysis of NILM algorithms across various datasets. Different deep learning algorithms are compared using NILMTK on both a synthetic and real dataset. The Seq2point algorithm outperformed the other deep learning algorithms in most houses of both datasets. Additional weather data improved the performance of the Seq2point and BERT models in predicting energy consumed by heat-pumps. This additional data also helped in improving the performance of the Seq2point algorithm in predicting energy consumed while charging an electric vehicle. Converting the BERT and Seq2point algorithm into a multi-output algorithm resulted in only a small decrease in performance. These multi-output models can be used in order to save time since one model can predict more than one appliance at a time.

Acknowledgments

The author would like to thank:

- Benedikt Köpfer for recommending the topic and for the constant support throughout my thesis.
- Matthias Hertel for the assistance and guidance he provided me throughout my thesis.
- Prof. Dr. Hannah Bast and Prof. Dr. Christof Wittwer for being my examiners and agreeing to review my thesis.
- Fraunhofer ISE for giving me this opportunity and providing with me the resources to complete the thesis.
- My girlfriend, friends and family for their emotional support.
- You, the reader, for your interest in this thesis.

Contents

List of Figures

List of Tables

1 Introduction

The war in Ukraine and a rebound from an economic slowdown during the COVID-19 pandemic have led to a sharp rise in energy prices and an energy crisis. The need for energy management has become more important than ever. Energy management includes planning and operation of energy production, energy consumption, energy distribution and energy storage. Energy monitoring is the process of tracking the energy consumption of either the individual appliances or the entire building/house. Energy monitoring is one of the most important aspects of energy management. There is a need to monitor energy consumption to plan and implement the technical measures needed to reduce energy consumption. Energy monitoring can also help end users to save energy by taking energy-saving measures. These include using energy-efficient devices, more efficient use of electrical equipment and eliminating unwanted energy activity. Energy monitoring not only helps end users to reduce their electricity bills, but is also an important step that is needed to reduce the emission of greenhouse gases and combat climate change.

In order to reduce the burden of the power sector by its major challenges like the cost of electricity, energy crisis and global warming, some of the critical inefficiencies of the sector can be spotted and removed using energy monitoring at a very low cost. There are great benefits if the patterns of appliance usage could be automatically detected to modify consumer habits (Darby et al. (2006)), with a potential reduction of 12%, depending on the type of feedback that is provided, as shown in Figure 1.

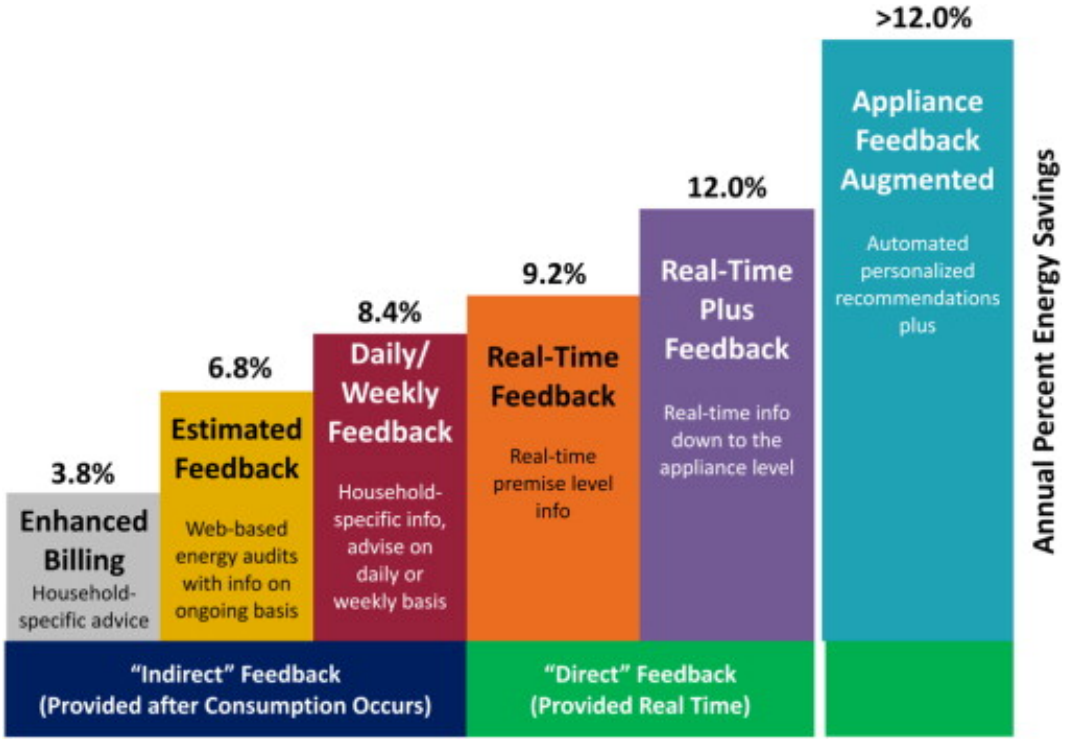

Figure 1: Residential savings due to energy consumption feedback. (taken from Armel et al. (2013)).

The two main ways of monitoring energy usage are Intrusive Load Monitoring (ILM) and Non-Intrusive Load Monitoring (NILM). ILM involves the installation of sensors at one or a few appliances in order to monitor the power consumed by them. Using these sensor readings, the energy consumption is monitored at the appliance level. NILM is the process of de-constructing the aggregate energy consumption time-series (main-meter readings) into its individual appliances as seen in Figure 2. This process does not require intrusion into the individual appliances to monitor their power consumption.

Figure 2 shows us that many appliances follow some characteristic patterns. NILM algorithms try to understand these underlying patterns and how it affects the overall consumption of the house. This is then used to predict the energy consumption of a particular appliance when only the overall consumption is known.

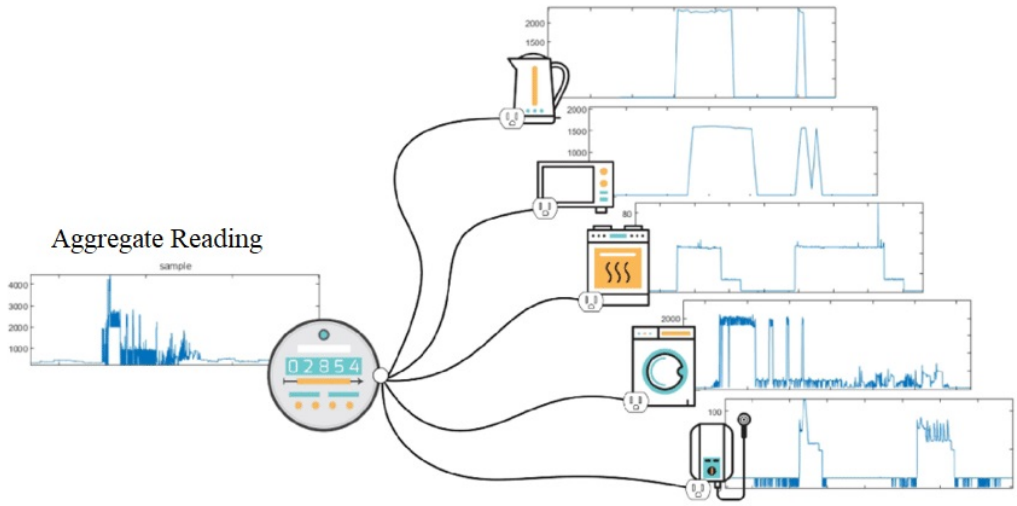

Figure 2: Non-Intrusive Load Monitoring concept (taken from Pujić et al. (2020))

NILM can be formulated as either a classification problem or a regression problem. The regression problem is when the algorithm needs to predict the power consumption of each device at each time interval. NILM can also be used for classification by determining whether the device is ON or OFF instead of predicting its consumption at each time interval.

1.1 Problem Statement

Let the time series of aggregate measurements $Y = (Y_1, Y_2, \ldots, Y_T)$ where $Y_t \in \mathbb{R}^+$ represent the energy or power measured in Watt-hours or Watts consumed by the building at time t. The building facility is assumed to have m appliances and for each appliance, the energy signal is represented as $X = (X_{i1}, X_{i2}, \ldots, X_{iT})$ where $X_{it} \in \mathbb{R}^+$. Then, the aggregate energy consumed at time t can be expressed as the

sum of energy consumed by individual appliances plus some error.

$$Y_t = \sum_{i=1}^{m} X_{it} + \epsilon_t \,,$$

where ϵ_t represents the error at time t.

NILM aims to retrieve the unknown signals X_i when the aggregate signal Y is given.

In the classification approach, NILM is used to predict whether a given appliance at time t is in ON state ($S_{it} = 1$) or OFF state ($S_{it} = 0$). It is not recommended to determine if the appliance is ON or OFF by using just its energy consumption (Precioso et al. (2020)). Thus, the usual criterion is to establish a threshold T_i for each appliance and define

$$S_{it} = H(X_{it} - T_i) \,,$$

where X_{it} is the energy consumed by individual appliance i, H(x) is the Heaviside step function

$$H(x) = \begin{cases} 1, & x >= 0 \\ 0, & x < 0 \end{cases}$$

.

1.2 Advantages of NILM

- Detailed information regarding how much energy is being used by individual appliances can be obtained with the help of NILM. This information allows the consumer to figure out which appliances are consuming a high amount of energy and helps end users in reducing their electricity consumption.

- Since NILM can detect which machines consume the most energy, the end users can avoid using these appliances when electricity is either costly or has a high carbon footprint. Demand Response (DR) is a term used for programs designed to encourage end-users to make short-term reductions in energy demand in response to a price signal from the electricity hourly market, or a trigger initiated by the electricity grid operator. This helps end users by reducing their consumption when electricity is costly.

- Peak demand is the highest amount of energy used during a 15-minute period during the month. This peak demand determines the rate at which the end users (who consume more than 100 MWh a year) are charged for the electricity they consume. With the help of NILM, industries can identify when they are using the most power each day, along with which machines they are using at that time. This information can help industries to find ways to reduce their peak demand. The process of reducing peaks either by temporarily scaling down consumption, activating an on-site power generation system or relying on a battery is known as peak shaving.

1.3 Challenges with NILM

- Different houses have different numbers and types of appliances. These appliances also have multiple versions that consume different amounts of energy. For example, many versions of the appliance microwave exist from various brands. These microwaves can consume varying amounts of energy.

- To use NILM, data must be collected to train the NILM algorithms. This data collection process is expensive. Smart meters are needed which monitor the energy consumption of various appliances and the overall energy consumption.

- NILM solutions need to give real-time results and depend on simple hardware infrastructure to also make it inexpensive.

- The NILM algorithms that are trained on a limited number of houses need to be able to generalize to other houses (or datasets). NILM solutions need to be able to give good results on different houses with different energy consumption patterns.

1.4 Motivation

In the transition to a renewable energy system, many devices with high-connection powers, for example, electric vehicle chargers, PV systems and heat-pumps are installed on the low voltage electrical grids. With the increasing number of such devices in the grid, it becomes a more complex task to keep the grid stable. In Germany, these devices have to be registered in big databases and hence are theoretically known to the grid operator but in reality, the knowledge that the grid operators have is often incomplete. Besides, electric vehicles are moved when used and may change the grid connection point. Therefore, the detection of high energy consuming appliances' load profiles enables grid operators to identify critical events early and react to them. A load profile is the time-series data that contains the electrical consumption of an appliance/house. Energy management and monitoring is thus a crucial step to help stabilize these grids.

Energy monitoring is also needed to acquire appliance-specific energy consumption statistics. Knowledge of statistics like electric vehicle charging patterns is also required for smart grid solutions like Vehicle-to-Grid (V2G) and DR. V2G is a technology that enables energy to be sent back to the power grid from the battery of an electric car. Information about the overall energy consumed as a result of charging EVs could be useful for households to manage the running costs of EVs just like fuel costs for

petrol and diesel cars. In this thesis, we detect electric vehicle charging events and energy consumption from heat-pumps and show the suitability of the used NILM algorithms for this task.

When the energy consumed by a client household is above 10000 kWh per year, the smart meter data will be transferred at a 15-minute rate to the grid operators. Since the grid operators receive the smart meter data at a 15-minute rate, it makes sense to test the performance of the NILM algorithms at this sample rate. Since the energy used by heat-pumps depends on the outdoor temperature, it would be useful to see if this information helps the model in the disaggregation task.

1.5 Contributions

- Creation of the synthetic dataset in NILMTK format from the outputs of the Synpro tool.
- Analysis of the different datasets by visualizing the data.
- Evaluating the impact of weather information used as additional input to the models.
- Converting the BERT and Seq2point models into multi-input or multi-output models.
- Adapting the NILMTK API to run variations of different deep learning models used like the multi-input or multi-output models.
- Converting the BERT model to the BERT2Point model.

- Comparing the performance of different deep learning models on the two datasets used in this thesis.

- Analyzing if the additional training data from the Synpro dataset improves the performance of Seq2point and BERT algorithms when training on both the Synpro and the Dataport dataset and testing it on the Dataport dataset.

- Analyzing the performance of the Seq2point and BERT algorithms in predicting EV charging events.

1.6 Overview

This thesis is structured as follows:

- Section 2 contains the literature review of the relevant research publications in NILM.

- In Section 3, we first explain some additional information about NILM. It then explains the background of the deep methods used in this work.

- In Section 4, we introduce the different datasets used.

- In Section 5, we describe the different algorithms used in this thesis and is followed by a description of evaluation metrics.

- In Section 6, we discuss the results of the experiments conducted.

- In Section 7, we summarize the findings of this thesis and give a conclusion.

- In Section 8, we propose future works on this topic.

- Appendix A contains results of some hyper-parameter optimization experiments.

2 Related Work

The field of NILM aroused great research interest after it was proposed in 1992 by Hart (Hart (1992)). Today there is a large amount of published information on the topic. In this section, we will explore some of the relevant and successful approaches used for NILM. Section 2.1 presents some related work using legacy algorithms. Section 2.2 presents some related work using deep learning based algorithms.

2.1 Legacy Algorithms

The first algorithm for NILM was proposed by George Hart in 1992 (Hart (1992)). It is based on dividing the aggregate energy time series data into steady time states as seen in Figure 3 below. The signal is then considered to be a sequence of stable states. A new sequence starts when the level of power changes as seen in Figure 3. Thus, edges in the steady time series data can be detected, which signifies that an appliance has changed its state. The difference between the old and new steady state power is used to determine which appliance has changed its state.

Figueiredo et al. (2011) and Dong et al. (2013) added to the solution suggested by Hart by improving the way unique signatures of individual appliances are collected. This results in better targets for signature matching. Another common approach used for NILM which does not involve neural networks is hidden Markov models (HMM) and their many adaptations. Parson et al. (2011) first proposed the idea

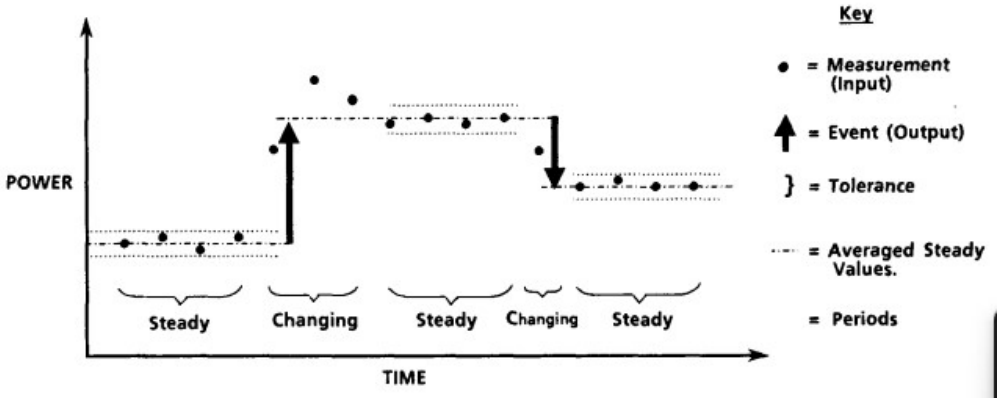

Figure 3: Edge detection from "Prototype Non Intrusive Appliance Load Monitoring", G. W. Hart, 1985

of using HMM. The main drawback of using this approach is that the number of possible states that result in aggregate energy consumption increases exponentially with the number of appliances. So the disaggregation task was limited to the top 3 energy-consuming devices. Kolter and Jaakkola (2012) solved this problem by replacing the HMM with an additive factorial hidden Markov model (AFHMM) for each appliance. Here each appliance has its own HMM which is independent of other appliances. This reduces the computational complexity and ignores the correlation between appliances. Bonfigli et al. (2017) improved on the previous approach by using both active and reactive power as inputs to AFHMM.

2.2 Deep learning Based Algorithms

Deep learning algorithms, with their ability to solve complex regression or time series forecasting tasks are perfect candidates for solving NILM.

The first use of deep learning for NILM only happened with Kelly and Knottenbelt (2015) where three different deep learning algorithms were compared. The first algorithm uses a denoising autoencoder (DAE) for the disaggregation task. An

autoencoder is an unsupervised artificial neural network that first compresses the data using an encoder and then learns to reconstruct the input from the reduced form. A DAE is an autoencoder that attempts to reconstruct a clean target from a noisy input. DAE was used for NILM by creating one model per appliance that considers the main meter reading (aggregate reading) as a noisy input. The network then reconstructs the clean power demand of the target appliance. DAE gets the main meter reading of a specific length and then outputs the target appliance consumption for the same sequence length. The second algorithm used a Recurrent Neural Network (RNN) for the disaggregation task. RNN is a type of neural network that has internal memory and works very well with sequential data. This network receives a sequence of main meter readings and outputs a single value of power consumption of the target appliance. Instead of having a normal RNN which has the vanishing gradient problem, LSTMs are used. A bi-directional LSTM is used as it improves the performance of the algorithm. The third algorithm was used to estimate the start, end and total energy consumed by each activation of the appliance. This was done as many applications of energy disaggregation do not require detailed second-by-second reconstruction of the appliance power demand.

Le et al. (2016) used a Gated Recurrent Unit (GRU) instead of an LSTM. The architecture used in this paper was a less complex model than LSTM that required lesser training time. Rafiqet et al. (2018) regularized these models using dropout for LSTM and GRU models and observed a performance improvement.

Research (Wibawa et al. (2022)) has shown that using one-dimensional Convolutional Neural Networks (CNNs) for time series classification has several important advantages over other methods. These 1D CNNs are able to extract very informative, deep features, which are independent of time. Zhang et al. (2018) suggest replacing the autoencoder from Kelly et al. (2015) which they call a sequence-to-sequence (Seq2seq) disaggregator, with a CNN. The Seq2seq model generates multiple outputs for a particular time point due to overlapping windows. Instead, this algorithm is

trained to predict only the midpoint of the window. This allows the neural network to focus on predicting the midpoint of the window, rather than on the more difficult outputs on the edges, yielding more accurate predictions. This model, which the author called sequence-to-point (Seq2point) disaggregator, will be discussed in further detail in Section 5 as it is used in this thesis.

