DEPARTMENT OF COMPUTER SCIENCE

Non-Intrusive Load Monitoring based on Residential Electricity Consumption

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Bachelor of Science in the Faculty of Engineering.

Wednesday 3rd May, 2023

Abstract

Approximately a quarter of the global electric energy consumption is thought to be consumed for residential purposes, and this figure has been rising in recent years. The usage of smart meters to provide real-time energy consumption feedback can cut energy use by up to 17% and it has been demonstrated that appliance-level insights result in even greater energy usage reductions.

Energy disaggregation, also referred to as non-intrusive load monitoring (NILM), is the process of inferring appliance-level energy consumption from the aggregated load drawn by a household. Deep learning has become increasingly popular for performing NILM due to outperforming traditional approaches and overcoming some of the setbacks inherent to them, such as the need for expert knowledge. Techniques like pattern matching and blind source separation often have difficulties in a variety of real-life scenarios due to factors such as measurement equipment noise and substantial variations in power signatures between appliances of the same type.

In this dissertation, a deep learning approach is proposed, with its main component being a deep neural network composed of convolutional, long short-term memory and dense layers as well as regularisation and normalisation blocks. The method also contained pre- and post-processing stages and it is trained and tested on data from four distinct appliances from the reference energy disaggregation data set (REDD), the most commonly used data set in NILM research. The proposed approach is shown to achieve state-of-the-art results, with performance metrics and training times being improved by as much as 41% and 88%, respectively.

Some of the main achievements and contributions of this dissertation are listed below:

- Learnt how to apply deep learning to a real-life problem, namely energy disaggregation, as a third-year Bachelor's student without a background in the field; worthy of note is that BSc students are not eligible to enrol in the deep learning unit.
- Engaged with NILM researchers from an external organisation and entered into productive dialogues with code and pre-processed data being shared.
- Remarkably improved the approach suggested by Zhang et al. [64], the baseline for seq2point energy disaggregation approaches.
- Obtained state-of-the-art results on the reference energy disaggregation data set, the most popular data set within the NILM research community.

Declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Taught Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, this work is my own work. Work done in collaboration with, or with the assistance of others, is indicated as such. I have identified all material in this dissertation which is not my own work through appropriate referencing and acknowledgement. Where I have quoted or otherwise incorporated material which is the work of others, I have included the source in the references. Any views expressed in the dissertation, other than referenced material, are those of the author.

 Wednesday 3rd May, 2023

Contents

1 Introduction **1**
- 1.1 Overview . . . 1
- 1.2 Motivation . . . 1
- 1.3 Key Aims and Objectives . . . 2
- 1.4 Summary . . . 2

2 Background **3**
- 2.1 Overview . . . 3
- 2.2 Load Monitoring . . . 3
- 2.3 Energy Disaggregation through Machine Learning . . . 6
- 2.4 Energy Disaggregation through Other Common Approaches . . . 14
- 2.5 Performance Metrics in NILM . . . 14
- 2.6 Main Anticipated Challenges . . . 16
- 2.7 Summary . . . 16

3 Project Execution **17**
- 3.1 Overview . . . 17
- 3.2 Proposed ANN Architecture . . . 17
- 3.3 Data Set . . . 18
- 3.4 Pre-Processing Stage . . . 18
- 3.5 Data Splits . . . 19
- 3.6 Sliding Window and Batch Processing Methods . . . 21
- 3.7 Post-Processing Stage . . . 21
- 3.8 Summary . . . 22

4 Results **23**
- 4.1 Overview . . . 23
- 4.2 Microwave Network Results . . . 23
- 4.3 Fridge Network Results . . . 25
- 4.4 Dishwasher Network Results . . . 26
- 4.5 Washing Machine Network Results . . . 27
- 4.6 Comparison with Other NILM Research . . . 28
- 4.7 Summary . . . 29

5 Critical Evaluation **30**
- 5.1 Overview . . . 30
- 5.2 Proposed ANN Architecture . . . 30
- 5.3 Data Set . . . 31
- 5.4 Pre-Processing Stage . . . 31
- 5.5 Data Splits . . . 31
- 5.6 Sliding Window Method . . . 31
- 5.7 Batch Processing Method . . . 32
- 5.8 Training and Testing Processes . . . 32
- 5.9 Post-Processing Stage . . . 33
- 5.10 Summary . . . 33

6 Conclusion **34**

List of Figures

2.1 Energy consumption signatures of the four different types: a) Fridge (type IV); b) Washing machine (type II); c) Fan (type I); d) Laptop (type III) [13] . . . 3
2.2 Typical load monitoring approaches [54] . . . 4
2.3 Energy disaggregation [46] . . . 5
2.4 Typical workflow of energy disaggregation approaches [54] . . . 5
2.5 Structure of different hidden Markov models: a) simple HMM; b) factorial HMM [17] . . . 6
2.6 Computation steps in a perceptron with the sigmoid activation function [42] . . . 7
2.7 Structure of an LSTM block and its different gates [60] . . . 8
2.8 Structure of a convolutional neural network with 2D inputs and 1D kernels [48] . . . 11
2.9 Architecture of the seq2seq and seq2point models proposed by Zhang et al. [64] . . . 13
2.10 Architecture of the LDwA model proposed by Piccialli et al. [47] . . . 13

3.1 Architecture of the proposed energy disaggregation model . . . 17
3.2 Structure of the proposed normalisation block . . . 18
3.3 Data splits used for the microwave network . . . 20
3.4 Data splits used for the fridge network . . . 20
3.5 Data splits used for the dishwasher network . . . 20
3.6 Data splits used for the washing machine network . . . 20