Transformer models proposed by Vaswani et al. (2017) and their variations have proven to be especially effective in the field of natural language processing. Transformers process the entire sequence in parallel and are thus fast in training. The attention mechanism used in transformers retrieves information from the entire sequence and thus helps with long-range dependencies. Bidirectional Encoder Representations from Transformers (BERT) is an encoder only transformer introduced for language understanding. Yue et al. (2020) use transformers for NILM and called their model BERT4NILM. This was a sequence-to-sequence model that had positional encoding and a transformer encoder block added. This model will be discussed in further detail in Section 5 as it is used in this thesis.

I had previously explored the topic of NILM for electric vehicle charging event detection [1]. In this project, the performance of five NILM algorithms (Seq2Point, Seq2seq, GRU, DAE, RNN) from NILMTK at a 15-minute sample rate was compared using UK-DALE dataset for three appliances (fridge, dish-washer, washer-dryer). The performance of the Seq2Point algorithm is better than other algorithms in almost all metrics and appliances. The sensitivity to the sample rate was analyzed for the Seq2point algorithm by a comparison of its performance at a 15-minute sample rate and a 5-minute sample rate. The Seq2point algorithm gave better results at a 5-minute sample rate as the amount of training data available to the algorithm when trained at a sample rate of 5-minutes is 3 times more than when sampled at a rate of 15-minutes. The sensitivity towards a higher number of datapoints in training in a 5-minute sample rate was analyzed for the Seq2point algorithm by a comparison of a 15-minute sample rate and a 5-minute sample rate with a similar number of

datapoints. The performance in terms of RMSE of the algorithm at a sample rate of 15-minutes compared to a sample rate of 5-minutes is better for the appliances fridge and washer-dryer but worse for the appliance dish-washer. This shows that the increased performance in the previous experiment is mainly due to an increase in training data. The performance of the best Seq2point in predicting power consumed while charging electric vehicles was analyzed and compared with other algorithms in a synthetic dataset called the Synpro dataset. The RMSE error of the Seq2point model is lower than the corresponding RMSE error in the RNN model in 11 houses out of the 15 houses in the dataset. The Seq2point algorithm outperforms all other algorithms in all 15 houses in all metrics used in the project. The performance of the Seq2point algorithm in electric vehicle charging event detection was then analyzed. The model was able to predict with high accuracy if the EVs were charging or not (F1-Score of above 0.9 in 14 out of the 15 houses).

In this thesis, the heat-pump appliance was added to the Synpro dataset which is a high energy-consuming device. This is expected to make the EV charging events prediction harder. BERT algorithm's performance is also compared using the datasets mentioned in Section 4. The other contributions of this thesis are mentioned in Section 1.5.

3 Background

This section gives information about NILM, neural networks, deep learning and background information, notation and definitions for the reader to understand the following chapters.

3.1 NILMTK

NILMTK (Batra et al. (2014)) is an open-source toolkit for comparative analysis of NILM algorithms across various datasets. It also provides a pipeline from datasets to metrics to lower the entry barrier for researchers. NILMTK was created for three main reasons. Firstly, it allows the comparison of state-of-the-art approaches. Secondly, it allows comparisons of the algorithm's performance on various datasets, so that it can be verified if the approach can be generalized to new data. Thirdly, it gives users access to a stable set of metrics that help researchers access the performance of the algorithms for various use cases. [1] The NILMTK workflow is shown in Figure 3. The differences in the purpose of using each dataset have led to completely different data formats being used. This results in a time-consuming barrier when using different datasets. The data from the datasets used are first converted to NILMTK-DF which is the standard energy disaggregation data structure used by the toolkit. NILMTK-DF is a common data set format inspired by the REDD format (Kolter et al. (2011)), into which existing data sets can be converted. NILMTK contains parsers for the different

[1] https://github.com/nilmtk/nilmtk

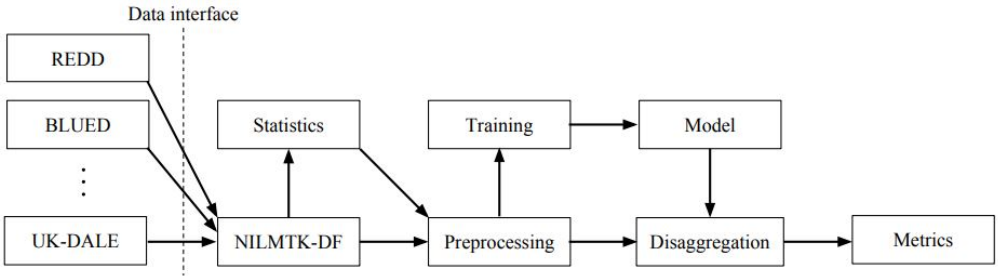

Figure 4: NILMTK-Workflow

datasets from the literature which convert the dataset into NILMTK-DF. Various statistics can be observed to analyze the dataset. NILMTK provides statistical and diagnostic functions which provide a detailed understanding of each data set. Preprocessing functions are available that help in mitigating the challenges with NILM datasets. NILMTK provides the easy use of different algorithms to disaggregate the data. Evaluation of results is possible with the metrics that are provided by the toolkit. NILMTK also provides an API to run different experiments. The API makes running experiments extremely quick and efficient, with an emphasis on creating finely tuned reproducible experiments where model and parameter performances can be easily evaluated at a glance.

3.2 Deep Learning

Deep learning is a subset of machine learning that uses algorithms inspired by the human brain to solve complex tasks. Machine learning is a subset of artificial intelligence that involves a computer algorithm that, given a particular context (data) makes a complex decision to achieve a goal. Artificial intelligence is the process of developing self-reliant machines that are inspired by human behavior. The techniques and methods in this thesis are from the field of supervised learning. Supervised learning is a branch of machine learning that involves using a model to learn a mapping between input examples and the target variable. Neural networks make up

the backbone of deep learning algorithms. In recent years, deep learning has quickly become a tremendously popular field of research

3.2.1 Artificial Neural Networks

Artificial Neural Networks (ANN) are algorithms that are loosely inspired by biological neural networks in the human brain. These networks contain multiple interconnected artificial neurons, also known as nodes, which are organized in layers. In an ANN, the output from certain artificial neurons is used as input to others allowing complex calculations. There are many types of neural networks that are used on a wide variety of tasks.

A Multi-Layer Perceptron (MLP) is a simple neural network that transforms the input data through a series of non-linear transformations that are performed at the different nodes to generate output. The MLP contains at least three layers, the input layer, the hidden layer, and the output layer. The input layer receives all the inputs needed by the model. Similarly, the output layer generates all the output predictions. The layers in between the input and the output layer are called the hidden layers and contain the hidden nodes. MLP can contain more than one hidden layer, with each layer's output serving as the next layer's input. An example of MLP is shown in Figure 5. It has n input neurons and m output neurons and two hidden layers.

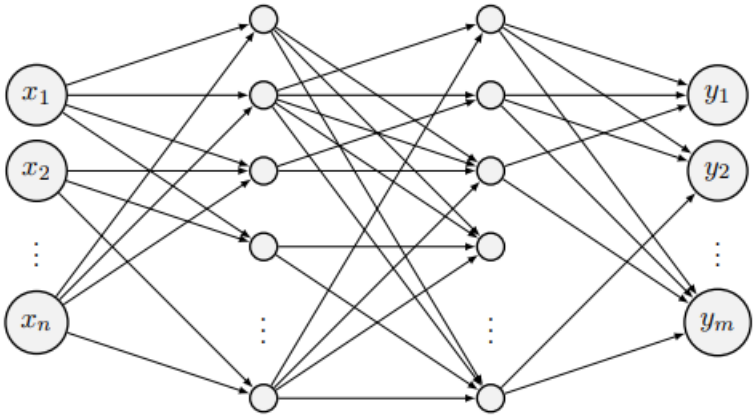

Figure 5: Multi Layer Perceptron

Let the input to the MLP be a vector x and the MLP contains $l-1$ number of hidden layers. The last layer l is then the output layer. The hidden layers i computes the output as shown below:

$$y_i = f_i(y_{i-1})\,,$$

where $y_1 = f_1(x)\,$, x is the input vector to the model, f_i is the transformation that occurs at the layer i and y_i is the output at layer i. The output of the first hidden layer is then used as input to the next layer. Similarly, each layer uses its previous layer's output as input. Thus, the output of the MLP is:

$$\hat{y} = f_l(y_{l-1})\,.$$

The output layer gives the prediction of the model $\hat{y}$.

The input to the hidden and the output layer is first transformed linearly. For an input vector x, the linear transformation is:

$$z = Wx + b\,,$$

where W is the weight matrix and b is the bias.

This is followed by a non-linear activation function $g(z)$. Thus, the transformation that occurs at a layer i can be described by

$$f_i(y_{i-1}) = g(W_i y_{i-1} + b_i)\,,$$

where g is the activation function, W_i is the weights connecting layer i and $i-1$, y_{i-1} is the output of the layer $i-1$ and b_i is the biases at layer i.

Non-linear activation functions are used since the neural network would also like to solve complex non-linear solutions. The rectified linear unit (ReLU) is a simple widely used activation function:

$$ReLU(z) = max(0, z)\,.$$

3.2.2 Training a Neural Network

The weights W and the biases b are called trainable parameters of the model. A neural network needs to be trained using the features (inputs) and ground truth labels (expected outputs) to find the best trainable parameters.

Loss Function

Loss functions are used to evaluate how good or bad the model is at predicting the outputs. Loss functions do this by determining the error between the output of our algorithms and the given target value. Loss functions selected need to be differentiable and continuous so that they can be used in training. A common loss function for the regression task is the Mean Squared Error (MSE). It can be formulated as follows

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y_i} - y_i)^2\,,$$

where $\hat{y}_i$ is the i^{th} prediction and y_i is the i^{th} ground truth and N is the total number of outputs to be predicted.

Training and Backpropogation

Firstly, the model's weight and biases are initialized. A common strategy used is random initialization. The model then uses the input data to generate predictions of the output. This is known as a forward pass and the error is calculated using the loss function. The model then needs to update its trainable parameters to minimize the loss function. The weights and biases are updated using the Gradient Descent algorithm with the following equation:

$$w_{new} = w_{old} - \alpha \frac{\partial L(y, \hat{y})}{\partial w},$$

where w_{new} is the updated weight and w_{old} is the old weight. α is the learning rate and $\frac{\partial L(y,\hat{y})}{\partial w}$ is the rate of change of the loss function L to the change in weight. Similarly, the biases need to be updated.

The **learning rate** is an important hyperparameter that controls how much to change the model in response to the estimated error each time the model weights are updated.

Adam (Kingma and Ba (2015)) is an optimization algorithm that can be used to update the trainable parameters. In addition to storing an exponentially decaying average of past squared gradients, Adam also keeps an exponentially decaying average of past gradients. The parameters are updated with the following rule:

$$\theta_i = \theta_{i-1} - \frac{\alpha m_t}{\sqrt{v_t} + \epsilon},$$

where m_t and v_t are estimates of the first moment (the mean) and the second moment (the uncentered variance) of the gradients respectively, θ_i is the i^{th}parameter, α is

the learning rate and $\epsilon = 10^{-8}$.

Regularization and Hyperparameters

One of the most important aspects when training neural networks is to avoid overfitting. Overfitting is the phenomenon where a neural network can give good results with the training data but fails to generalize when it sees new data from the same problem domain. A model with very high complexity can pick up and learn noise in the data that is just caused by some random fluctuation or error. Thus, regularization refers to a set of different techniques that lower the complexity of a neural network model during training, thereby reducing the chance of overfitting.

Dropout is a commonly used regularization technique. During training, a neuron of the neural network gets turned off with some probability P. This results in temporarily removing the neuron that is turned off from the network, along with all its incoming and outgoing connections. Dropout reduces the number of nodes in training which results in reducing the complexity of the model. A less complex model reduces the chance of overfitting.

Hyperparameters are the variables that determine the network structure and the variables which determine how the network is trained. Several hyperparameters exist that control model complexity, characteristics and optimization choices. The number of nodes in each layer, the number of hidden layers and the learning rate are common hyperparameters. Given these hyperparameters, the training algorithm learns the trainable parameters from the data. Thus, finding the optimal hyperparameters is an important task.

3.2.3 Convolutional Neural Networks

A Convolutional neural network (CNN) is a neural network that has one or more convolutional layers. CNNs are mainly used for image processing, image classification, image segmentation and signal processing tasks. CNNs use convolution operations that can handle spatial information available in images. CNNs have recently been used for both natural language processing and time series data. CNNs use weight sharing and are computationally efficient, requiring less training time compared to many other neural networks. CNNs are very effective in reducing the number of parameters and they can extract very informative and deep features.

CNNs can use convolutions in either one, two, or three dimensions for time-series, images, and videos respectively. Figure 6 shows one-dimensional convolution used for time series data.

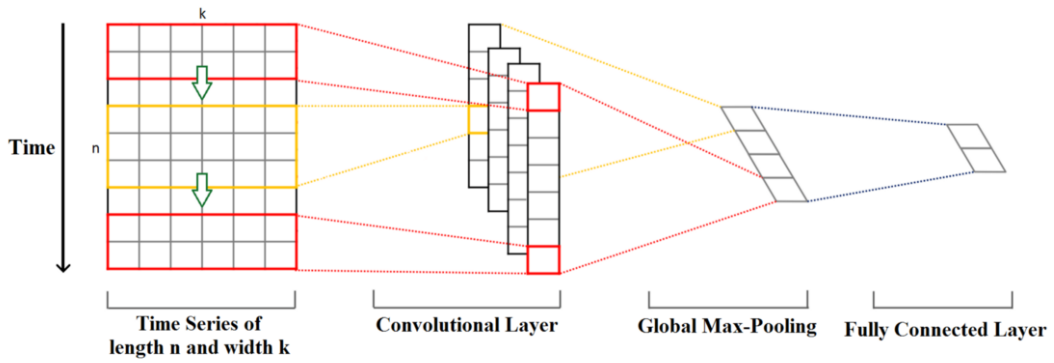

Figure 6: 1D Convolution for time series data (taken from `https://towardsdatascience.com/how-to-use-convolutional-neural-networks-for-time-series-classification-56b1b0a07a57`)

Let the length of the time series input x be n and have k features. As seen in Figure 6 convolution kernels always have the same width as the time series input. Unlike with image data where 2D Convolutions move in both length and width, 1D

convolutions only move in one direction from the beginning of a time series towards its end, performing convolution. A 1D convolutional operation involves a filter $w \in \mathbb{R}^{n \times k}$, which is applied to a window of l time series inputs to produce a feature. For example feature c_i is generated from a window of inputs $x_{i:i+l-1}$ by

$$c_i = g(w \cdot x_{i:i+l-1} + b)\,,$$

where g is the activation function and b is the bias term.

This filter is then applied through each possible window of inputs $x_{1:l}, x_{2:l+1}, ..., x_{n-l+1:n}$ to produce a feature map

$$c = [c_1, c_2, ..., c_{n-l+1}],$$

with $c \in \mathbb{R}^{n-l+1}$. This is the process by which one feature is extracted from one filter. The model uses multiple filters with varying window sizes to obtain multiple features.

Pooling

Convolutional layers in a CNN systematically apply learned filters to inputs to create feature maps that summarize the presence of those features in the input. One of the problems associated with the feature map output of convolutional layers is that they record the precise position of features in the input. Therefore, small movements in the position of the feature in the inputs will result in a different feature map. Pooling layers are used to solve this problem. They operate upon each feature map separately to create a new set of the same number of pooled feature maps. Pooling layers provide an approach to down-sampling feature maps by summarizing the presence of features in patches of the feature map. Thus pooling layers are commonly used after convolutional layers. Two common functions used in the pooling operation are:

- **Average Pooling**: Calculate the average value for each patch on the feature map.

- **Max Pooling**: Calculate the maximum value for each patch on the feature map.

3.2.4 Recurrent Neural Networks

Recurrent Neural Networks (RNN) are a type of neural network that is designed for sequential data. When it comes to sequential data, a data point would be dependent on the previous values in the data sequence. So, the MLP architecture cannot be used in this case, since the inputs of the MLP are independent of one another. RNNs take into consideration the input, as well as previous inputs to produce the output.

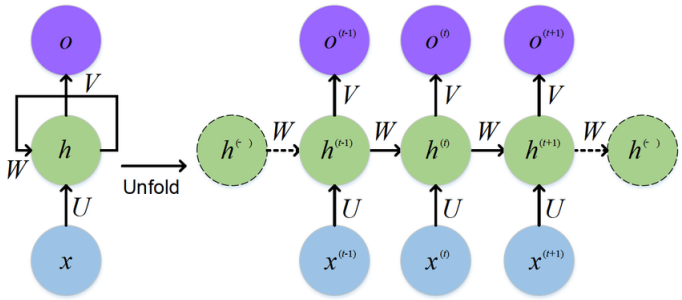

Figure 7: RNN Architecture: (taken from Feng et al. (2017))

As seen in Figure 7 the data flows through the RNN in a loop. When the architecture is unfolded, it is observed that at each time step t, the model uses the current input $\left(x^{(t)}\right)$ as well as the hidden state from a previous timestep $(h^{(t-1)})$, to produce the output $(o^{(t)})$. Hidden state h is a representation of previous inputs. Thus the formula for a vanilla RNN can be described by the equations:

$$a^{(t)} = b + Wh^{(t-1)} + Ux^{(t)}$$

$$h^{(t)} = g(a^{(t)})$$

$$o^{(t)} = c + Vh^{(t)}\,,$$

where b and c biases, U, V and W are weight matrices and g is a non-linear activation function.

Long short-term memory

RNNs suffer from the problem of vanishing gradients. The hidden state $h^{(t)}$ carries information used in the RNN, and when it becomes too small, the parameter updates become insignificant. This makes the learning of long data sequences difficult. Conversely, if the slope tends to grow exponentially instead of decaying, the gradients would explode resulting in very large updates to the RNN model weights during the training process. The Long short-term memory (LSTM) model (Hochreiter and Schmidhuber (1997)) was a special version of RNN designed to overcome some of the problems of the vanilla RNN. The architecture of LSTM is shown in Figure 8. LSTM makes use of three gates to overcome the vanishing gradient problem. The following transformations occur in an LSTM recurrent unit. The hidden state from a previous timestep (h_{t-1}) and the input at a current timestep (x_t) are combined and are then passed to the various gates.

The **forget gate** controls what information should be forgotten. It sets which values in the cell state should be discarded (multiplied by 0), remembered (multiplied by 1), or partially remembered (multiplied by some value between 0 and 1). The forget gate vector $\mathbf{f}_t$ can be formulated as,

$$\mathbf{f}_t = \sigma(\mathbf{W}_f.[\mathbf{h}_{t-1}, \mathbf{x}_t] + b_f)\,,$$

where σ is the sigmoid function, $\mathbf{W}_f$ is the weight matrix at forget gate, b_f is the bias at forget gate.