4.1 Results on the testing set and training history of the microwave network . . . 24
4.2 Results on the testing set and training history of the fridge network . . . 25
4.3 Results on the testing set and training history of the dishwasher network . . . 26
4.4 Results on the testing set and training history of the washing machine network . . . 27

List of Tables

3.1 Values used to standardise the data for each target appliance network 19
3.2 Training and testing houses used in each target appliance network 19
3.3 Input window length used in each target appliance network 21
3.4 Predictions threshold used in each target appliance network 21

4.1 Results of the microwave network on performance metrics 23
4.2 Training and inference times of the microwave network 24
4.3 Results of the fridge network on performance metrics 25
4.4 Training and inference times of the fridge network 26
4.5 Results of the dishwasher network on performance metrics 26
4.6 Training and inference times of the dishwasher network 27
4.7 Results of the washing machine network on performance metrics 27
4.8 Training and inference times of the washing machine network 28
4.9 Comparison of the results obtained by the proposed approach with other NILM research . 28

5.1 Number of parameters in each target appliance network 32

Ethics Statement

This project did not require an ethical review, as determined by my supervisor, Sion Hannuna.

Supporting Technologies

The following supporting technologies were used under the scope of this dissertation:

- The Anaconda software distribution [4] was utilised in order to facilitate the management and deployment of Python packages and it can be accessed here.
- The TensorFlow open-source machine learning library [1] was used to implement the deep neural network proposed in this thesis and the software is available from here.
- The non-intrusive load monitoring workflow suggested by Zhang et al. [64] was used as a starting point for the proposed approach and its source code can be found here.

Notation and Acronyms

LM	:	Load monitoring
NILM	:	Non-intrusive load monitoring
ILM	:	Intrusive load monitoring
REDD	:	Reference energy disaggregation data set
ML	:	Machine learning
DL	:	Deep learning
HMM	:	Hidden Markov model
CNN	:	Convolutional neural network
LSTM	:	Long short-term memory
PM	:	Pattern matching
GSP	:	Graph signal processing
DTW	:	Dynamic time warping
MVM	:	Minimum variance matching
SS	:	Source separation
BSS	:	Blind source separation
SCA	:	Sparse component analysis
NMF	:	Non-negative matrix factorization
ICA	:	Independent component analysis
ANN	:	Artificial neural network
MSE	:	Mean squared error
RNN	:	Recurrent neural network
Tanh	:	Hyperbolic tangent
ReLU	:	Rectified linear unit
1D	:	One-dimensional
2D	:	Two-dimensional
RMS	:	Root mean square
P	:	Real/active power
Q	:	Reactive power
V	:	Voltage
I	:	Current
FFT	:	Fast Fourier transform
DNN	:	Deep neural network
UK	:	United Kingdom
UK-DALE	:	UK domestic appliance-level electricity data set
NILMTK	:	Non-intrusive load monitoring toolkit
dAE	:	Denoising autoencoder
CO	:	Combinatorial optimisation
FHMM	:	Factorial hidden Markov model
seq2point	:	Sequence-to-point
seq2seq	:	Sequence-to-sequence
ED	:	Event detection
EE	:	Energy estimation
TP	:	True positive
TN	:	True negative
FP	:	False positive
FN	:	False negative

RMSE	:	Root mean square error
MAE	:	Mean absolute error
SAE	:	Signal aggregate error
Max pooling	:	Maximum pooling
USA	:	United States of America
Hz	:	Hertz
W	:	Watts
SCANet	:	Scale- and context-aware network
LDwA	:	Load disaggregation with attention
CPU	:	Central processing unit
GPU	:	Graphics processing unit

Chapter 1

Introduction

1.1 Overview

This chapter gives some context about the current energy crisis and how energy load monitoring can be effectively used as a mitigating strategy. The motivation behind the project is described as well as its main aims and objectives.

1.2 Motivation

Electrical energy consumption has been consistently increasing since the industrial revolution [50] mainly due to a rise in the prevalence of automation across industries and a growth in the number of devices which rely on electric energy to function [54]. Before the Covid-19 pandemic, the worldwide electricity demand grew by 4%, which was roughly twice as quickly as the increase in overall energy consumption [24]. Despite the fact that clean energy sources like nuclear power and renewables were mostly used to meet this increase in demand, there was still a sizable increase in the production of electrical energy from coal and gas, which increased CO_2 emissions by 2.5%. This tendency strains the planet's natural resources and exacerbates global warming, hence, several mitigation strategies for the negative repercussions of this escalation in energy consumption have been proposed in recent years.

Residential electricity consumption is estimated to account, globally, for around 26% of the total electric energy consumption and its portion has been growing in recent decades [25]. This upwards trend in residential electricity consumption is in part explained by the improvement in quality of life seen across the globe [3] and the growth of the world population, which can be expected to follow the same tendency in the foreseeable future.

Previous studies indicate that real-time energy consumption feedback through the usage of smart meters can reduce energy consumption by as much as 17% [16]. This reduction is beneficial in many aspects, notably regarding the savings it implies for the final consumers and the decrease in carbon emissions.

Appliance-level insight, often called load monitoring (LM), has been shown to lead to bigger reductions in energy consumption [6] when compared to aggregated household energy consumption information, which confirms the intuition that specific feedback is more effective than generic one. Load monitoring, combined with posterior user behaviour analyses, allows the uncovering of mitigating strategies, such as suggesting the replacement of some appliances by more energy-efficient ones or nudging the users to their own inefficient practices so that they are aware and make efforts to change them [7]. Reductions in electricity consumption are likely to grow significantly as a result of multiple external factors, such as the growth in awareness of the general population regarding the consequences of climate change and the rise of energy prices [33].

Moreover, the findings uncovered by load monitoring can be used by energy suppliers and network operators in order to enhance their operations and infrastructure. By knowing the final consumers' patterns of electrical energy consumption, actions can be taken in order to increase the electric power distribution system's economic efficiency, improve the reliability of the network by better managing the equilibrium between demand and supply, and decrease its environmental implications [61, 39].

The current energy crisis, mainly caused by the aftermath of the Covid-19 pandemic and the Russian invasion of Ukraine [26], makes this topic even more relevant due to the record-high prices and shortages in supply seen throughout the energy industry.

1.3 Key Aims and Objectives

The main aim of this dissertation is to propose a new approach to performing non-intrusive load monitoring (NILM), also known as energy disaggregation, which outperforms previous methods. Furthermore, the proposed approach ought to be practical to implement, both from a financial and deployment standpoint, as well as fundamentally scalable. If accomplished, it could potentially contribute to the proliferation of energy disaggregation models to obtain feedback on electricity usage, both in real-time and otherwise.

Several steps will be taken over the course of this dissertation in order to achieve the stated goal. Previous NILM research will be reviewed so that an overview of the field can be attained, including best practices and frequently used techniques. Different approaches will be implemented and tested with the aim of progressively improving performance. Finally, the proposed workflow will be compared with other research in the area, preferably by using a commonly used data set as a benchmark. Additionally, the suggested approach will also be critically evaluated so that its strengths and weaknesses are discussed and future research streams can be considered.

1.4 Summary

In this chapter, the worldwide rise in electricity usage, especially in a residential setting, was analysed, as well as some of its repercussions. The potentially significant impact of load monitoring in reducing residential energy consumption was also described, alongside positive implications for energy suppliers and network operators. Moreover, succinct overviews were made about what is desired to be accomplished by this research project and the intended course of action. In the next chapter, load monitoring is discussed in-depth, a review of relevant research in the area is carried out and a summary of anticipated challenges is presented.

Chapter 2

Background

2.1 Overview

In this chapter, load monitoring is introduced and described in detail, with a particular focus on the non-intrusive variant and its problem formalisation. Relevant NILM research is reviewed, alongside commonly used performance metrics, and an outline of anticipated challenges is presented.

2.2 Load Monitoring

Load monitoring consists of obtaining information about which appliances are consuming energy and quantifying this consumption. In order to categorise the different types of appliances, four main categories of energy consumption signatures have been proposed by previous studies [13]:

1. Type I: appliances with only ON/OFF states; for instance, kettles and microwaves.
2. Type II: appliances with multiple but finite possible states when active; washing machines and dishwashers, for example.
3. Type III: appliances with an infinite number of active states which are continuously changing their energy consumption; such as power drills and light dimmers.
4. Type IV: appliances that are always ON and consuming energy which can either have only one active state (similarly to type I) or multiple active states (similarly to type II and III); e.g., fridges and smoke alarms.

The difference between patterns of electricity usage across the four types is significant and Figure 2.1 shows examples of appliances' signatures that fall under each one of these categories.

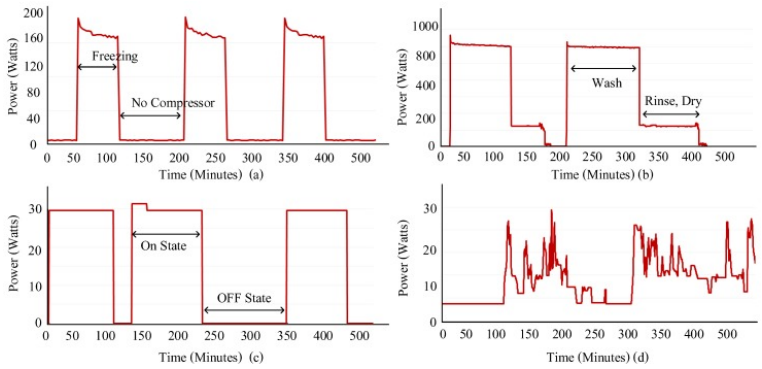


Figure 2.1: Energy consumption signatures of the four different types:
a) Fridge (type IV); b) Washing machine (type II); c) Fan (type I); d) Laptop (type III) [13]

It can be seen that the complexity of the energy consumption pattern of type III appliances is substantially higher than that observed in the remaining power signature types. It is also worth mentioning

that in Figure 2.1 the fridge is always consuming energy, although it displays a pattern of consumption with two clearly distinct stages.

Load monitoring is usually performed through one of two overall approaches: intrusive load monitoring (ILM) and non-intrusive load monitoring. However, it can be achieved by employing many techniques, especially within the NILM variant, as suggested by Figure 2.2.