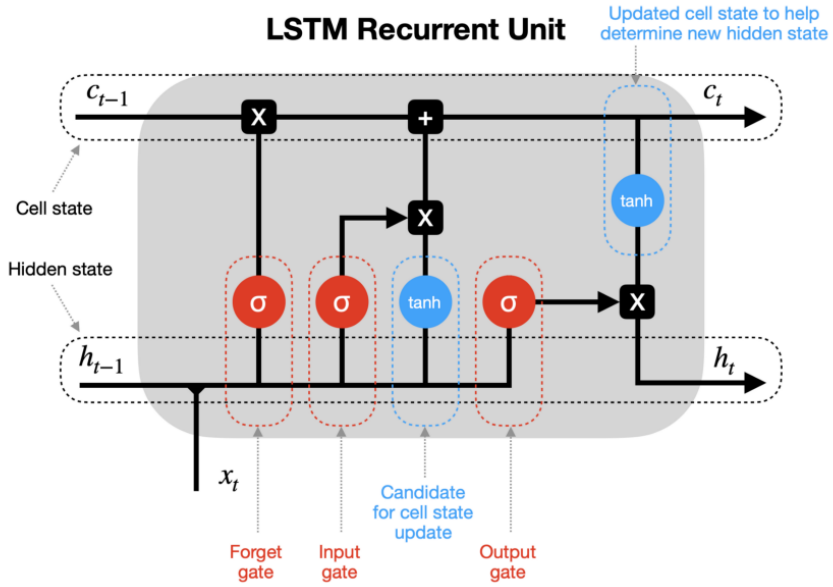

Figure 8: LSTM Architecture: (taken from `https://towardsdatascience.com/lstm-recurrent-neural-networks-how-to-teach-a-network-to-remember-the-past-55e54c2ff22e`)

The **input gate** identifies which elements are important and is used to update the cell status. The input gate vector $\mathbf{i}_t$ and cell state $\mathbf{C}_t$ can be formulated as,

$$\mathbf{i}_t = \sigma(\mathbf{W}_i.[\mathbf{h}_{t-1}, \mathbf{x}_t] + b_i)$$

$$\mathbf{C}_t = \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot tanh(\mathbf{W}_C.[\mathbf{h}_{t-1}, \mathbf{x}_t] + b_C)\,,$$

where $\mathbf{W}_i$ is the weight matrix at input gate, b_i is the bias at input gate. $\mathbf{W}_C$ is the weight matrix, b_C is the bias and $\odot$ denotes the element-wise dot product.

The **output gate** is used to update the hidden state. The latest cell state (C_t) is passed through the tanh activation function and multiplied by the results of the output gate. The input gate vector $\mathbf{o}_t$ and hidden state $\mathbf{h}_t$ can be formulated as,

$$\mathbf{o}_t = \sigma(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + b_o)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot tanh(\mathbf{C}_t)\,,$$

where $\mathbf{W}_o$ is the weight matrix at output gate, b_o is the bias at output gate, $\mathbf{o}_t$ is the output gate vector

The current cell state $\mathbf{C}_t$ and hidden state $\mathbf{h}_t$ are then used in the next time step t+1. This process repeats until the end of the sequence is reached

Gated Recurrent Unit (GRU)

Gated Recurrent Units (GRU) is a type of RNN that is similar to an LSTM. GRU is a more lightweight RNN that has fewer trainable parameters compared to LSTM. The architecture of GRU can be observed in Figure 9. GRU makes use of two gates to overcome the vanishing gradient problem. It does not maintain any cell state and solely relies on the hidden state for memory transfer between recurrent units. The following transformations occur in a GRU recurrent unit.

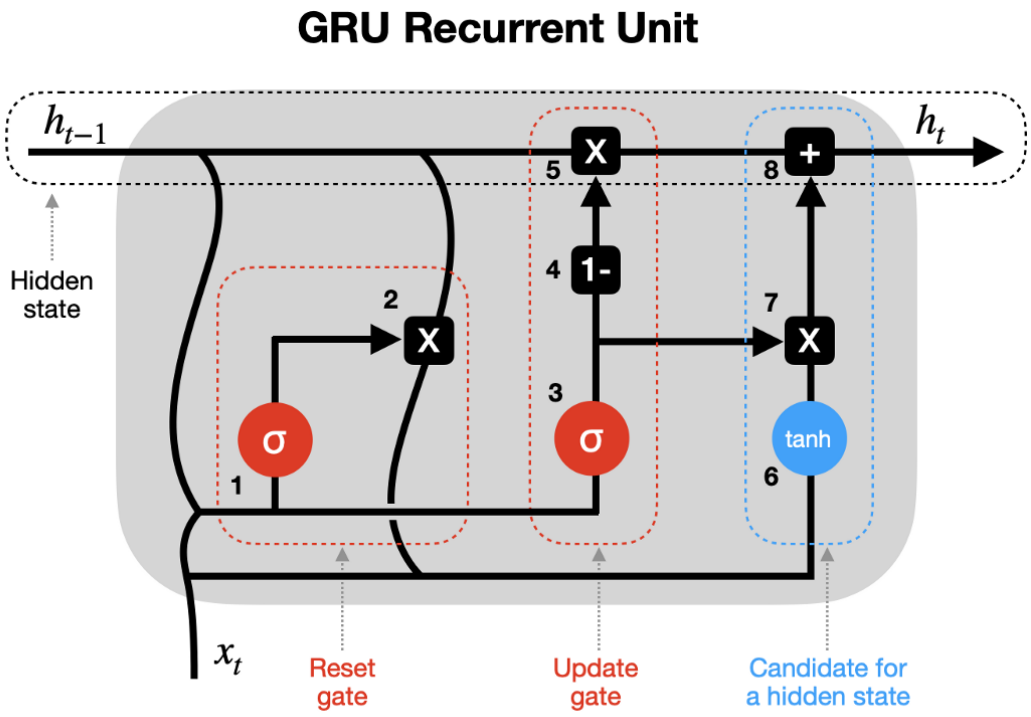

Figure 9: GRU Architecture: (taken from `https://towardsdatascience.com/gru-recurrent-neural-networks-a-smart-way-to-predict-sequences-in-python-80864e4fe9f6`)

Similar to LSTM, the hidden state from a previous timestep (h_{t-1}) and the input at a current timestep (x_t) are combined and are then passed to the two gates.

The **reset gate** is used to decide how much of the past information to forget. The reset gate vector $\mathbf{r}_t$ can be formulated as,

$$\mathbf{r}_t = \sigma(\mathbf{W}_r.[\mathbf{h}_{t-1}, \mathbf{x}_t] + b_r)\,,$$

where $\mathbf{W}_r$ is the weight matrix and b_r is the bias

The **update gate** is used to determine how much of the past information needs

to be passed along to the future. The update gate vector $\mathbf{z}_t$ can be formulated as,

$$\mathbf{z}_t = \sigma(\mathbf{W}_z.[\mathbf{h}_{t-1}, \mathbf{x}_t] + b_z)\,,$$

where $\mathbf{W}_z$ is the weight matrix and b_z is the bias

The hidden state then can be calculated using the formula

$$\mathbf{h}_t = \mathbf{z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \odot tanh(\mathbf{W}.[\mathbf{r}_t \odot \mathbf{h}_{t-1}, \mathbf{x}_t] + b)\,,$$

where $\mathbf{W}$ is the weight matrix, b is the bias and $\odot$ denotes the element-wise dot product.

The hidden state $\mathbf{h}_t$ is then used in the next time step t+1. This process repeats until the end of the sequence is reached.

3.2.5 Transformers

The Transformer is a type of neural network that is characterized by an 'attention mechanism'. This helps the model learn dependencies in long-range sequential data. The Transformer architecture was first introduced in Vaswani et al. (2017) to solve NLP tasks. Before the introduction of Transformers, RNNs, specifically LSTM networks, were state-of-the-art for NLP tasks. But Transformers are more advantageous since they can be efficiently parallelized. So larger models can be trained and a large amount of data can be processed by the Transformer.

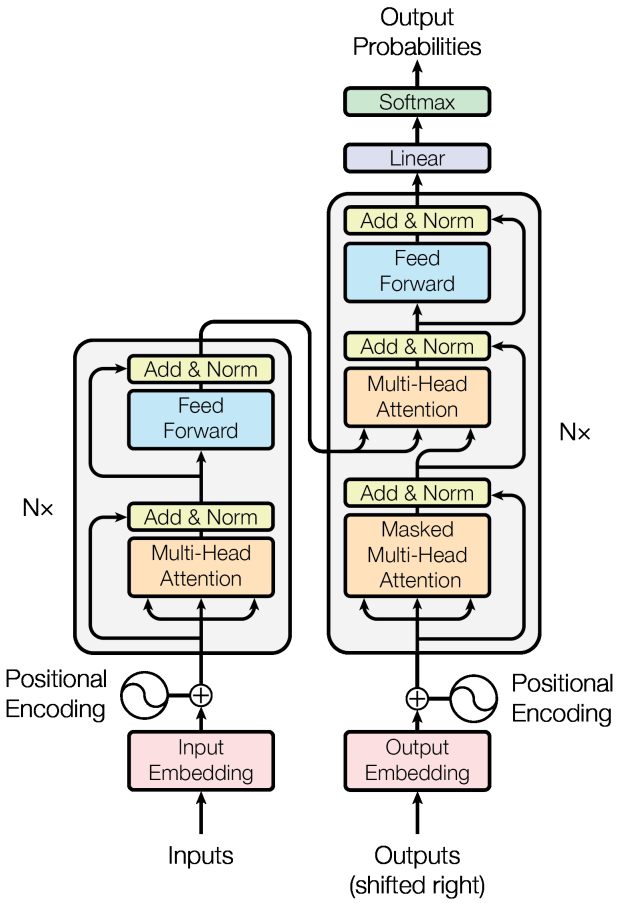

Figure 10: Transformer Model Architecture: (Image from Vaswani et al. (2017))

Figure 10 shows the architecture of the Transformer as proposed by Vaswani et al. (2017). The main parts of the Transformer architecture are:

- Positional Encoding: This gives the model an idea of where an input lies in the input sequence. So, each input is assigned a vector that denotes its position in the sequence. This is useful since Transformer does not process the input sequence one by one (like RNNs), but takes in the entire input sequence at once. So the positional encoding gives the model information about the order of the inputs in the sequence.

- Token Embedding: The role of this layer is to transform inputs to the encoder into vector representations of fixed dimensions.

- Attention Mechanism: This mechanism allows the model to go through every single input in the input sequence and learn how important each input is to produce the required output. The model learns this during training.

- Encoder: The transformer contains multiple encoder blocks stacked on top of each other. In Figure 10, the encoder block can be seen to the left of the architecture. The encoder takes the input and passes it through an attention layer and then a feed-forward layer. The attention mechanism in the encoder is applied between the same sequence, to learn underlying relationships within the input data. This type of attention is known as self-attention.

- Decoder: The decoder block is on the right of Figure 10. The input to the decoder is the expected output (during training). This input is masked using a **look-ahead mask** which prevents the model from looking at the entire sequence to make predictions. The future sequences that the model has to predict are hidden from the decoder and it is forced to use only the known values to predict the next values in the sequence. The decoder input passes through this masked attention layer (self-attention) and then is fed to the **encoder-decoder attention** layer. This attention layer uses 2 inputs, the output of the previous masked attention layer and the output of the encoder. This encoder-decoder attention is applied between two different sequences and is known as cross-attention. This then leads to a forward layer and then the final linear layer to produce the output.

In this thesis, an encoder-only transformer model is used. The main hyper-parameters of the model are :

- The number of encoder layers used.

- The number of attention heads each encoder layer contains
- The dimensions of the feed-forward network in the encoder
- The dimension of the embedding used in token and positioning embedding

Attention Mechanism

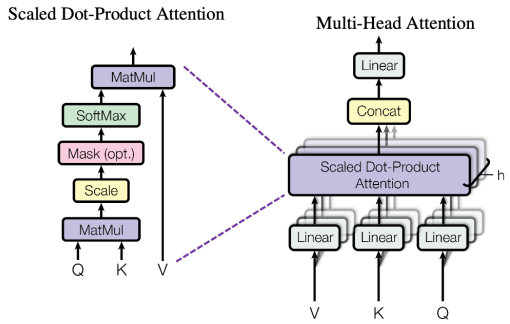

Figure 11: Attention Mechanism: Image from Vaswani et al. (2017)

The attention mechanism is presented in more detail in Figure 11. The input to the attention block is used as 3 tensors: the query, the key and the value, shown as Q, K and V. These tensors are used as follows in one attention block:

$$attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}V\right).$$

Figure 11 also shows multiple attention heads (h) in the Transformer. Hence the attention blocks in the transformer are referred to as multi-head attention. Each attention head is run in parallel. The output of the multi-head attention is as follows:

$$multihead(Q, K, V) = concat(head_1, head_2, ..., head_h)W^0,$$

Each head implements its attention as follows:

$$head_i = attention(QW_i^Q, KW_i^K, VW_i^V)\,,$$

where h is the number of attention heads and W^O, W_i^Q, W_i^K, W_i^V are weight matrices.

When the query, key and value tensors come from the same input sequence, then it is known as **self-attention**. When the query, key and value tensors are all not derived from the same input sequence, it is known as **cross-attention**.

4 Datasets

This section gives information about the two different datasets used in this thesis. Section 4.1 gives a brief description of both datasets, Section 4.2 gives information on how the datasets were converted into NILMTK format and Section 4.3 gives the basic statistics about both datasets.

4.1 Description of Datasets

NILM datasets can be divided into a low-frequency dataset being up to 1 Hz and a high-frequency dataset when above that. So, the dataset that contains electricity consumption for all the appliances (sub-meters) and main-meter (aggregate consumption) at a rate of at least one measurement per second is considered a high-frequency dataset. In this thesis, we make use of two low-frequency datasets: a publicly available dataset from the literature and a synthetic dataset.

4.1.1 Synthetic Dataset (Synpro)

The synthetic dataset is created using the Synpro tool (Fischer et al. (2015)) which was developed at Fraunhofer ISE. This tool is used to simulate the power demand for households based on the harmonized European time usage study HETUS. Each house in this dataset contains power time series for the entire year of 2017 at a sample rate of 15 minutes. In addition to the demand of the house, the charging of electric

vehicles at home with different charging powers and energy consumed by heat-pumps is simulated. This dataset also has information about the outdoor temperature.

In this dataset, 4 houses are of type "Single Family house" and 8 of type "Multi-family house". These houses have different number of occupants ranging from 1-8. Each of these houses has a charging station that charges at either of the 3 charging powers: 3.7 kW, 7.2 kW and 11 kW. A similar dataset without the appliance 'heat-pump' was used in my master's project. In addition to the 12 houses used in the thesis, the old dataset also contained 3 houses of type "Large-multi-family house". There was some issue in generating energy consumed by the appliance 'heat-pump' for the 3 houses of type "Large-multi-family house" and thus these houses were excluded.

4.1.2 Dataport

Pecan Street Dataport (Parson et al. (2015)) database is the world's largest publicly available resource for residential energy use data. They provide free access to energy consumption time-series data for 75 houses and the appliances in these houses to students for academic research. These homes are from Austin, New York and California regions. Out of the 75 houses, 11 have electric vehicle charging data. Out of these 11 houses, 6 houses contain data for one entire year. Each of these 6 houses in this dataset contains the power time-series for the entire year of 2018 at a sample rate of 15 minutes.

4.2 Conversion of Datasets into NILMTK-DF

4.2.1 Synthetic Dataset (Synpro)

The Synpro tool is used to generate four files for each house. The first file contains time-series energy consumption of all appliances other than the heat-pump, heat-pump backup, electric vehicle charging and the aggregate energy consumption of the house. The second file contains time-series energy consumption while charging an electric vehicle. The third file contains time-series energy consumption of the heat-pump and heat-pump backup appliances. The fourth file contains the ambient temperature of the house. These files from the Synpro tool are used to generate the output folder. This output folder contains one file for each appliance which is its energy consumption time series and one file for the aggregate energy consumption of the house. A parser is then implemented which converts the output files into a dataset in NILMTK format.

4.2.2 Dataport

NILMTK contains a parser for the Dataport dataset which is used to convert this dataset into NILMTK-DF.

4.3 Basic Statistics of the Datasets

4.3.1 Synthetic Dataset (Synpro)

The synthetic dataset contains 12 houses, all possessing the same appliances. An overview of the different houses is given in Table 1. All of these houses have different input settings and this affects the usage and energy consumption of different devices in these houses. This dataset is a comprehensive and clean dataset with no missing

sections. In Figure 12, the fraction of energy consumed by all devices in house number

House Number	House type	Number of occupants	Charging rate
1	Single Family house	1	3.7
2	Single Family house	2	7.2
3	Single Family house	3	11
4	Single Family house	4	3.7
5	Multi-family house	2	7.2
6	Multi-family house	2	3.7
7	Multi-family house	4	11
8	Multi-family house	4	7.2
9	Multi-family house	6	11
10	Multi-family house	6	3.7
11	Multi-family house	8	11
12	Multi-family house	8	7.2

Table 1: Overview of the Synpro dataset houses

10 of the Synpro dataset is visualized as a pie chart. Heat-pumps consume the most amount of energy in this house. The EV also consumes a lot of energy but is not used as often as the heat-pump and hence consumes less energy comparatively. The heat-pump and the EV charger are the top consumers in most other houses in this dataset.

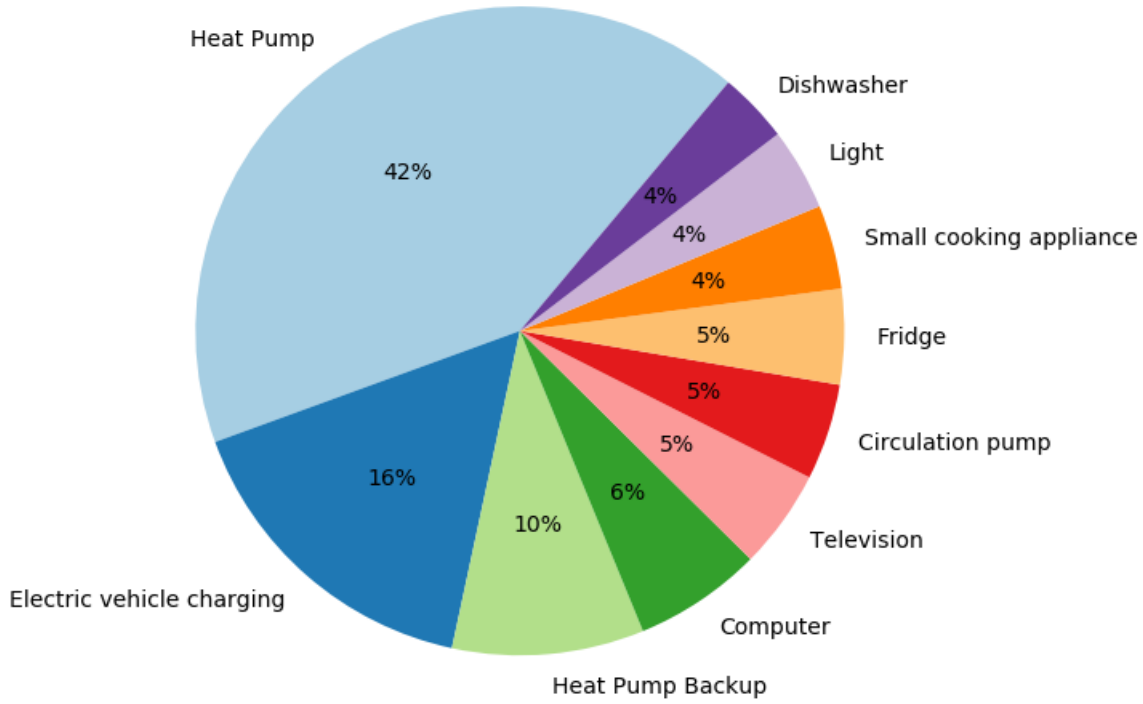

Figure 12: Fraction of energy consumption of appliances in house 10 of the Synpro dataset

Figure 13 shows the energy consumption of heat-pump in house 10. Heat-pumps consume more energy in the colder months and less energy in the hotter months. Thus, heat-pumps add seasonality to the dataset as seen in Figure 14, which shows the aggregate energy consumption in house 10 of the Synpro dataset.