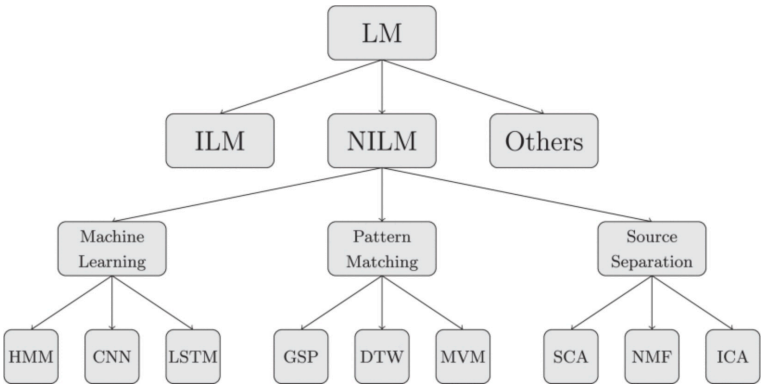


Figure 2.2: Typical load monitoring approaches [54]

Energy disaggregation is usually performed through machine learning (ML), pattern matching (PM) and source separation (SS) methods. ML approaches include hidden Markov models (HMMs), convolutional neural networks (CNNs) and long short-term memory (LSTM) recurrent neural networks, whilst pattern matching techniques often involve graph signal processing (GSP), dynamic time warping (DTW) and minimum variance matching (MVM). Additionally, source separation is typically attained in a NILM context by sparse component analysis (SCA), non-negative matrix factorization (NMF) and independent component analysis (ICA).

Intrusive load monitoring and NILM and the three general approaches that fall under it will be discussed throughout this chapter. Nonetheless, non-intrusive load monitoring is generally preferred when compared to ILM and applied more often in real-life scenarios due to being a more practical, cheaper and scalable alternative [2].

2.2.1 Intrusive Load Monitoring

Intrusive load monitoring consists of placing energy consumption meters directly at the appliance level in order to identify which appliances are ON and how much energy they are consuming.

ILM is not widely applied in order to perform load monitoring because there are several disadvantages inherent to this approach. The higher costs due to the increase in wiring and communications and the retrofitting needed in order to equip these sensors on appliances which are already being used are some of the setbacks that make this approach impracticable, especially at a large scale [54].

2.2.2 Non-Intrusive Load Monitoring

Non-intrusive load monitoring is a technique used to extract the energy consumption of each individual appliance from a single point of aggregated energy consumption measurement [7]. A smart meter is usually used to measure the total energy load, this data is then pre-processed so that it can finally be modelled by a variety of predictive algorithms. It is important to note that the appliances' energy consumption signatures will be superimposed and the algorithms aim to find what these signatures are and separate them [49], as illustrated in Figure 2.3.

The problem of energy disaggregation was first introduced in 1992 by Hart et al. in their *'Nonintrusive appliance load monitoring'* paper [18] and it was originally mathematically defined by (2.1).

$$A_t = \sum_{m=1}^{M} y_t^{(m)} + \sigma_t \tag{2.1}$$

A_t represents the aggregated energy consumption at time step t. $y_t^{(m)}$ is the energy consumption of appliance m at time step t, where $m \in \{1, 2, 3, \ldots, M\}$ and M is the number of appliances that were being individually monitored when the data was collected. σ_t encapsulates the energy consumption of

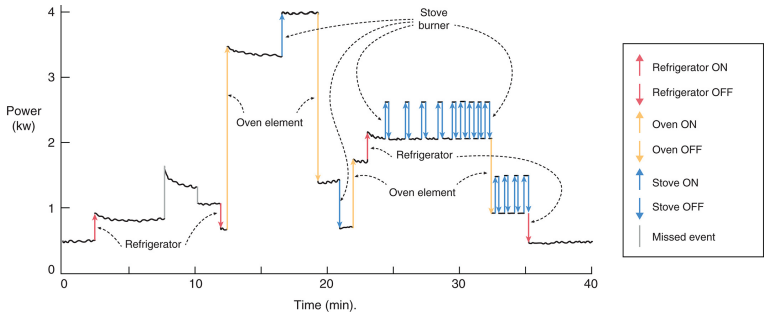


Figure 2.3: Energy disaggregation [46]

appliances which were not being discretely accounted for at the time of data acquisition and the noise inherent to the measurement equipment at time step t. NILM algorithms aim to estimate $y_t^{(m)}$ from A_t, i.e., deduct the energy consumption of an appliance m, at time step t, from the aggregated energy consumption at that time step.

Huber et al. [22] proposed an alternative formalisation of the NILM problem, described by (2.2), which divides the noise term into two distinct components.

$$A_t = \sum_{m=1}^{M} y_t^{(m)} + \sum_{k=1}^{N-M} y_t^{(k)} + \epsilon_t \tag{2.2}$$

$y_t^{(k)}$ is the energy consumption of appliance k at time step t, where $k \in \{1, 2, 3, \ldots, N-M\}$ and N is the total number of appliances, monitored and not monitored, that contribute to the aggregated energy consumption signal. Due to significant differences in their nature, the noise term previously represented as σ_t was divided into two separate elements. The $\sum_{k=1}^{N-M} y_t^{(k)}$ term represents the sum of the energy consumption of the appliances that were not being individually monitored at the time of data collection at time step t. It usually accounts for a big portion of the aggregated energy consumption signal at every time step and is unlikely to follow a normal distribution. ϵ_t is the characteristic noise term, at time step t, of the measurement equipment used to gather data, which most likely follows a z-distribution and does not substantially influence the aggregated energy consumption signal.

The typical workflow of most approaches from the three main NILM research streams, i.e. machine learning, pattern matching and source separation, is shown in Figure 2.4.

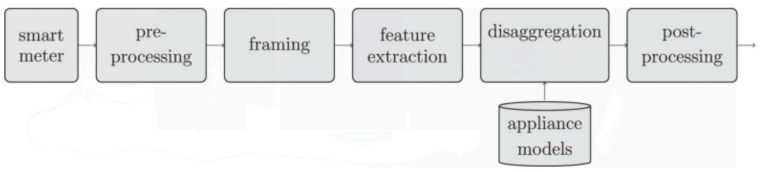


Figure 2.4: Typical workflow of energy disaggregation approaches [54]

Most workflows start by pre-processing the smart-meter aggregated data, which is sampled at a consistent frequency. It is usually done by normalising or standardising it and removing outliers and faulty measures. The next step, framing, involves using what is called a sliding window technique which divides the input data into frames, an approach that can greatly vary when it comes to the length of the windows and the overlap between consecutive frames. Feature extraction consists of detecting patterns in the appliances' energy consumption signatures that uniquely characterise them. It was previously done with the support of experts in the field but with the introduction of deep learning approaches, especially convolutional neural networks, this step can now be done as part of the disaggregation stage, reducing human intervention during the training process. Disaggregation can be performed in order to predict whether the appliances are ON/OFF or the energy they are consuming, making it a classification (event-based NILM) or regression (state-based NILM) problem, respectively. The approaches used at this stage vary accordingly to whether machine learning, pattern matching or source separation is being performed. Post-processing generally consists of state corrections such as mapping negative energy consumption predictions to zero and inverting the normalisation or standardisation performed at the pre-processing stage.

Several approaches were applied to the NILM problem throughout the years and even though some of them showed great results, such as Hidden Markov Models, Deep Neural Networks have been recently getting a lot of attention from researchers in the field due to their generalisation ability, scalability and notable performance results [43, 5].

2.3 Energy Disaggregation through Machine Learning

Hidden Markov models are used to extract the most likely sequence of hidden variables from a sequence of observed data [57]. A simple HMM that models an individual appliance would have as its hidden states the target appliance's operating statuses and as the observed states the appliance's energy load [15].

However, in the context of energy disaggregation, models should be able to extract appliance-level information from the aggregated energy consumption signal and HMMs can be modified in order to achieve this. The observed states can be seen as the aggregated energy load, which combines each appliance's individual consumption, and the hidden variables as a joint state that encapsulates the individual operating statuses of each monitored appliance [65]. This approach leads to state exponentiality, which is a major setback in real-life applications due to the computational costs it implies [40].

Factorial hidden Markov models have been proposed as a workaround for this issue. They have been used extensively in research due to being able to model multiple appliances from the aggregated energy consumption signal efficiently. Instead of aggregating the appliances' operating statuses in an HMM hidden state, each appliance is individually modelled by a Markov chain and the observed variable is statistically obtained from these, as illustrated in Figure 2.5.

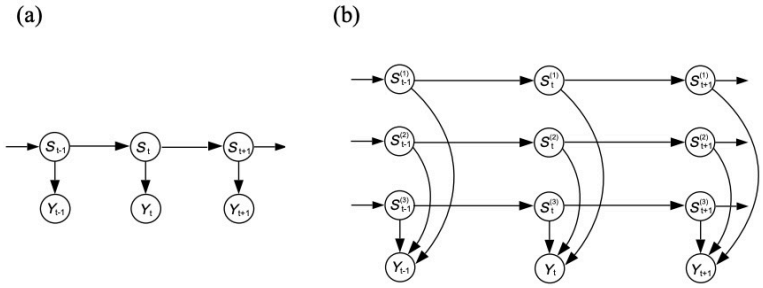


Figure 2.5: Structure of different hidden Markov models:
a) simple HMM; b) factorial HMM [17]

The observed variable is represented by Y, whilst the hidden variables are denoted by S. Y_t stands for the ground-truth total energy consumption at time step t. In the simple HMM, S_t is the super-state composed of the individual states at time step t of the monitored appliances. In the factorial HMM, $S_t^{(n)}$ stands for the state of appliance n at time step t. In this example, $n \in \{1, 2, 3\}$, and thus only three appliances were being monitored, which is also suggested by the number of Markov chains.

It is important to note that FHMMs rely on approximate inference and lose information about the relationship between appliances' energy consumption due to their loads being modelled independently by each Markov chain [40], which does not accurately represent what happens in a real-life scenario.

Even though hidden Markov models seem to fit the NILM problem well and are a concept easier to understand and explain than deep learning techniques, the latter have become prevalent in recent years. This is partly due to overcoming some of the setbacks inherent to HMMs, such as the need for expert knowledge in order to define prior distributions over the models' parameters [15].

Artificial neural networks are inspired by how the brain functions and how neurons interact with each other [23]. Deep learning models consist of such structures with one or more hidden layers and are able to model extremely complicated problems. This is due to their hierarchical structure, where each layer builds on top of what was extracted at the previous one, starting from the raw input and ending up with a more abstract and pertinent representation for the task at hand [30, 37]. In the context of energy disaggregation, ANNs are usually trained to predict the energy consumption of a single target appliance instead of all the monitored devices, which helps to reduce the complexity of the models' architectures.

In a fully connected feedforward ANN, non-input layers' units receive as input the output of every node in the previous layer. They perform a weighted sum over the inputs, add a bias and feed the result to an activation function, which is usually non-linear to be able to model complex problems. The output of the j^{th} node in the l^{th} layer, y_j^l, in a fully connected feedforward ANN can be formalised by (2.3) [58].

$$y_j^l = \sigma^l(\sum_{i=1}^{N} w_{i,j}^{l-1,l} x_i^{l-1} + b_j^l) \tag{2.3}$$

$w_{i,j}^{l-1,l}$ represents the connection weight between the i^{th} node of the $l-1^{th}$ layer and the j^{th} unit of the l^{th} layer. $i \in \{1, 2, 3, \ldots, N\}$, where N is the number of nodes in the $l-1^{th}$ layer. The absolute values of these weights can be seen as the relevance the network assigns to each one of these connections [58]. b_j^l is the bias term of the j^{th} node of the l^{th} layer, which is added to the result of the weighted sum over the inputs of that unit. σ^l symbolises the activation function used in the l^{th} layer of the network, which is usually the same for all the neurons within a single layer.