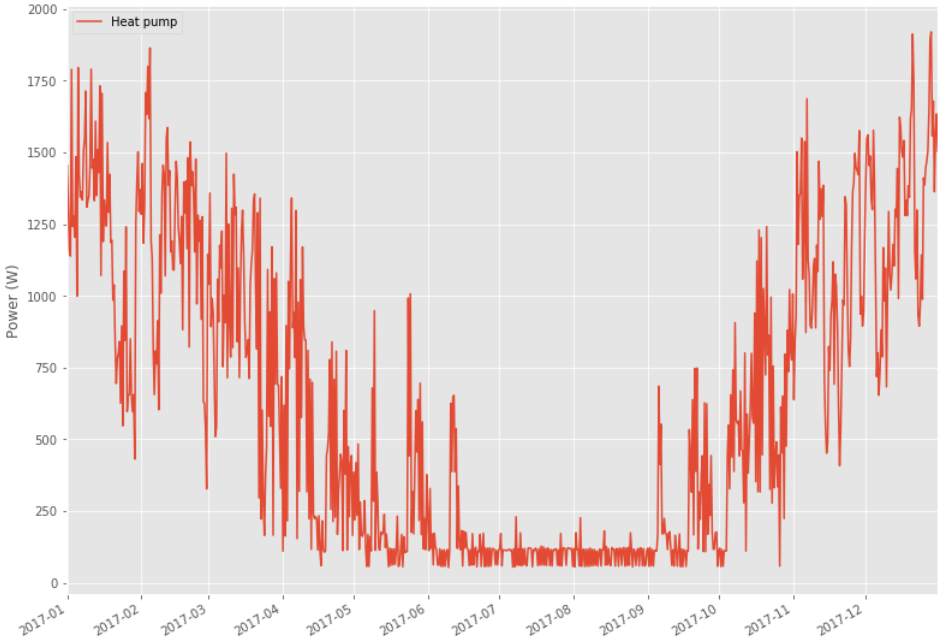

Figure 13: Energy consumption of heat-pump in house 10 of the Synpro dataset

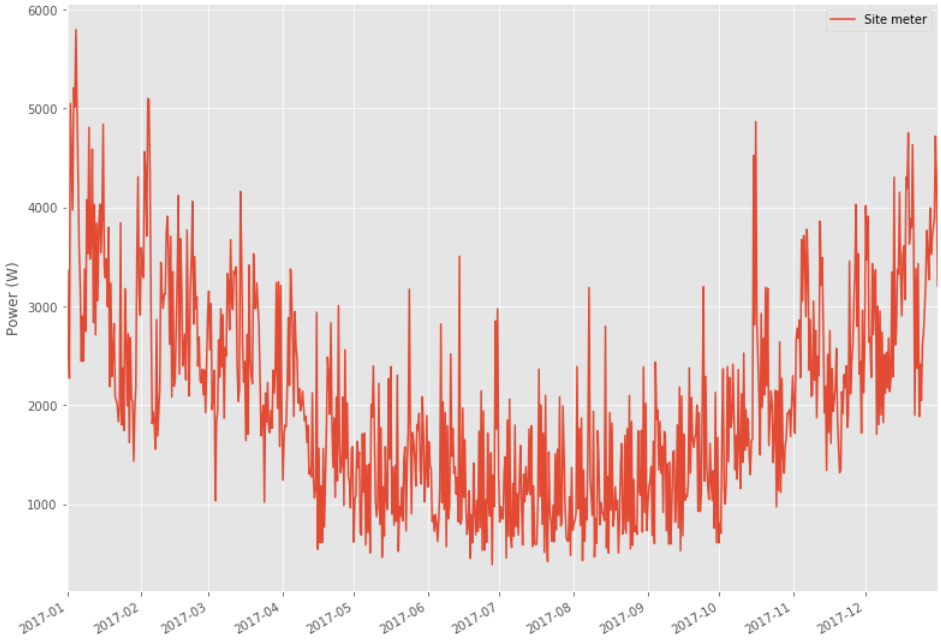

Figure 14: Energy consumption of the site meter (aggregate energy consumption) in house 10 of the Synpro dataset

Figure 15 shows the aggregate energy consumption along with the energy consumption of an EV charger and the heat-pump for a single day in house 4 of the Synpro dataset. The EV charger is used once between 15:00 and 17:30 while the heat-pump has a repeating on-and-off pattern. These two devices are mostly responsible for the aggregate energy consumption of this house for this day. The NILM algorithms need to learn the characteristic patterns of the appliances and understand how it affects the aggregate consumption of the house.

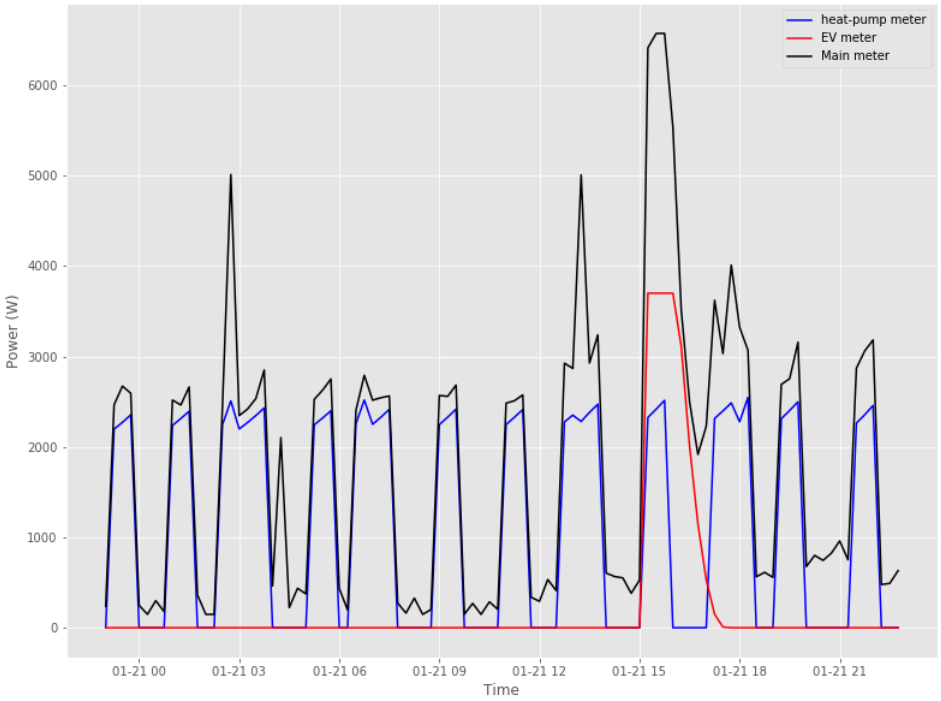

Figure 15: Energy consumption of the main meter (aggregate energy consumption), an EV charger and the heat-pump for a single day in house 4 of the Synpro dataset

4.3.2 Dataport

The Dataport dataset which we use contains 6 houses. These houses are either from California or Texas region. The Dataport dataset is a clean and comprehensive dataset. Figure 16 shows the fraction of energy consumption of the top 8 appliances

in house 1 of the Dataport dataset. The air conditioner, an EV and an electric furnace are some of the appliances that consume the most amount of energy.

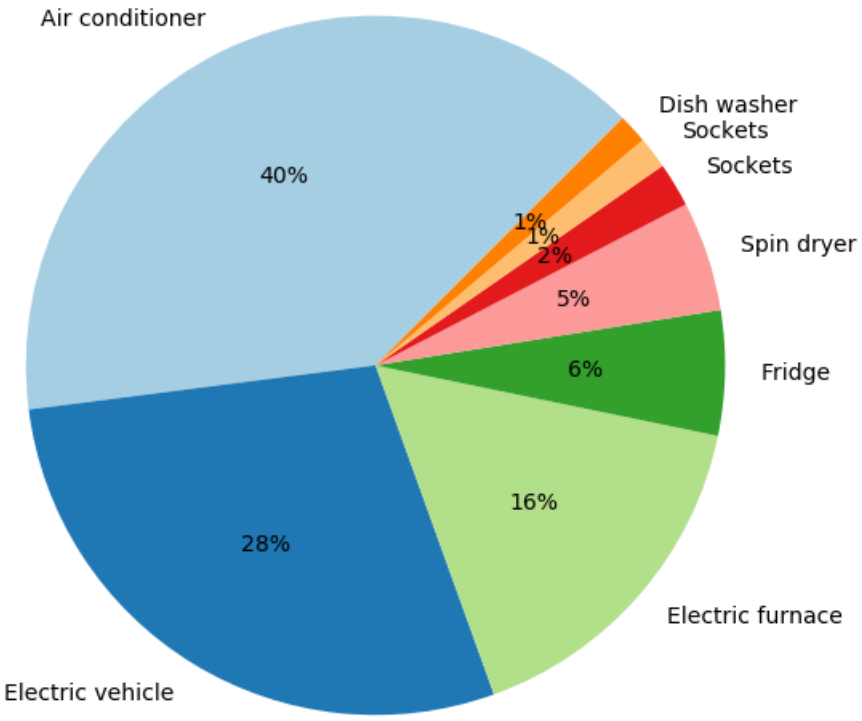

Figure 16: Fraction of energy consumption of top 8 appliances in house 1 of the Dataport dataset

Figure 17 shows the energy consumption of the air conditioner in house 1. Air conditioners are also seasonal appliances and consume more energy in the hotter months and less energy in the colder months. Thus, air conditioners add seasonality to the dataset as seen in Figure 18 which shows the aggregate energy consumption of house 1.

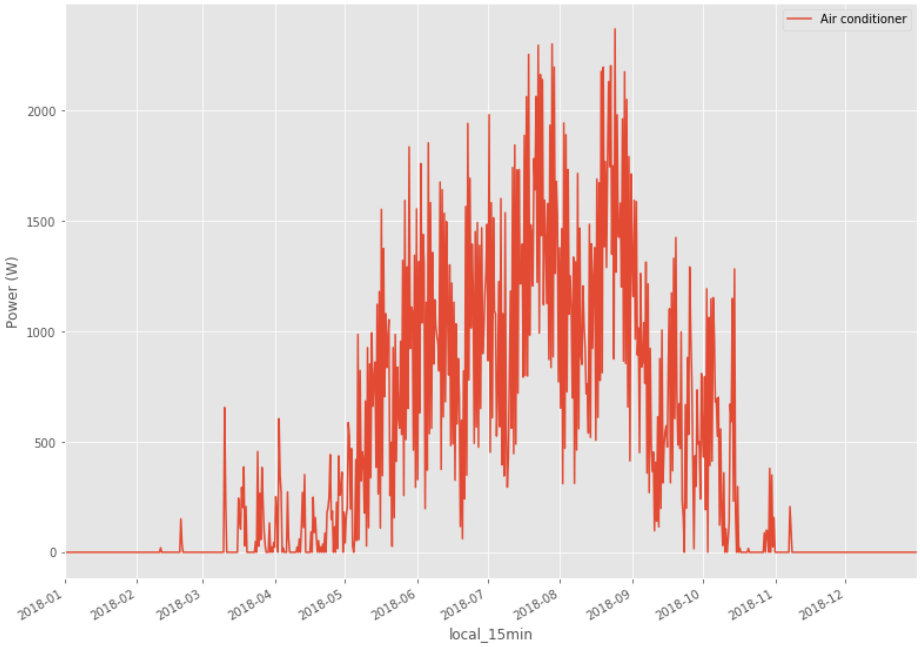

Figure 17: Energy consumption of air conditioner in house 1 of the Dataport dataset

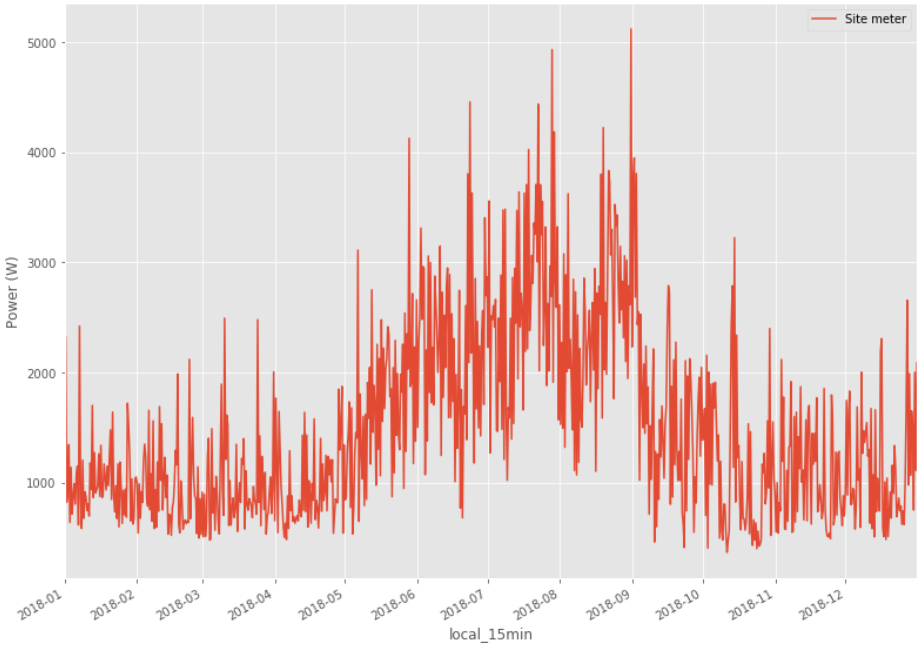

Figure 18: Energy consumption of the site meter (aggregate energy consumption) in house 1 of the Dataport dataset

Figure 19 shows the aggregate energy consumption along with the energy consumption of an EV charger, an air conditioner, an electric furnace and a spin dryer for a single day in house 4 of the Synpro dataset. The first large peak in consumption is caused by using the EV charger between 12:30 and 13:30. The second peak is caused by using the spin dryer between 17:00 and 18:00 and the magnitude of the peak is similar to the first one. The three appliance EV charger, air conditioner and electric furnace are used at the same time between 23:30 and 03:30 of the next day resulting in a large peak. This house has many large energy consuming appliances which are sometimes used concurrently.

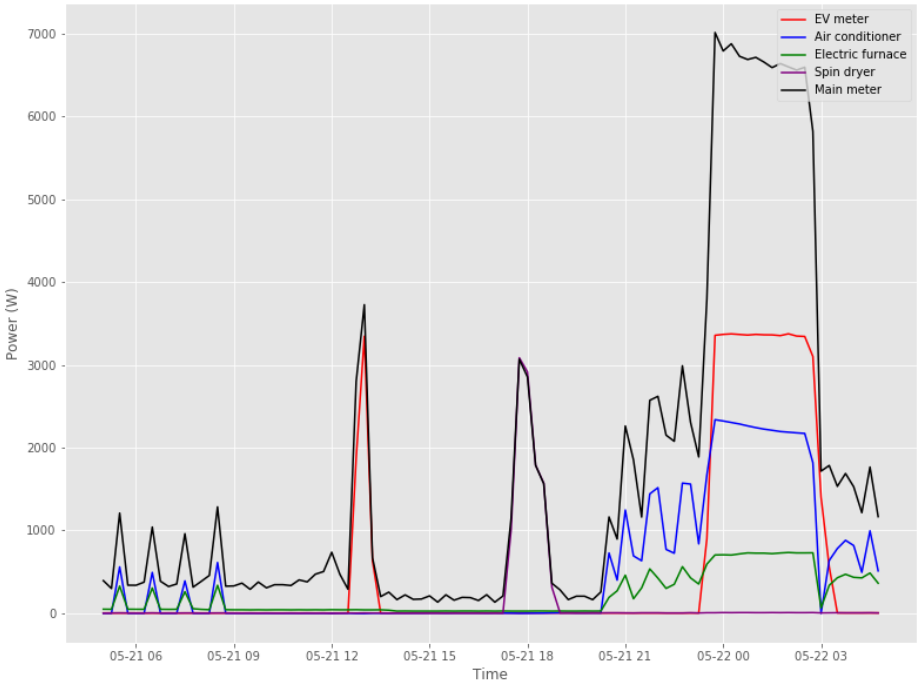

Figure 19: Energy consumption of the main meter (aggregate consumption), an air conditioner, an EV charger, a spin dryer and an electric furnace for a single day in house 3 of the Dataport dataset

5 Deep Learning Algorithms for NILM and Evaluation Metrics

This chapter explains the different deep learning algorithms used in this thesis. It is then followed by a description of the evaluation metrics used in this thesis.

5.1 Deep learning Algorithms

The deep learning algorithms that are used in this are taken from NILMTK-Contrib (Batra et al. (2019)). This repository contains many state-of-the-art algorithms in the NILM domain, which have been made compatible with NILMTK and are added here. [1]

[1] https://github.com/nilmtk/nilmtk-contrib

5.1.1 Sequence-to-Sequence

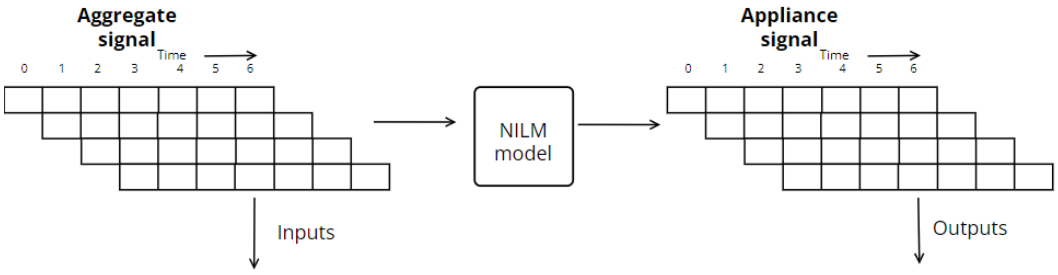

Figure 20: Sequence-to-Sequence concept

Sequence-to-Sequence learning is about training models to convert sequences of one type to sequences of another type. The Sequence-to-Sequence algorithm (Seq2seq) learns a regression map from the main meter sequence to the corresponding target appliance sequence. The neural network maps a sliding window $X_{t:t+W-1}$ of the aggregate input power to corresponding windows $Y_{t:t+W-1}$ of the output appliance power where W is window length (sequence length). Figure 20 shows the input and output of the Seq2seq model where the window length is 7. Each input sequence has a corresponding output sequence of the same length. Since there are multiple predictions for a particular time, we take the mean for each sliding window that contains that time. Architecture of Seq2seq is show in Figure 21.