Equation (2.3) can be rewritten as (2.4) [23], by taking the dot product between the weight and input vectors and considering a node's bias an extra unit in the previous layer with a constant output of one.

$$y_j^l = \sigma^l(\sum_{i=0}^{N} w_{i,j}^{l-1,l} x_i^{l-1}) = \sigma^l(w_j^{l-1,l} \cdot x^{l-1}) \tag{2.4}$$

$w_j^{l-1,l}$ is the vector containing the connection weights between every unit in the $l-1^{th}$ layer, including the bias node, and the j^{th} neuron of the l^{th} layer. x^{l-1} is the input vector, which holds the outputs of all the nodes in the $l-1^{th}$ layer, once more including the bias unit. It is worth mentioning that, in contrast to what happens in (2.3), the weighted sum over the inputs of the j^{th} unit of the l^{th} layer starts at index 0, which is the position assigned to the bias node. These computations, which are performed within every fully connected feedforward ANN neuron, are illustrated in Figure 2.6.

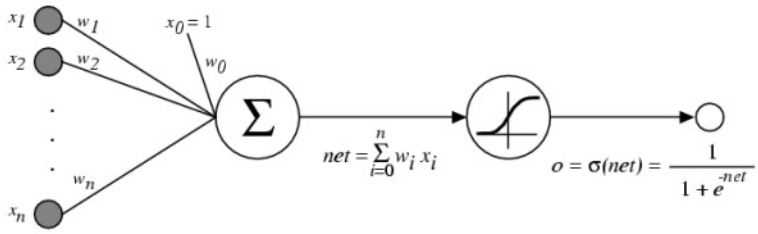


Figure 2.6: Computation steps in a perceptron with the sigmoid activation function [42]

x_i is the i^{th} input to the node being considered, where $i \in \{0, 1, 2, \ldots, n\}$ and n represents the number of units in the previous layer, excluding the bias. The bias node is denoted by x_0 and has a constant value of one. w_i is the i^{th} weight, which connects input i to the analysed neuron. net is the weighted sum over the inputs, including the bias unit, and it is fed to the sigmoid activation function, usually represented by σ, which returns the output of the node being studied.

ANNs learn how to model data by updating the connection weights between nodes, including the biases [29]. In a supervised learning context, the network updates the weights so that a loss function that measures how closely the model's predictions relate to the ground truth values is minimised [37]. When the network is performing regression, the mean squared error (MSE) which is given by (2.5) [44], is often used as the loss function.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(target_i - predicted_i)^2 \tag{2.5}$$

$target_i$ and $predicted_i$ are the output ground truth value and the model's prediction, respectively, for input i, where $i \in \{1, 2, 3, \ldots, N\}$ and N is the number of data points being considered in order to calculate the MSE.

This learning procedure is usually done through what is called the mini-batch gradient descent algorithm [23]. For each iteration of weight updates, the network uses a small sample of data points from the training set, called a mini-batch, as input and computes their predictions and corresponding errors. Through backpropagation, the average over these errors is propagated backwards through the network (from the output layer to the input) and used to calculate the partial gradients with respect to each connection. Finally, the computations performed in order to update the weights can be described by (2.6) [58, 38].

$$W_{i,j}^{k+1} = W_{i,j}^{k} - \eta \frac{\partial L(W^k)}{\partial W_{i,j}^{k}} \tag{2.6}$$

$W_{i,j}^{k}$ is the connection weight between node i and j at iteration k of weight updates. Each iteration is characterised by a different mini-batch and thus, the normalised loss function associated with it, $L(W^k)$, is most likely distinct. $-\frac{\partial L(W^k)}{\partial W_{i,j}^{k}}$ is the negated partial gradient, i.e. the direction of greatest decrease, of the normalised loss function with respect to $W_{i,j}^{k}$. η is the learning rate and defines how fast the weights should be updated.

The network stops training when a stopping criterion is met. This can be due to many reasons, namely due to reaching a maximum number of epochs, i.e. iterations over the whole training data set, or due to not decreasing the loss function, on the training or validation set, by a specified threshold for multiple consecutive epochs [9].

There are multiple techniques to optimise the training process of a network. Optimisers that make use of momentum, such as adaptive moment estimation (Adam), reduce the convergence time and smooth the trajectory of the loss function as well as the oscillation in the model's weights. Regularisation methods, like dropout, help prevent overfitting the training data and thus improve the generalisation ability of the networks. Moreover, standardisation procedures, such as min-max and z-score scaling, also make the convergence of the model faster and often improve its performance, especially in cases where the features in the data have vastly different scales. Cross-validation, although time-consuming, particularly in networks trained with vast amounts of data, increases the reliability of the network due to different portions of the training data being used for training and validation.

A recurrent neural network, which is a type of ANN, has an architecture that contains cycles between nodes, often called feedback loops, which are associated with a delay in time [45]. It enables the model to remember previous information and consequently to identify relevant relationships between data points, despite their temporal gap [45]. Backpropagation through time is the most commonly used algorithm to train recurrent neural networks [56] and the name arises from the intuition that backpropagation is being carried across the time domain.

They are particularly well-suited for modelling sequential data, and thus, since NILM is performed with time series, they have been extensively used in the field. However, in order to avoid the vanishing/-exploding gradient problem inherent to basic RNNs, which limits their memory capacity, a new variant called long short-term memory was proposed by Hochreiter et al. [21]. The structure of an LSTM unit and its components is illustrated in Figure 2.7.

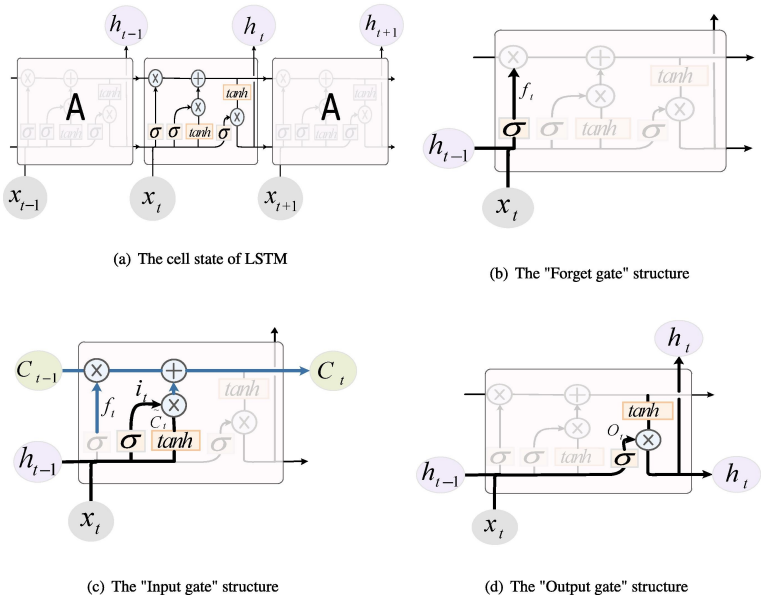


Figure 2.7: Structure of an LSTM block and its different gates [60]

C, h and x represent the long-term memory (or cell state), short-term memory (or output) and input, respectively. Their subscript denotes the time step being considered, with t being the current time step. $\oplus$ and $\otimes$ represent element-wise summation and element-wise multiplication, respectively. In this context, the activation functions, when given a vector as input, are also applied element-wise.

Each repeating neural network module consists of four layers which can be categorised into three distinct parts: the forget, input and output gates. The forget gate does a weighted sum over the current input vector, x_t, and the previous time step short-term memory vector, h_{t-1}, adds a bias and feeds the result to the sigmoid activation function, σ, which outputs a vector representing the amount of long-term memory that the unit wants to preserve. The forget gate's sigmoid layer, given its bias, b_f, and set of weights, W_f, which contains the weights for the input and the recurrent output, can be formally described by (2.7) [60].

$$f_t = \sigma((W_f \otimes [x_t, h_{t-1}]) \oplus b_f) \tag{2.7}$$

The input gate consists of two layers of neural networks, the sigmoid and tanh layers, which together determine how the LSTM cell will update the long-term memory. In the tanh layer, the potential long-term memory is calculated by going through the same steps as the forget gate's sigmoid layer but using the hyperbolic tangent activation function instead, which is often represented by tanh and is given by (2.8). The tanh layer of the input gate, given its set of weights, W_c, and bias, b_c, can be mathematically represented by (2.9) [60].

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.8}$$

$$\tilde{C}_t = \tanh((W_c \otimes [x_t, h_{t-1}]) \oplus b_c) \tag{2.9}$$

Similarly, in the input gate's sigmoid layer, the amount of potential long-term memory that the cell wants to remember is computed. The sigmoid layer of the input gate, given its bias, b_i, and set of weights, W_i, can be formalised by (2.10) [60].

$$i_t = \sigma((W_i \otimes [x_t, h_{t-1}]) \oplus b_i) \tag{2.10}$$

The current cell state is given by (2.11) [60], which multiplies i_t by $\tilde{C}_t$ and adds the long-term memory that the unit decided to remember, i.e., the result of the multiplication between the output of the forget gate and the previous time step long-term memory.

$$C_t = (f_t \otimes C_{t-1}) \oplus (i_t \otimes \tilde{C}_t) \tag{2.11}$$

The amount of potential short-term memory that the LSTM wants to remember, o_t, is obtained through the same process as in the other two gates. It is computed through the sigmoid layer of the output gate, which given its set of weights, W_o, and bias, b_o, can be formalised by (2.12) [60]. The output gate decides how to update the short-term memory by passing the current cell state to a tanh activation function, that outputs the potential short-term memory, and multiplying the result by o_t. Therefore, the updated short-term memory, h_t, is given by (2.13) [60].

$$o_t = \sigma((W_o \otimes [x_t, h_{t-1}]) \oplus b_o) \tag{2.12}$$

$$h_t = o_t \otimes \tanh(C_t) \tag{2.13}$$

In the context of NILM, it is often the case that deep LSTM architectures, which are able to model more complex problems, are used in order to perform energy disaggregation. These models stack LSTM structures on top of each other, with the condition that within-structure parameters are the same and between-structures parameters are different [60]. In order to avoid overfitting, it is also common to make use of dropout layers between LSTM structures, which usually results in better generalisation at the cost of longer training times [12].