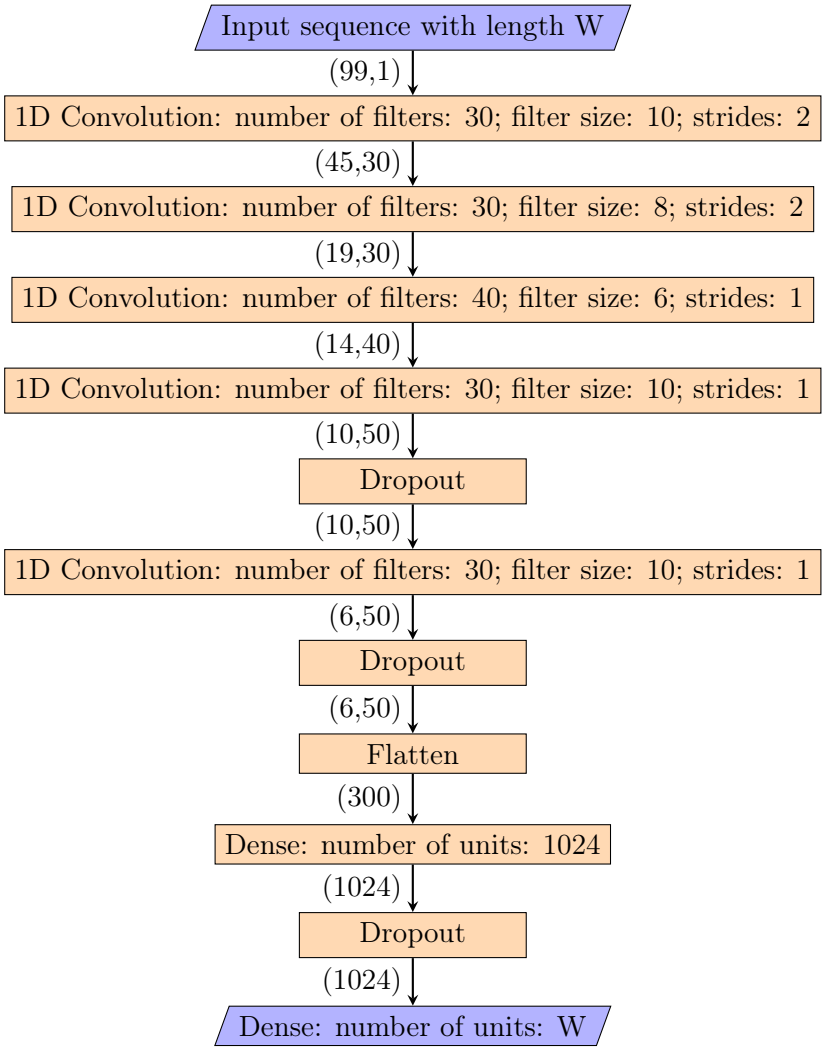

Figure 21: Seq2seq model: Output shape of each layer is mentioned next to arrows when W is 99

5.1.2 Sequence-to-Point

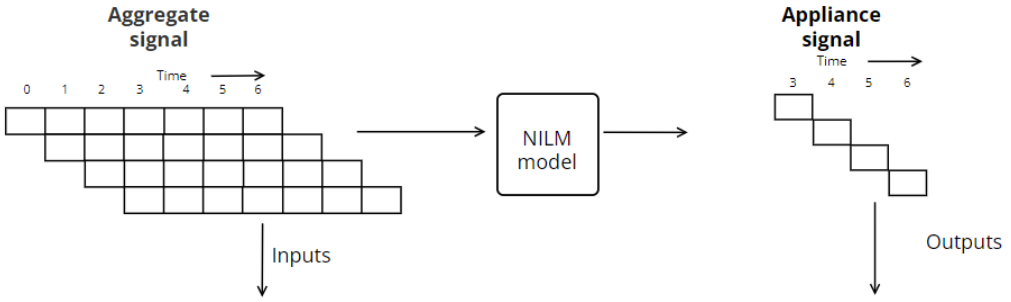

Figure 22: Sequence-to-Point concept

Sequence-to-Point (Seq2point) is a similar algorithm to Seq2seq but is trained to predict only the midpoint element of that sliding window. Thus, the output of this model only has 1 node whereas the output of Seq2Seq has multiple nodes. The neural network maps a sliding window Xt:t+W−1 of the aggregate input power to the midpoint element of the corresponding window of the target appliance. Figure 22 shows the input and output of the Seq2point model where the window length is 7. Each input has only one corresponding output which is the midpoint of the sequence. $[W/2]$ zeros need to be padded at the beginning and the end to make the first and last $[W/2]$ predictions. This allows the neural network to focus on the midpoint of the window, rather than on the more difficult outputs on the edges, yielding more accurate predictions. Many variants of Seq2point are also discussed in this thesis. The architecture of the original Seq2point is shown in Figure 23.

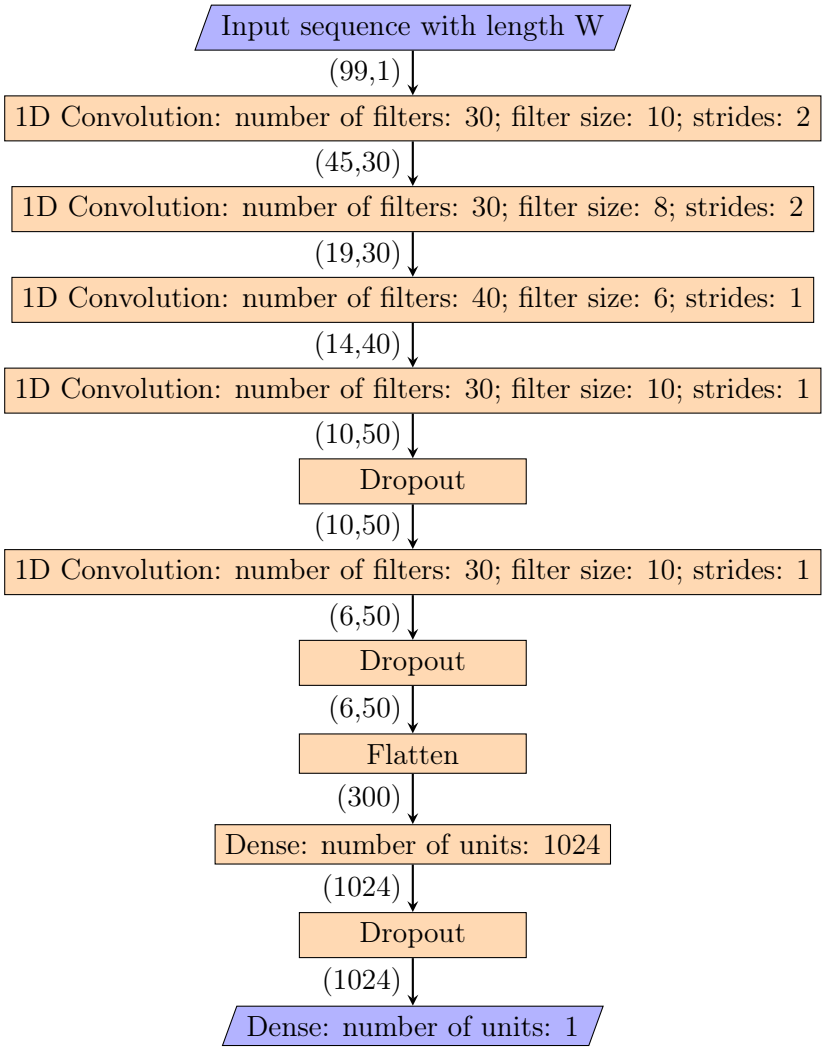

Figure 23: Seq2Point model: Output shape of each layer is mentioned next to arrows when W is 99

5.1.3 Recurrent Neural Network

A Recurrent Neural Network (RNN) is a type of neural network that has internal memory and works very well with sequential data. This network receives a sequence of main meter readings and outputs a single value of power consumption of the target appliance like the Seq2point algorithm. Instead of having a vanilla RNN which has

the vanishing gradient problem, LSTMs are used. Additionally, Bi-directional LSTMs are used to improve performance. The architecture of RNN is shown in Figure 24

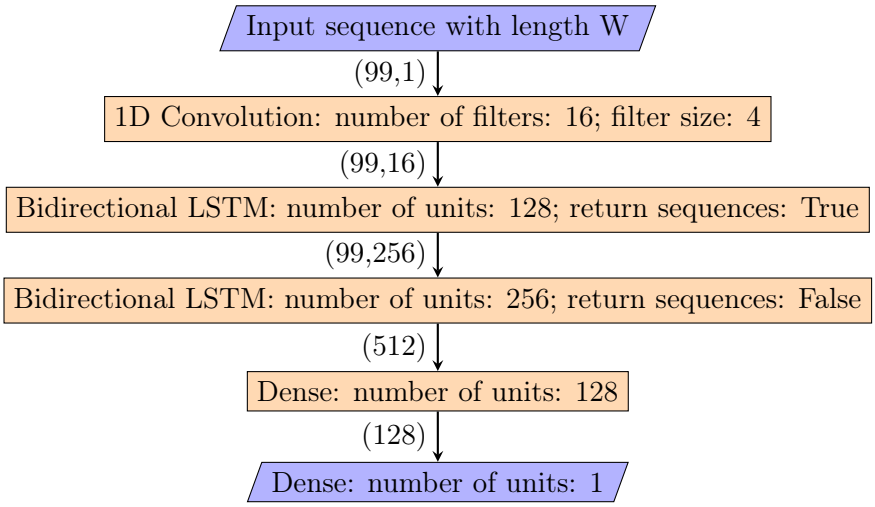

Figure 24: RNN model: Output shape of each layer is mentioned next to arrows when W is 99

5.1.4 Gated recurrent unit

This network is very similar to the RNN network but replaces the LSTMs with a more lightweight RNN called Gated Recurrent Units (GRU). Similar to the RNN network, Window GRU gets main meter readings of a specific length and then outputs a single value of power consumption of the target appliance. The architecture of GRU is shown in Figure 25.

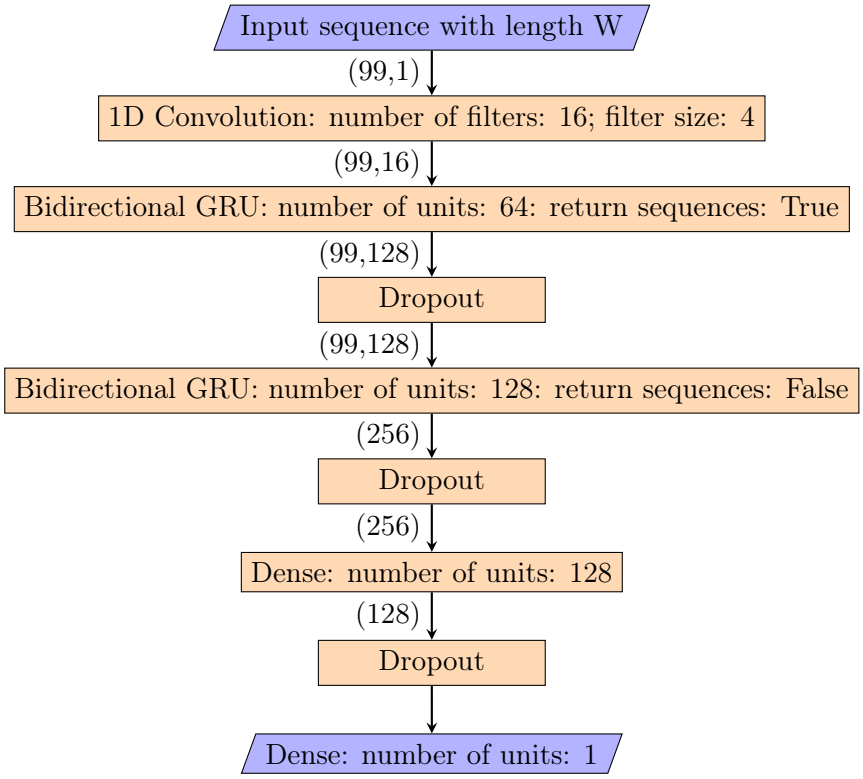

Figure 25: GRU model: Output shape of each layer is mentioned next to arrows when W is 99

5.1.5 Bidirectional Encoder Representations from Transformers

Bidirectional Encoder Representations from Transformers (BERT) is a deep learning technique that only contains the encoder part of the transformer. The input and output of the model are the same as that of sequence-to-sequence model. It learns a regression map from the main meter sequence to the corresponding target appliance sequence. Many variants of BERT are also discussed in this thesis. The architecture of the original BERT is shown in Figure 26.

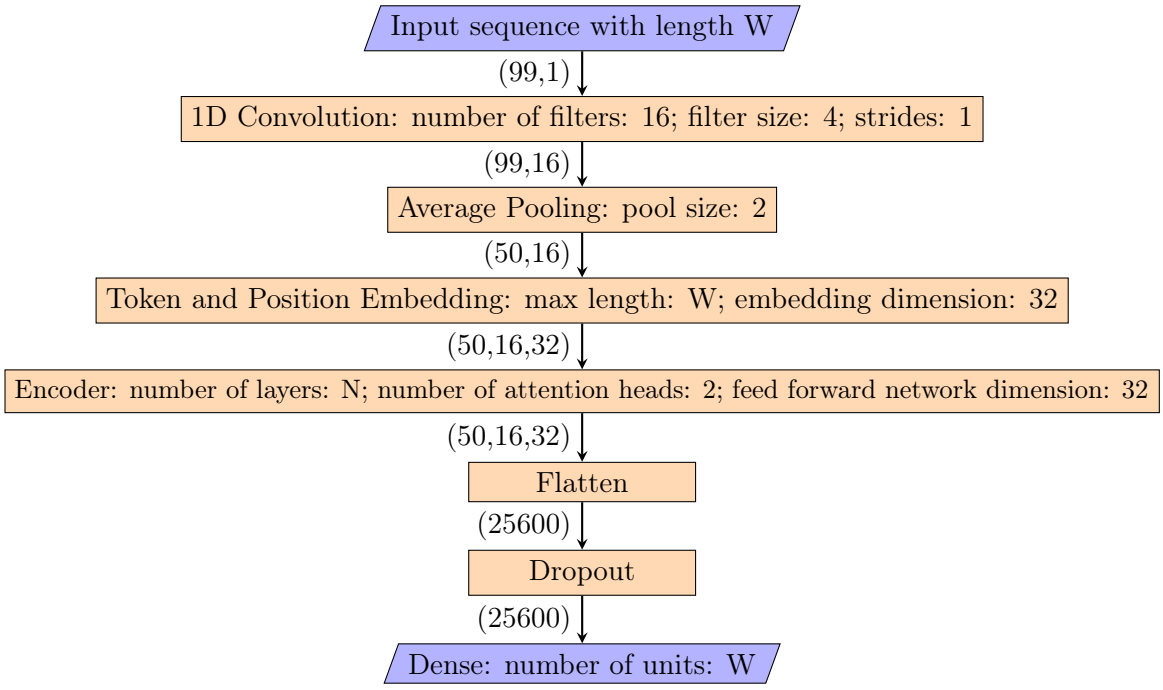

Figure 26: BERT model: Output shape of each layer is mentioned next to arrows when W is 99

5.2 Evaluation Metrics

Mean Absolute Error

Mean Absolute Error(MAE) is an error metric corresponding to the arithmetic mean of the absolute error loss. If $\hat{y}_i$ is the predicted value of the i-th sample, and y_i is the corresponding true value, then the MAE over n samples is defined as

$$MAE(y, \hat{y}) = \frac{1}{n_{sample}} \sum_{i=0}^{n_{sample}-1} |y_i - \hat{y}_i| .$$

Root mean square error

Root mean square error (RMSE) is an error metric that computes the square root of the average of squared differences between prediction and actual observation. If $\hat{y}_i$ is the predicted value of the i-th sample, and y_i is the corresponding true value, then the RMSE over n samples is defined as

$$RMSE(y, \hat{y}) = \sqrt{\frac{1}{n_{sample}} \sum_{i=0}^{n_{sample}-1} (y_i - \hat{y}_i)^2}.$$

Normalized Disaggregation Error

The comparison between appliances having a high difference in power consumption is problematic using RMSE. To compare the error between the different appliances, Normalized Disaggregation Error (NDE) metric is used. If $\hat{y}_i$ is the predicted value of the i-th sample, and y_i is the corresponding true value, then the NDE over n samples is defined as

$$NDE(y, \hat{y}) = \sqrt{\frac{\sum_{i=0}^{n_{sample}-1} (y_i - \hat{y}_i)^2}{\sum_{i=0}^{n_{sample}-1} (y_i)^2}}$$

NDE allows the comparison between errors in the prediction of energy consumed in charging vehicles and heat-pumps in different houses in the synthetic dataset even though they have different input settings, since the error is normalized.

Confusion matrix

The confusion matrix is a table that contains four entries as defined below. Since the models are designed to predict the energy consumed by an appliance and not whether the appliance is on or off, the appliance is considered to be on if its value is above a certain threshold(T). Threshold T is set to 10% of the maximum power of the appliance, which was seen in the ground truth.

Let the predicted value of energy consumed by a device by a model be P and the ground truth value of energy consumed by a device be G.

- The prediction at a particular time point is considered to be True Positive (TP) if both P and G are above T.

- The prediction at a particular time point is considered to be True Negative (TN) if both P and G are below T.

- The prediction at a particular time point is considered to be False Positive (FP) if P is above T but G is below T.

- The prediction at a particular time point is considered to be False Negative (FN) if P is below T but G is above T.

Accuracy, Precision, Recall, F1-Score

The accuracy of the model is the ratio of correctly predicted observations to the total number of observations. Using the results from the confusion matrix the accuracy is given by

$$Accuracy = \frac{(TP + TN)}{(TP + FP + FN + TN)} .$$

The precision of the model is the ratio of correctly predicted positive observations to the total predicted positive observations. It is defined as

$$Precision = \frac{TP}{(TP + FP)} .$$

The recall of the model is the ratio of correctly predicted positive observations to the total actual positive observations. It is defined as

$$Recall = \frac{TP}{(TP + FN)} .$$

The F1-score (F1) of the model is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. F1 is a better metric than accuracy when there is an uneven class distribution as the accuracy of the model can be largely contributed by a large number of True Negatives (if the device is mostly off). F1 is defined as

$$F1 = 2 * \frac{(Precision * Recall)}{(Precision + Recall)} .$$

6 Experiments

This section explains the experiments we did to compare the performance of different models using the two datasets. In Section 6.1 we explain the experimental setup. In Section 6.2 we compare the performance of the NILM algorithms on the Synpro Dataset. In Section 6.3 we compare the performance of the NILM algorithms on the Dataport Dataset. In Section 6.4 we compare the performance of multi-input models using weather as additional input to the Synpro dataset. In Section 6.5 we compare the performance of multi-output models by using the same model to predict more than one appliance at a time on the Synpro dataset. In Section 6.6 we convert the BERT model into the BERT2point model and evaluate its performance. In Section 6.7 we compare the performance of Seq2Point and BERT algorithms when tested on an unseen house. In Section 6.8 we compare the performance of Seq2Point and BERT algorithms when trained on both the datasets and tested on the Dataport dataset. In Section 6.9 we compare the performances of Seq2Point and BERT algorithms in electric vehicle charging event detection.

6.1 Set-up

The training for experiments is completed using an NVIDIA Quadro RTX 8000 GPU. The dataset is first converted into NILMTK format and then data is split into training and test data depending on the experiment. 15% of the training data is used as the validation dataset. The optimizer used is the Adam optimizer and the

loss function used is the mean square error in all the deep learning based algorithms. Early stopping is used while training these neural networks to avoid overfitting. When no improvement was seen after 15 epochs, the training would stop and the best model would be saved. The sample rate of 15 minutes is used in all experiments. The training time for the different NILM algorithms when the training period is set to nine months is shown in Table 2. Multiple runs for the NILM algorithms are needed because these algorithms are stochastic due to the random initialization of weights. The results of Seq2point, Seq2seq, and RNN are averaged across five runs. Since the GRU models and the BERT models with 1 encoder layer require approximately 1 hour of training time for every model, its results are averaged across two runs. The BERT models with 4 and 6 encoder layers are only run once as these models require approximately 3 hours or more to train.

Algorithm	Number of epochs	Average time taken per epoch	Average total training time
Seq2seq	50	1 Second	50 Seconds
Seq2point	50	1 Second	50 Seconds
RNN	50	10 Seconds	500 Seconds
GRU	50	67 Seconds	3350 Seconds
BERT with 1 encoder layer	50	24 Seconds	4800 Seconds
BERT with 4 encoder layers	50	47 Seconds	9400 Seconds
BERT with 6 encoder layers	50	71 Seconds	14200 Seconds

Table 2: Training time of different NILM algorithms when training period is set to nine months

6.2 Comparison between the NILM algorithms on the Synpro dataset

The performance in terms of the NDE metric of the different algorithms in predicting energy consumed while charging EV and energy consumed by heat-pumps using the Synpro dataset are shown in Table 3 and Table 4 respectively. The default hyper-parameter settings values are used in this experiment with sequence length set to 99. Training is done in the first 9 months of the year 2017 and testing is

done in the last 3 months of the year 2017 for all the different algorithms. Since no default value for the number of encoder layers when using the BERT algorithm was mentioned in the BERT paper (Yue et al. (2020)), BERT models with 1, 4 and 6 encoder layers are used. The results across different houses can be compared because the error is normalized when using the NDE metric.

House	RNN NDE	Seq2seq NDE	Seq2point NDE	GRU NDE	BERT-1 NDE	BERT-4 NDE	BERT-6 NDE
1	0.406	0.321	**0.315**	0.434	0.452	0.408	0.453
2	0.417	0.328	**0.285**	0.394	0.374	0.367	0.335
3	0.601	0.298	**0.238**	0.242	0.259	0.249	0.259
4	0.672	**0.585**	0.597	0.648	0.701	0.697	0.652
5	0.614	0.311	**0.306**	0.361	0.329	0.332	0.302
6	0.765	0.597	**0.591**	0.621	0.633	0.628	0.614
7	0.518	0.335	**0.253**	0.319	0.281	0.288	0.261
8	0.751	0.555	0.556	0.579	0.530	0.585	**0.537**
9	0.542	0.329	**0.263**	0.320	0.311	0.345	0.326
10	0.641	0.442	**0.441**	0.611	0.587	0.562	0.562
11	0.752	0.305	**0.259**	0.314	0.345	0.344	0.346
12	0.607	0.416	**0.403**	0.491	0.457	0.438	0.416

Table 3: Results of NILM algorithms using Synpro dataset on electric vehicle charging power prediction where BERT-1, BERT-4, BERT-6 are BERT models with 1,4 and 6 encoder layers respectively

House	RNN NDE	Seq2seq NDE	Seq2point NDE	GRU NDE	BERT-1 NDE	BERT-4 NDE	BERT-6 NDE
1	0.437	0.273	**0.141**	0.148	0.453	0.395	0.403
2	0.688	0.300	**0.201**	0.209	0.533	0.516	0.464
3	0.625	0.254	0.149	**0.133**	0.492	0.488	0.499
4	0.544	0.268	**0.162**	0.164	0.531	0.474	0.466
5	0.738	0.238	0.141	**0.124**	0.446	0.396	0.419
6	0.612	0.239	**0.156**	0.165	0.495	0.425	0.424
7	0.473	0.251	0.163	**0.146**	0.537	0.489	0.520
8	0.716	0.318	0.244	**0.229**	0.583	0.520	0.572
9	0.594	0.334	**0.252**	0.256	0.609	0.554	0.545
10	0.626	0.316	**0.234**	0.237	0.595	0.552	0.538
11	0.546	0.299	**0.219**	0.224	0.548	0.547	0.552
12	0.484	0.393	**0.310**	0.321	0.583	0.555	0.559

Table 4: Results of NILM algorithms using Synpro dataset on heat-pump energy consumption prediction where BERT-1, BERT-4, BERT-6 are BERT models with 1,4 and 6 encoder layers respectively.

The performance of the Seq2point algorithm is better than the other algorithms in 10 out of the 12 houses when considering the EV charger appliance. The performance of all models is better when predicting the energy consumed by the heat-pump compared to predicting energy consumed while charging an EV. The GRU algorithm had comparable performance to the Seq2point algorithm in predicting energy consumed by the heat-pump. The performance of the different BERT models used is worse than the Seq2point model in 11 out of the 12 houses when considering the EV charger appliance and worse in all houses when considering the heat-pump appliance. BERT models with more than 1 encoder layer outperformed the BERT model with just 1 encoder layer in most houses. The RNN had a high variance in its evaluation metrics when it was run multiple times on the same house for the same appliance. None of

the RNN models outperform any of the Seq2point models.

All models other than the RNN model perform better in terms of predicting energy consumed by an EV in a house that contains a charger with a higher charging rate compared to a house that contains an EV charger with a lower charging rate. This is true when both houses have the same number of occupants and are of the same type. For example house 5 and house 6 have the same number of occupants and are of Multi-family house type. Predictions of the energy consumed by the EV charger at house 5 which has an EV charger that charges at 7.2 kW is better than that of house 6 which has an EV charger that charges at 3.7 kW. This is because the NILM algorithms can detect larger spikes in energy consumption when charged at a higher charging rate. Houses that have an EV charger that charges at the rate of 11 kW (Houses 3, 7, 9 and 11) have the best results in terms of predicting energy consumed by an EV using the Seq2point algorithm.

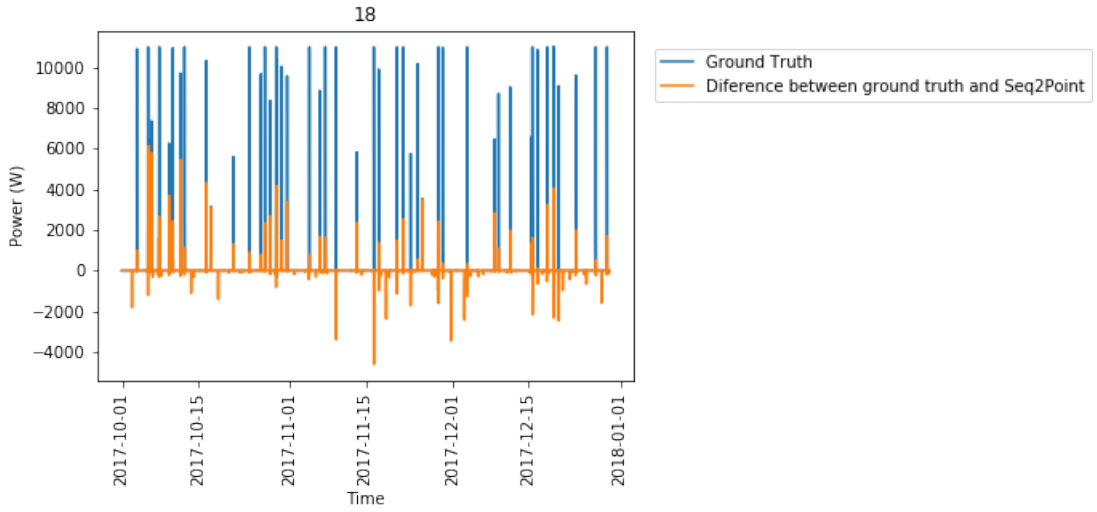

Figure 27: Results of Seq2point algorithm on house 3 of the Synpro dataset on electric vehicle charging power prediction

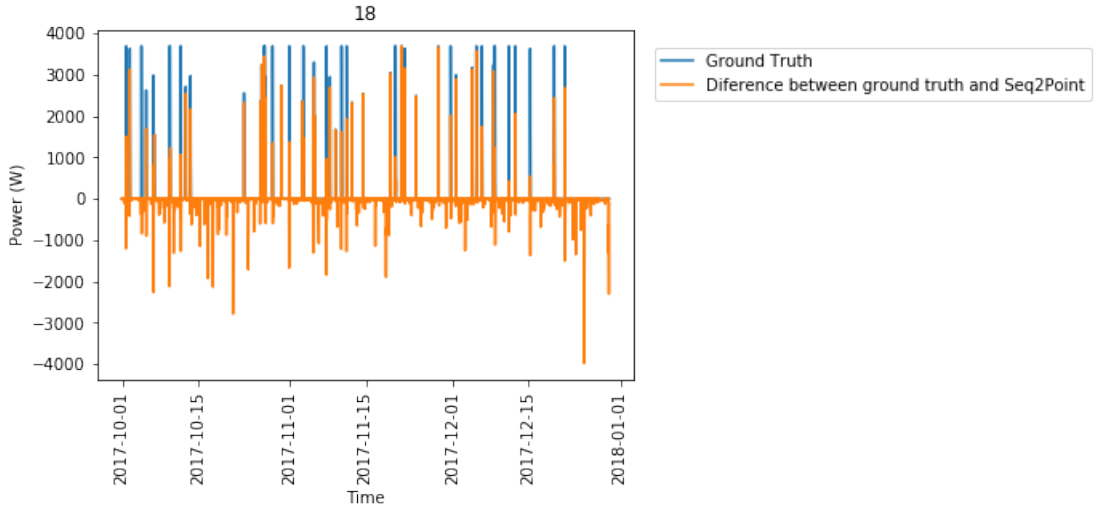

Figure 28: Results of Seq2point algorithm on house 4 of the Synpro dataset on electric vehicle charging power prediction

Figure 27 and Figure 28 shows the ground truth of the energy consumed by the EV charger and the difference between the ground truth and the prediction of the Seq2point algorithm for house 3 and 4 respectively. The Seq2point algorithm has the lowest NDE in house 3 and the highest NDE in house 4 when predicting energy consumed by an EV charger. In house 4 of the Synpro dataset, the EV charger is only used once between 15th and 1st November. However, the Seq2point algorithm continues to make large predictions in this period. This change in the pattern when the EV charger is not used is one of the reasons why its performance is poor in house 4. Another factor for this poor performance in house 4 could be because the NILM algorithms are unable to distinguish between the peaks caused by heat-pumps from the peaks caused by a 3.7 kW EV charger. Even though both house 1 and house 4 have the same type of EV charger, the higher number of occupants in house 4 results in more energy consumed by heat-pumps. This might explain the higher error in house 4 as the NILM algorithms might find it harder to distinguish between high energy consumed by the heat-pump and the EV chargers.

The BERT algorithm mentioned in Yue et al. (2020) only mentions the houses

in which training and testing are done. It does not mention exactly which dates are used. The experiments conducted in the BERT paper compare the performance of the NILM algorithms (RNN, GRU, BERT, Seq2seq) at a 6-second sample rate. They do not compare the performance of the BERT algorithm with Seq2point. One of the reasons for the drop in performance seen in this experiment could be attributed to the change in sample rate since the algorithms in this thesis are compared at a 15-minute sample rate.

6.3 Comparison between the NILM algorithms on the Dataport dataset

The performance of the different algorithms in predicting energy consumed while charging an EV using the Dataport dataset using NDE metric is shown in Table 5. The default hyper-parameter settings values are used in this experiment with sequence length set to 99. Training is done in the first 9 months of the year 2018 and testing is done in the last 3 months of the year 2018 for all the different algorithms.

House	RNN NDE	Seq2seq NDE	Seq2point NDE	GRU NDE	BERT-1 NDE	BERT-4 NDE	BERT-6 NDE
1	0.317	0.256	**0.202**	0.393	0.415	0.388	0.453
2	0.399	0.306	**0.229**	0.253	0.417	0.369	0.335
3	0.503	0.404	**0.371**	0.398	0.478	0.444	0.438
4	0.364	0.310	**0.250**	0.262	0.316	0.290	0.302
5	1.12	1.03	1.14	1.046	1.072	**1.009**	1.041
6	0.689	0.539	**0.524**	0.608	0.633	0.538	0.561

Table 5: Results of NILM algorithms using Dataport dataset on electric vehicle charging power prediction where BERT-1, BERT-4, BERT-6 are BERT models with 1,4 and 6 encoder layers respectively

The Seq2point algorithm outperformed the other algorithms for this dataset as well when predicting energy consumed while charging an EV. The performance of all the algorithms in house 5 of this dataset was poor. Electric vehicle charger was not used after November 15^{th} and thus it was not used for approximately half of the test dataset. This sudden change in the pattern of usage resulted in the bad performance of these algorithms in house 5.

6.4 Multi-input models using weather as additional input on the Synpro dataset

This experiment is similar to the experiment in Section 6.2 but additional input of weather information (temperature) is used as the dataset has seasonality as shown in Section 4. Experiments are only conducted for the Seq2point and BERT models as the Seq2point model has the best performance in the previous experiments and Transformer based models are state-of-the-art for many tasks. The best performing model after comparing the performance of BERT with 1, 4 and 6 encoder layers in terms of NDE for each house and the results averaged across 5 runs for Seq2point are used in this experiment. Sequence length and learning rate hyperparameters are optimized for the Seq2point algorithm. The learning rate hyperparameter is optimized for the BERT model. The results of hyperparameter optimization for both algorithms are shown in the Appendix (Section 9). The performance of these algorithms in predicting energy consumed while charging EV and energy consumed by heat-pumps using the Synpro dataset are shown in Table 6 and Table 7 respectively. Additionally, these tables also show the performance of the best BERT model and Seq2point model from Table 3 and Table 7 to compare their performances.

House	multi-input Seq2point			Original Seq2point	multi-input BERT			Original BERT
	RMSE (Watt)	MAE (Watt)	NDE	NDE	RMSE (Watt)	MAE (Watt)	NDE	NDE
1	205	39	0.258	0.315	289	86	0.394	0.408
2	289	49	0.253	0.285	456	135	0.399	0.335
3	198	20	0.175	0.238	369	62	0.331	0.249
4	243	38	0.500	0.597	322	77	0.648	0.652
5	201	29	0.230	0.306	301	61	0.339	0.302
6	299	70	0.52	0.591	370	109	0.647	0.614
7	208	29	0.211	0.253	360	62	0.360	0.261
8	377	62	0.440	0.556	471	109	0.551	0.537
9	287	46	0.225	0.263	518	120	0.406	0.311
10	319	92	0.375	0.441	502	256	0.591	0.562
11	229	26	0.234	0.259	365	65	0.371	0.344
12	386	60	0.367	0.403	502	119	0.476	0.416

Table 6: Results of NILM algorithms using the Synpro dataset on electric vehicle charging power prediction with temperature time-series data used as additional input. The results from Table 3 are added for comparison.

House	multi-input Seq2point			Original Seq2point	multi-input BERT			Original BERT
	RMSE (Watt)	MAE (Watt)	NDE	NDE	RMSE (Watt)	MAE (Watt)	NDE	NDE
1	192	52	0.134	0.141	296	183	0.205	0.403
2	272	84	0.178	0.201	397	242	0.258	0.464
3	218	74	0.142	0.149	331	206	0.217	0.499
4	211	52	0.136	0.162	337	211	0.218	0.466
5	186	49	0.116	0.141	329	214	0.205	0.419
6	213	53	0.133	0.156	336	221	0.210	0.424
7	232	75	0.144	0.163	366	238	0.232	0.520
8	352	137	0.227	0.244	507	345	0.330	0.572
9	363	142	0.236	0.252	515	367	0.335	0.545
10	315	110	0.205	0.234	461	318	0.300	0.538
11	290	108	0.188	0.219	486	341	0.317	0.552
12	429	153	0.287	0.310	581	411	0.392	0.559

Table 7: Results of NILM algorithms using the Synpro dataset on heat-pump energy consumption prediction with temperature time-series data used as additional input. The results from Table 4 are added for comparison.

The additional input of weather information (temperature) improved the performance of Seq2point in both appliances. BERT algorithms' performance improved

considerably in all houses for the heat-pump appliance. However, BERT's performance did not improve in most houses for predicting the energy consumed while charging electric vehicles.

6.5 Converting the model into a multi-output model by using the same model to predict the energy consumption of more than one appliance at a time on the Synpro dataset

In this experiment, the original Seq2point and BERT models are converted into multi-output models that can be used to predict the energy consumption of more than one appliance at a time. In this experiment, these models are used to predict energy consumed while charging electric vehicles and energy consumed by heat-pumps concurrently. The best performing model after comparing the performance of BERT with 1, 4 and 6 encoder layers in terms of NDE for each house and the results averaged across 5 runs for Seq2point are used in this experiment. The performance of BERT and Seq2point multi-output algorithms in predicting both the energy consumed while charging EV and the energy consumed by heat-pumps using the Synpro dataset are shown in Table 8 and Table 9 respectively.

House	multi-output Seq2point			Original Seq2point	multi-output BERT			Original BERT
	RMSE (Watt)	MAE (Watt)	NDE	NDE	RMSE (Watt)	MAE (Watt)	NDE	NDE
1	233	54	0.319	0.315	336	128	0.459	0.408
2	333	78	0.291	0.285	394	132	0.345	0.335
3	295	43	0.264	0.238	321	70	0.287	0.249
4	298	67	0.601	0.597	320	104	0.644	0.652
5	269	56	0.303	0.306	287	74	0.323	0.302
6	337	78	0.588	0.591	369	118	0.645	0.614
7	323	48	0.323	0.253	300	77	0.300	0.261
8	474	109	0.555	0.556	449	113	0.525	0.537
9	394	91	0.309	0.263	554	133	0.435	0.311
10	278	150	0.445	0.441	479	252	0.564	0.562
11	325	56	0.331	0.259	361	83	0.367	0.344
12	441	194	0.419	0.403	447	121	0.424	0.416

Table 8: Results of NILM algorithms using the Synpro dataset on electric vehicle charging power prediction using a multi-output model. The results from Table 3 are added for comparison.

House	multi-output Seq2point			Original Seq2point	multi-output BERT			Original BERT
	RMSE (Watt)	MAE (Watt)	NDE	NDE	RMSE (Watt)	MAE (Watt)	NDE	NDE
1	242	102	0.168	0.141	587	439	0.408	0.403
2	341	165	0.222	0.201	716	574	0.466	0.464
3	283	142	0.185	0.149	745	597	0.488	0.499
4	280	129	0.180	0.162	731	577	0.472	0.466
5	236	102	0.147	0.141	669	511	0.418	0.419
6	267	116	0.167	0.156	707	572	0.443	0.424
7	282	131	0.178	0.163	781	621	0.494	0.520
8	413	211	0.267	0.244	859	718	0.555	0.572
9	398	183	0.260	0.252	899	750	0.586	0.545
10	381	220	0.247	0.234	868	726	0.564	0.538
11	371	199	0.241	0.219	830	680	0.540	0.552
12	489	272	0.330	0.310	869	708	0.586	0.559

Table 9: Results of NILM algorithms using the Synpro dataset on heat-pump energy consumption prediction using a multi-output model. The results from Table 4 are added for comparison.

The performance of Seq2point was only slightly worse when comparing it with the original Seq2point model when predicting both appliances using just one model. The BERT algorithm's performance was similar to that of the original BERT model

in both appliances. These multi-output algorithms can be used to save training time as one model can be trained for more than one appliance.

6.6 Converting the BERT model to the BERT2Point model

The original BERT model has similar input and output architecture as the Seq2seq model. In this experiment, the original BERT model is converted to the BERT2Point model by changing the output to predict only the midpoint of the sequence just like the Seq2point model. The original BERT model with 4 encoder layers was converted to the BERT2point algorithm. The performance of the BERT2point algorithm in predicting both the energy consumed while charging EV and the energy consumed by heat-pumps using the Synpro dataset are shown in Table 10 and Table 11 respectively.

House	BERT2point			Original BERT
	RMSE (Watt)	MAE (Watt)	NDE	NDE
1	385	149	0.529	0.408
2	444	118	0.388	0.335
3	290	39	0.260	0.249
4	351	97	0.707	0.652
5	355	86	0.400	0.302
6	379	106	0.662	0.614
7	326	57	0.325	0.261
8	491	115	0.574	0.537
9	444	77	0.348	0.311
10	510	212	0.600	0.562
11	353	50	0.359	0.344
12	495	118	0.470	0.416

Table 10: Results of BERT2point using the Synpro dataset on electric vehicle charging power prediction. The results from Table 3 are added for comparison.

House	BERT2point			Original BERT
	RMSE (Watt)	MAE (Watt)	NDE	NDE
1	647	431	0.449	0.403
2	801	623	0.521	0.464
3	825	630	0.540	0.499
4	767	584	0.495	0.466
5	875	763	0.546	0.419
6	786	574	0.546	0.424
7	823	639	0.520	0.520
8	863	686	0.558	0.572
9	873	682	0.569	0.545
10	1048	966	0.681	0.538
11	860	666	0.560	0.552
12	897	729	0.605	0.559

Table 11: Results of BERT2point using the Synpro dataset on heat-pump energy consumption prediction. The results from Table 4 are added for comparison.

Converting the BERT algorithm to the BERT2point algorithm failed in improving the performance of the algorithm. In all houses, considering both appliances, BERT2point's performance was worse than the original BERT algorithm.

6.7 Performance of the Seq2Point and the BERT algorithms when tested on an unseen house

The experiments that were carried out so far were trained and tested on the same house of the dataset. This experiment is carried out to test the ability of these algorithms to generalize when tested on an unseen house. The ability to generalize to similar house types, with the same EV charging rate and different numbers of occupants is analyzed in this experiment. The best model for the BERT and Seq2point algorithm for each of the houses in the Synpro dataset from the experiment in Section 6.2 is used as the trained model. Results of Seq2point and BERT algorithms using the Synpro dataset on predicting energy consumed by an EV charging and heat-pump when trained and tested on different houses are shown in Table 12 and Table 13 respectively. House 2 and House 3 of the Synpro dataset do not have any similar houses in the dataset as there are no other houses of type 'Single Family House' where the charging rate of EV is 7kW or 11 kW.

Serial Number	House Train	House Test	Seq2point			BERT		
			RMSE (Watt)	MAE (Watt)	NDE	RMSE (Watt)	MAE (Watt)	NDE
1	1	4	355	77	0.714	448	168	0.902
2	4	1	355	80	0.485	386	115	0.526
3	5	8	560	116	0.655	492	176	0.576
4	5	12	546	130	0.519	643	197	0.611
5	6	10	577	200	0.679	575	244	0.676
6	7	9	364	72	0.285	407	98	0.319
7	7	11	320	53	0.325	425	94	0.433
8	8	5	474	90	0.533	468	113	0.526
9	8	12	560	142	0.531	604	186	0.574
10	9	7	304	52	0.304	293	76	0.293
11	9	11	356	153	0.232	804	672	0.524
12	10	6	384	90	0.671	409	172	0.714
13	11	7	443	54	0.442	346	78	0.346
14	11	9	592	89	0.465	416	102	0.326
15	12	5	409	54	0.460	368	72	0.414
16	12	8	521	84	0.610	464	97	0.543

Table 12: Results of NILM algorithms using the Synpro dataset on electric vehicle charging power prediction when trained and tested on different houses

Serial Number	House Train	House Test	Seq2point			BERT		
			RMSE (Watt)	MAE (Watt)	NDE	RMSE (Watt)	MAE (Watt)	NDE
1	1	4	341	175	0.220	812	66	0.524
2	4	1	260	99	0.180	736	571	0.511
3	5	8	429	190	0.277	874	718	0.565
4	5	12	601	327	0.405	1074	940	0.724
5	6	10	480	224	0.312	978	826	0.635
6	7	9	400	167	0.261	911	775	0.594
7	7	11	363	156	0.236	903	773	0.588
8	8	5	334	144	0.209	790	643	0.493
9	8	12	507	249	0.342	912	778	0.615
10	9	7	319	136	0.202	876	713	0.554
11	9	11	356	153	0.232	804	672	0.524
12	10	6	398	166	0.250	967	808	0.606
13	11	7	326	154	0.206	957	756	0.605
14	11	9	406	187	0.265	888	731	0.579
15	12	5	480	247	0.300	1027	818	0.641
16	12	8	432	220	0.279	922	738	0.596

Table 13: Results of NILM algorithms using the Synpro dataset on heat-pump energy consumption prediction when trained and tested on different houses

The BERT algorithm outperforms the Seq2point algorithm in 8 out of the 16 tests conducted in this experiment when considering the EV charging appliance. The Seq2point algorithm outperforms the BERT algorithm in all tests when considering the heat-pump appliance. This might be because consumption patterns of the heat-pump appliances are similar in these houses but the EV usage patterns are different. However, the Seq2point algorithm when trained on House 9 and tested on House 11 performs better than when the Seq2point algorithm is trained and tested on House 11 when predicting energy consumed while charging an EV. The Seq2point algorithm can generalize well to some houses with minimal loss in performance and is unable to do so in other cases when considering EV charging appliance. Both the Seq2point and the BERT algorithms do not have a large loss in performance in all houses when predicting energy consumed by heat-pump.

6.8 Training on both datasets and testing on Dataport dataset

This experiment was run to check if the additional training data from a house of the Synpro dataset helps improve the performance of the NILM algorithms in predicting energy consumed while charging an EV in a house of the Dataport dataset. When the energy consumed by an EV charger in the Dataport dataset was visualized, it was observed that these houses had an EV charger that charged at the rate of less than 4kW. Thus, only the houses which contained an EV charger at a rate of 3.7kW from the Synpro dataset are used as additional data for this experiment (Houses 1,4,6,10). Training data consists of one of the 4 houses from the Synpro dataset for the entire year 2017 and the first 9 months of the year 2018 for each of the houses in the Dataport dataset. Testing is done for the same house of the Dataport dataset for the last 3 months of 2018. For the Seq2point algorithm, this experiment is carried out for each possible combination of the 4 houses in the Synpro dataset and the 6 houses in the Dataport dataset. Thus, the house from the Synpro dataset that had the "best" result (lowest NDE) when it was added to a house in Dataport dataset can be observed for the Seq2point algorithm. The experiment is carried out for each house in the Dataport dataset and its corresponding "best" house from the Synpro dataset when using the BERT algorithm. This was done as training for each possible combination of houses using the BERT algorithm would take many days to complete. The results of this experiment are shown in Table 14.

House Dataport	House Synpro	Seq2point			Seq2point-old	BERT			BERT-old
		RMSE (Watt)	MAE (Watt)	NDE	NDE	RMSE (Watt)	MAE (Watt)	NDE	NDE
1	6	159	33	0.195	0.202	724	259	0.886	0.388
2	6	273	69	0.277	0.229	836	326	0.849	0.335
3	6	441	131	0.381	0.371	520	98	0.826	0.438
4	1	372	78	0.227	0.250	520	98	0.318	0.290
5	4	673	176	1.120	1.140	578	170	0.961	1.009
6	10	395	118	0.509	0.524	642	298	0.827	0.538

Table 14: Results of NILM algorithms trained on both the datasets and tested on the Dataport dataset. Where Seq2point-old and BERT-old refer to results of the Seq2point model and best BERT model from Table 5

The two datasets are very different as shown in Section 4. One dataset is from the USA and is a real dataset where the weather is warm for most of the year and the other is a synthetic dataset from Germany where the weather is comparatively cold. Even though this weather difference does not affect the energy consumed by EV, it adds seasonality (as shown in Figure 18 and Figure 14) to the overall energy consumed by the house. However, the performance in 4 out of 6 houses improve with the additional data for the Seq2point algorithm. The additional data does not help improve the performance of EV energy consumption prediction in any house when the BERT algorithm is used. The additional data confuses the BERT algorithm and results in worse performance.

6.9 Performance of Seq2Point and BERT algorithms in electric vehicle charging event detection

A particular timepoint is considered to be positive in the ground truth if the electricity consumed while charging is above 10% of the maximum power consumed while charging the EV, else it is considered negative. Similarly, a particular timepoint is considered to be positive in the prediction if the electricity consumed while charging is above 10% of the maximum power consumed while charging the appliance, else it is

considered negative. The performance of Seq2Point and BERT algorithm in electric vehicle charging event detection is shown in Table 15.

House	Seq2point		BERT	
	Accuracy	F1-score	Accuracy	F1-score
1	99.0%	0.902	97.6%	0.794
2	99.0%	0.911	97.9%	0.821
3	99.8%	0.931	99.6%	0.885
4	98.5%	0.742	97.7%	0.628
5	99.7%	0.946	99.2%	0.843
6	97.6%	0.708	96.1%	0.843
7	99.6%	0.903	99.3%	0.825
8	98.6%	0.770	98.0%	0.695
9	99.4%	0.907	98.7%	0.768
10	96.5%	0.861	87.9%	0.637
11	99.5%	0.88	99.2%	0.795
12	98.9%	0.851	97.9%	0.734

Table 15: Results of NILM algorithms using Synpro dataset on electric vehicle charging event detection

The accuracy of the model in predicting charging events is very high. This is because the task of predicting if the car is charging is simpler than predicting how much energy is consumed while charging the car. As the car is normally only charged a few times a week, most events are negative (Non-charging events). The accuracy of the Seq2point algorithm has decreased compared to its performance in my project

[1]. This is because the energy consumed by the heat-pump was added to the dataset in this thesis.

7 Conclusion

Non-intrusive load monitoring is used in order to estimate the electrical consumption of individual appliances using the aggregate power meter reading. The different deep learning NILM algorithms that were used created one model per appliance. These NILM algorithms considered the aggregate time-series energy consumption of the house as input and the appliance time-series energy consumption as output. The appliances of interest in this thesis are the energy consumed while charging electric vehicles and the energy consumed by heat-pumps. Different algorithms were compared using both a synthetic and a real dataset in order to identify the best performing model. The performance of the Seq2point algorithm is better than other algorithms in predicting both appliances' energy consumption in both datasets. All models other than the RNN model perform better in terms of predicting energy consumed by an EV in a house that contains a charger with a higher charging rate compared to a house that contains an EV charger with a lower charging rate. The NILM algorithms failed to adapt to sudden behavioral changes when the EV charger was not used for long periods of time as seen in house 5 of the Dataport dataset. The performance of the Seq2point algorithm in predicting both appliances' energy consumption improved when the model also received additional weather input data (outdoor ambient temperature). Seq2point and BERT models were then converted into multi-output models to predict the energy consumption of more than one appliance using just one model. Converting the Seq2point model into a multi-output model resulted in a small loss in performance. The multi-output BERT model had a similar performance to that of the original

BERT model. Converting a model to a multi-output model can be used to reduce the training time.

The additional training data from Synpro improved the performance of the Seq2point algorithm in predicting energy consumed while charging an EV in 4 out of 6 houses in the Dataport dataset. However, the additional data did not improve the performance of the BERT model in all houses of the Dataport dataset. Converting the BERT model to BERT2point did not improve the performance of the algorithm in both appliances of interest using the Synpro dataset. The BERT and Seq2point algorithms were able to generalize when tested on an unseen house of the Synpro dataset only in some cases when considering the EV charging appliance. The BERT and Seq2point algorithms suffered a small loss in performance when they were tested on an unseen house when predicting energy consumed by heat-pumps.

8 Future Work

- **Testing the performance of an encoder-decoder transformer model for NILM tasks:** In our experiments, an encoder-only transformer model's performance was compared to other models. It would be interesting to see if adding a decoder improves the current BERT algorithm's performance.

- **Training on a larger dataset in which data is available for more than one-year:** In most of our experiments, the NILM algorithms were trained for 9 months and tested on the last three months of the year. It would be interesting to compare the performance of these algorithms when they could be trained for more than one year. Additionally, it would be interesting to compare the performances of these algorithms when tested on the other months of the year.

- **Creating a model that can be used to detect EVs that are charging at different rates and are used in different types of houses:** In most of our experiments we created one model per house per appliance in a dataset. Another possible future approach could be to test the performance of one model which is trained on multiple different types of houses and tested on an unseen house.

- **Generating a synthetic dataset which improves the performance of the model on a real dataset:** Since the collection of real data for NILM is a complex and expensive process, a synthetic dataset could be used to improve

the performance of NILM algorithms on a real dataset. The Synpro dataset and the Dataport dataset have different aggregate energy consumption patterns and EV chargers that are charged at different rates. Synthetic data can be generated that is very similar to the real data to improve the performance of NILM algorithms on a real dataset.

- **Creating a more robust algorithm that is able to handle sudden changes in patterns of usage of appliances:** The NILM algorithms failed to adapt to the changes in patterns of usage of the EV charger in house 5 of the Dataport dataset. Models should be able to adapt to changes in patterns of usage.

9 Appendix

This chapter contains some additional experiments and hyperparameter optimization experiments.

9.1 Effect of sequence length hyperparameter on the multi-input Seq2Point algorithm using the Synpro dataset

This experiment is carried out to visualize how the RMSE varies with sequence length for both the appliances of interest using the Synpro dataset. This experiment is carried out by increasing the sequence length by 10 from 39 to 149 for each of the 12 houses. Results of how the RMSE in predicting energy consumed while charging EV varies with sequence length for the first 4 houses of type Single Family house and the next 8 houses of type Multi-family house can be seen in Figure 29 and Figure 30 respectively. Results of how the RMSE varies with sequence length in predicting energy consumed by heat-pump for the first 4 houses of type Single Family house and the next 8 houses of type Multi-family house can be seen in Figure 31 and Figure 32 respectively.

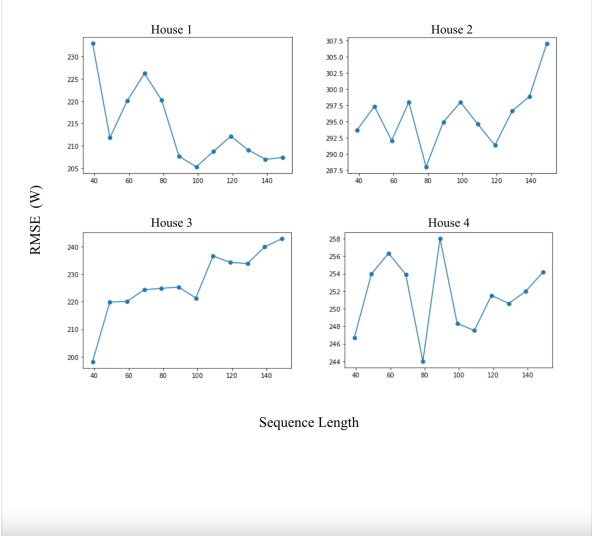

Figure 29: Sequence length vs RMSE in predicting energy consumed while charging EV for houses 1-4 of the by heat-pump for houses 5-12 of the SynproSynpro Dataset

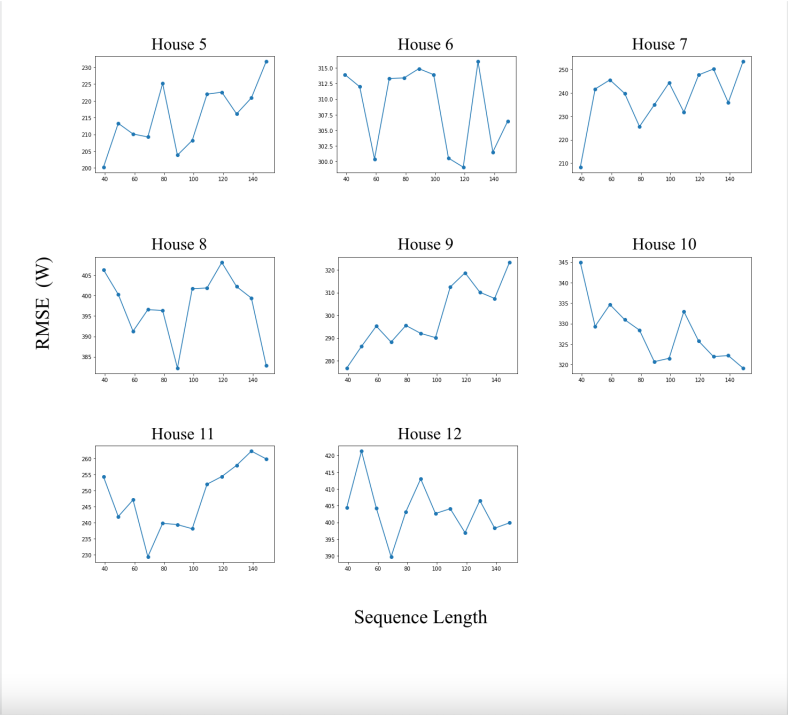

Figure 30: Sequence length vs RMSE in predicting energy consumed while charging EV for houses 5-12 of the Synpro Dataset

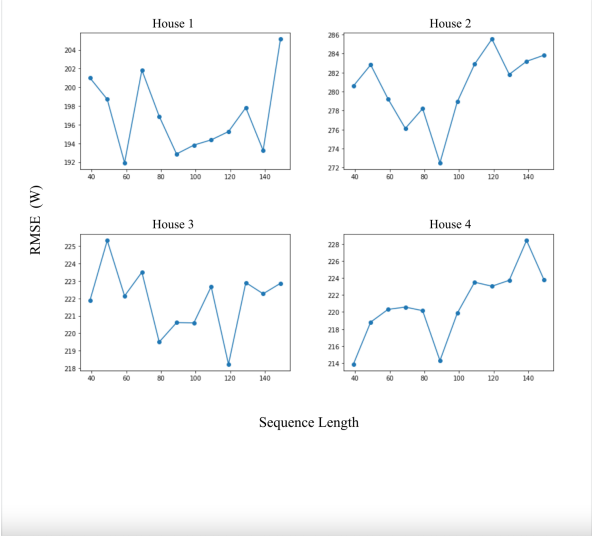

Figure 31: Sequence length vs RMSE in predicting energy consumed by heat-pump for houses 1-4 of the Synpro Dataset

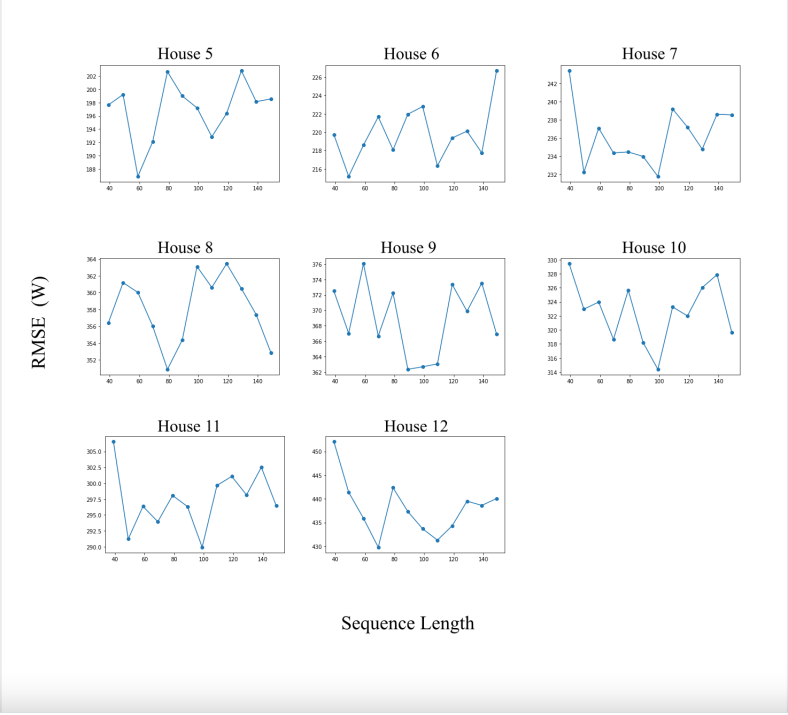

Figure 32: Sequence length vs RMSE in predicting energy consumed by heat-pump for houses 5-12 of the Synpro Dataset

In some houses, a lower sequence length than the default value (99) results in lower RMSE whereas in other houses higher sequence length than 99 results in improved performance. A direct correlation between sequence length modifications and RMSE error cannot be made for both appliances.

9.2 Effect of learning rate hyperparameter on the multi-input Seq2Point algorithm using the Synpro dataset

This experiment is carried out to test different learning rates for the muti-input Seq2point algorithm. The learning rates are varied between 0.01, 0.005, 0.001, 0.0005 and 0.0001 for each of the 12 houses. The results averaged across all houses in terms of NDE for an EV charger and the heat-pump respectively are shown in Table 16.

Learning rate	EV charger NDE	Heat-pump NDE
0.01	0.924	0.307
0.005	0.877	0.187
0.001	0.334	0.180
0.0005	0.364	0.185
0.0001	0.402	0.203

Table 16: Learning rate hyperparameter optimization for the Seq2point algorithm

The default value of 0.001 for the learning rate has the best results for the multi-input Seq2point algorithm when predicting energy consumed by both appliances of interest.

9.3 Effect of learning rate hyperparameter on the multi-input BERT algorithm using the Synpro dataset

This experiment is carried out to test different learning rates for the muti-input BERT algorithm with 4 encoder layers. Only 3 houses were selected for the BERT learning rate hyperparameter experiment because of the long training time required by the BERT models. One house for each of the different rates at which EV charging takes place is selected. House 1, house 5 and house 9 are selected. The learning rates are varied between 0.01, 0.005, 0.001, 0.0005 and 0.0001 for these 3 houses. The results averaged across these houses in terms of NDE for an EV charger and the heat-pump respectively are shown in Table 17.

Learning rate	EV charger NDE	Heat-pump NDE
0.01	0.989	0.451
0.005	0.894	0.304
0.001	0.421	0.259
0.0005	0.461	0.274
0.0001	0.491	0.301

Table 17: Learning rate hyperparameter optimization for the BERT algorithm

The default value of 0.001 for the learning rate has the best results for the multi-input BERT algorithm when predicting energy consumed by both appliances of interest.

9.4 Visualising predictions of BERT and Seq2point algorithm for a single day

Figure 33 and Figure 35 show the energy consumption of the main-meter, the heat-pump and the predictions of energy consumed by the heat-pump using BERT and Seq2point algorithms respectively. Figure 34 and Figure 36 show the energy consumption of the main-meter, an EV charger and the predictions of energy consumed by an EV-charger using the BERT and Seq2point algorithms respectively.

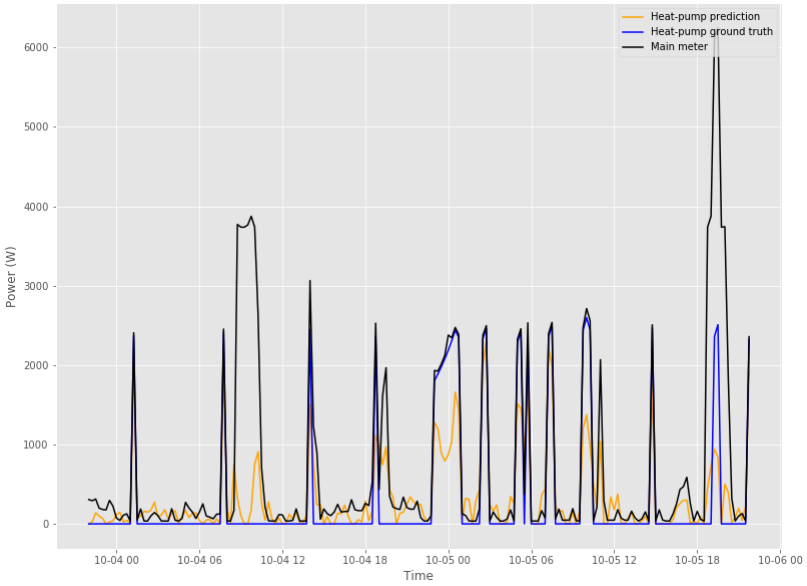

Figure 33: Energy consumption of the main meter, the heat-pump and the heat-pump prediction by BERT algorithm for a single day in house 1 of the Synpro dataset

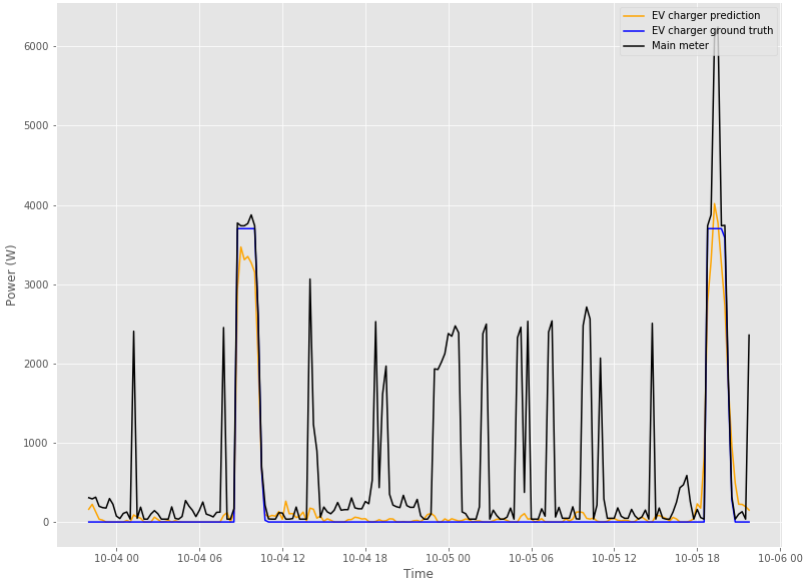

Figure 34: Energy consumption of the main meter, an EV charger and the EV charger prediction by BERT algorithm for a single day in house 1 of the Synpro dataset

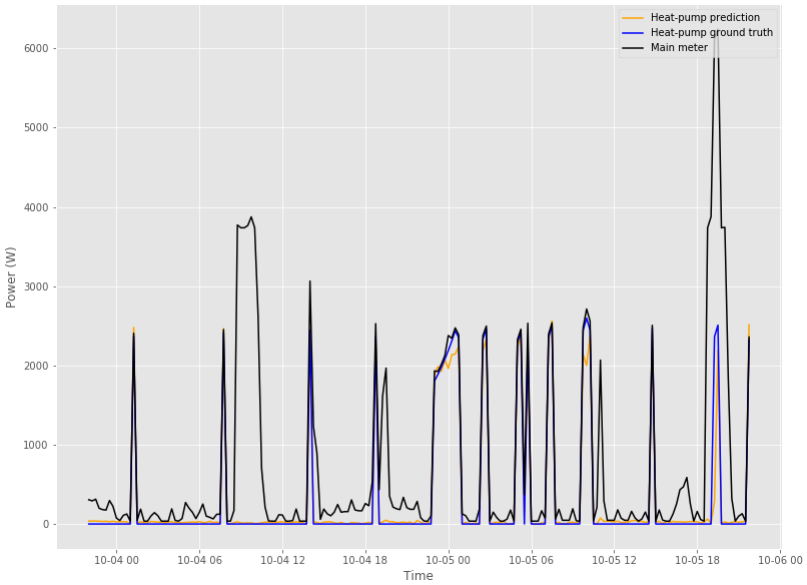

Figure 35: Energy consumption of the main meter, the heat-pump and the heat-pump prediction by Seq2point algorithm for a single day in house 1 of the Synpro dataset

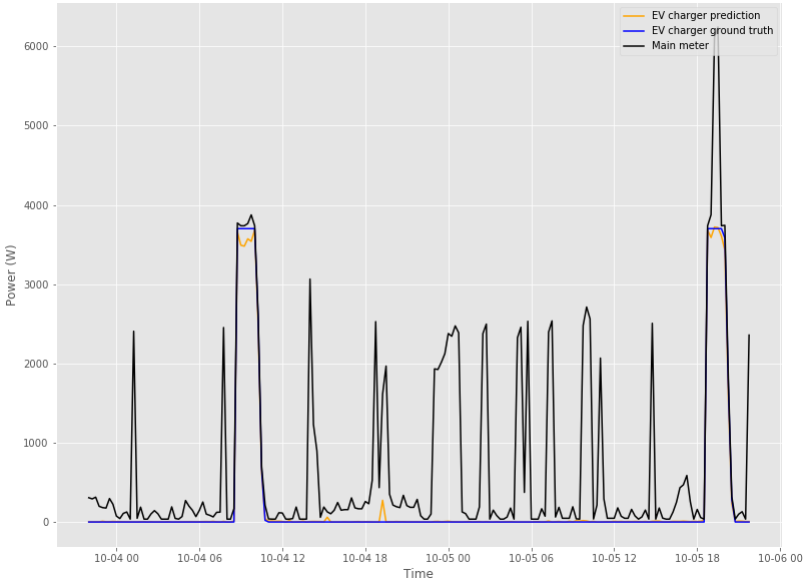

Figure 36: Energy consumption of the main meter, an EV charger and the EV charger prediction by Seq2point algorithm for a single day in house 1 of the Synpro dataset

Both Seq2point and BERT models are able to correctly predict both the peaks caused when an EV is charged. The BERT model fails to correctly predict the magnitude of energy consumed when the heat-pump is used in many cases. The BERT model falsely predicts many small peaks in energy consumption when the heat-pump is not in use. The BERT model also predicts the energy consumed by the heat-pump to be greater than the main-meter consumption in some cases. However, the Seq2point model is able to correctly predict most peaks and their magnitudes caused by the heat-pump.

Bibliography

[1] K. R. Rohit, "Detection of electric vehicle charging events using non-intrusive load monitoring," 2022. Available at https://ad-blog.cs.uni-freiburg.de/post/detection-of-electric-vehicle-charging-events-using-non-intrusive-load-monitoring/.

[2] S. Darby, "The effectiveness of feedback on energy consumption," *A Review for DEFRA of the Literature on Metering, Billing and direct Displays*, vol. 486, 01 2006.

[3] K. Carrie Armel, A. Gupta, G. Shrimali, and A. Albert, "Is disaggregation the holy grail of energy efficiency? the case of electricity," *Energy Policy*, vol. 52, pp. 213–234, 2013. Special Section: Transition Pathways to a Low Carbon Economy.

[4] D. Pujić, M. Jelić, N. Tomasevic, and M. Batic, *Chapter 10 Case Study from the Energy Domain*, pp. 165–180. 07 2020.

[5] "Verbraucherzentrale." https://www.verbraucherzentrale.de. Accessed: 2022-04-25.

[6] "Marktstammdatenregister." https://www.marktstammdatenregister.de/MaStR. Accessed: 2022-04-25.

[7] G. Hart, "Nonintrusive appliance load monitoring," *Proceedings of the IEEE*, vol. 80, no. 12, pp. 1870–1891, 1992.

[8] M. Figueiredo, A. De Almeida, and B. Ribeiro, "An experimental study on electrical signature identification of non-intrusive load monitoring (nilm) systems," pp. 31–40, 01 2011.

[9] M. Dong, P. C. M. Meira, W. Xu, and C. Y. Chung, "Non-intrusive signature extraction for major residential loads," *IEEE Transactions on Smart Grid*, vol. 4, no. 3, pp. 1421–1430, 2013.

[10] O. Parson, S. Ghosh, M. Weal, and A. Rogers, "Using hidden markov models for iterative non-intrusive appliance monitoring," 12 2011.

[11] J. Z. Kolter and T. Jaakkola, "Approximate inference in additive factorial hmms with application to energy disaggregation," in *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics* (N. D. Lawrence and M. Girolami, eds.), vol. 22 of *Proceedings of Machine Learning Research*, (La Palma, Canary Islands), pp. 1472–1482, PMLR, 21–23 Apr 2012.

[12] R. Bonfigli, E. Principi, M. Fagiani, M. Severini, S. Squartini, and F. Piazza, "Non-intrusive load monitoring by using active and reactive power in additive factorial hidden markov models," *Applied Energy*, vol. 208, pp. 1590–1607, 2017.

[13] J. Kelly and W. Knottenbelt, "Neural NILM," in *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, ACM, nov 2015.

[14] T.-T.-H. Le, J. Kim, and H. Kim, "Classification performance using gated recurrent unit recurrent neural network on energy disaggregation," in *2016 International Conference on Machine Learning and Cybernetics (ICMLC)*, vol. 1, pp. 105–110, 2016.

[15] H. Rafiq, H. Zhang, H. Li, and M. K. Ochani, "Regularized lstm based deep learning model: First step towards real-time non-intrusive load monitoring," in *2018 IEEE International Conference on Smart Energy Grid Engineering (SEGE)*, pp. 234–239, 2018.

[16] A. Wibawa, A. Utama, H. Elmunsyah, U. Pujianto, F. Dwiyanto, and L. Hernandez, "Time-series analysis with smoothed convolutional neural networkg," vol. 9, 2022.

[17] C. Zhang, M. Zhong, Z. Wang, N. Goddard, and C. Sutton, "Sequence-to-point learning with neural networks for nonintrusive load monitoring," 2016.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017.

[19] Z. Yue, C. R. Witzig, D. Jorde, and H.-A. Jacobsen, "Bert4nilm: A bidirectional transformer model for non-intrusive load monitoring," *Proceedings of the 5th International Workshop on Non-Intrusive Load Monitoring*, 2020.

[20] D. Precioso Garcelán and D. Gomez-Ullate, "Nilm as a regression versus classification problem: the importance of thresholding," 10 2020.

[21] N. Batra, J. Kelly, O. Parson, H. Dutta, W. Knottenbelt, A. Rogers, A. Singh, and M. Srivastava, "NILMTK," in *Proceedings of the 5th international conference on Future energy systems*, ACM, jun 2014.

[22] J. Kolter and M. Johnson, "Redd: A public data set for energy disaggregation research," *Artif. Intell.*, vol. 25, 01 2011.

[23] N. Batra, R. Kukunuri, A. Pandey, R. Malakar, R. Kumar, O. Krystalakos, M. Zhong, P. Meira, and O. Parson, "Towards reproducible state-of-the-art energy disaggregation," in *Proceedings of the 6th ACM International Conference*

on Systems for Energy-Efficient Buildings, Cities, and Transportation, BuildSys '19, (New York, NY, USA), p. 193–202, Association for Computing Machinery, 2019.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[25] "How to use convolutional neural networks for time series classification." `https://towardsdatascience.com/how-to-use-convolutional-neural-networks-for-time-series-classification-56b1b0a07a57`. Accessed: 2022-10-28.

[26] W. Feng, N. Guan, Y. Li, X. Zhang, and Z. Luo, "Audio visual speech recognition with multimodal recurrent neural networks," in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 681–688, 2017.

[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[28] "Lstm recurrent neural networks — how to teach a network to remember the past." `https://towardsdatascience.com/lstm-recurrent-neural-networks-how-to-teach-a-network-to-remember-the-past-55e54c2ff22e`. Accessed: 2022-10-28.

[29] "Gru recurrent neural networks — a smart way to predict sequences in python." `https://towardsdatascience.com/gru-recurrent-neural-networks-a-smart-way-to-predict-sequences-in-python-80864e4fe9f6`. Accessed: 2022-10-28.

[30] D. Fischer, A. Härtl, and B. Wille-Haussmann, "Model for electric load profiles with high time resolution for german households," *Energy and Buildings*, vol. 92, pp. 170–179, 2015.

[31] O. Parson, G. Fisher, A. Hersey, N. Batra, J. Kelly, A. Singh, W. Knottenbelt, and A. Rogers, "Dataport and nilmtk: A building data set designed for non-intrusive load monitoring," in *2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 210–214, 2015.