Another common modification to the traditional LSTM recurrent neural networks is the use of bidirectional layers, which enables the network to interpret the input signal forwards and backwards before combining both outputs [32]. It increases the models' capacity to make use of the target appliances' temporal properties, which are usually learnt through features extracted from the input data with convolutional neural networks [54]. This variant of LSTMs is not suitable for real-time energy disaggregation

in the original sense of the term, but this problem can be addressed if sustained offline disaggregation executions can produce the desired outcomes [32].

Convolutional neural networks have been extensively used in the field of image recognition due to their innate capacity of learning how to identify useful patterns, usually called features, in the input space in order to classify images. They can be used to analyse 1D data [31], such as time series, and have been successfully applied to NILM in order to extract distinguished patterns of the target appliance [54].

CNNs are characterised by having convolutional layers, which consist of a number of fixed-size filters, also known as kernels, that convolve their input signal [14] and then feed the result to an activation function, generally the Rectified Linear Unit (ReLU) function [44]. The ReLU activation function and the computation steps carried out in a convolutional layer can be mathematically formalised by (2.14) and (2.15), respectively [44, 62].

$$f(x) = max(x, 0) = \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{otherwise} \end{cases} \tag{2.14}$$

$$y_j^l = f[\sum_{i=1}^{N}(y_i^{l-1} * K_{i,j}^{l-1,l}) + b_j^l] \tag{2.15}$$

y_j^l is the j^{th} feature map of the l^{th} convolutional layer and $K_{i,j}^{l-1,l}$ is the filter that maps the i^{th} feature map of the $l-1^{th}$ convolutional layer to the j^{th} feature map of the l^{th} convolutional layer. $i \in \{1, 2, 3, \ldots, N\}$, where N is the total number of inputs to the j^{th} kernel of the l^{th} convolutional layer. Additionally, b_j^l is the bias of the j^{th} kernel of the l^{th} convolutional layer. In this context, the ReLu activation function, f, receives a vector as input and thus, it is applied element-wise.

These filters are updated throughout the training process of the network through backpropagation [11] until an optimal combination is learnt, i.e., the best possible predictive features are extracted. The filters typically identify low-level features in the initial convolutional layers, and the succeeding layers use these to extract patterns that are more complex and abstract [23].

In the context of NILM, the features that were previously obtained with the help of experts in the field are now being replaced with the ones extracted by filters in the convolutional layers. These features are usually categorised as steady-state or transient [34]. Steady-state features identify patterns of the target appliance's energy consumption derived from differences between stable working statuses [66]. They can be generally grouped as follows [34]:

1. Power change: features that use information from the real (P) and reactive power consumption (Q) of the appliances [52].

2. Low-frequency V-I Features: features that draw from the current and voltage information of the appliances, such as the root-mean-squared (RMS) and peak values [52, 51].

3. V-I Trajectory: features that rely on information from the shape of the V-I curve, e.g. the enclosed area, the number of self-intersections and the curvature direction [19].

4. Harmonics: features that are extracted from the harmonics of the current or power waveforms, usually through fast Fourier transforms (FFT) [52].

This is not an exhaustive list of the steady-state features that have been previously used in the NILM literature to identify appliances, but a comprehensive breakdown of these can be found in [52]. These features have been extensively studied mainly due to not requiring high-frequency data and thus reducing the associated costs with hardware [66].

Transient features are based on transitions between stable working statuses [66], hence relying on high-frequency sampling rates, with around 1200 cycles per second being considered the minimum in order to extract them effectively [63]. The size, shape and duration of these waveforms are deeply connected to the appliance's physical task [36, 34] and thus constitute a rich source of information for energy disaggregation models.

Due to the higher sampling rate requirement, these approaches are usually more expensive than the ones based on steady-state features [66]. However, these high-frequency transient patterns can be extremely useful to distinguish between appliances that have overlapping steady-state features, which frequently happens for appliances with low power consumption due to having similar profiles of energy consumption [52, 66, 34].

CNNs, in addition to convolution layers, are usually composed of other types of artificial neural network layers, such as pooling and dense layers [14]. Pooling layers allow the networks to downsample feature maps, often by taking the maximum or average over a region, which makes the training process of the network faster by reducing the number of parameters used in subsequent layers [62]. They also enhance the invariability of the network to small translations of the input due to the output of these layers remaining roughly the same [14]. Fully connected layers are used at the end of the architecture in order to perform either classification or regression, depending on the number of output nodes. The first fully connected layer receives as input a flattened version of the output of the last convolution/pooling layer, i.e., an array composed of the feature maps computed in the previous layer, and then it either feeds the result to subsequent fully connected layers or outputs the final prediction made by the network [44, 62]. The described architecture, using 2D inputs and 1D kernels, can be seen in Figure 2.8.

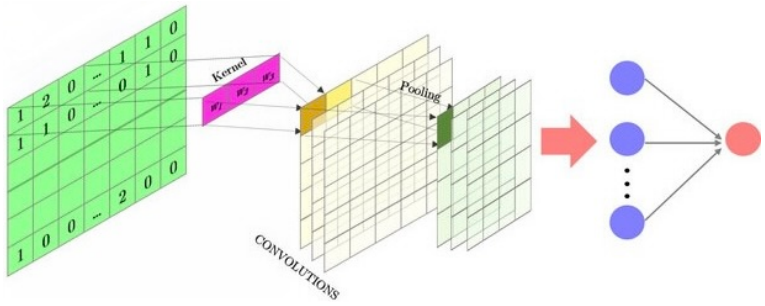


Figure 2.8: Structure of a convolutional neural network with 2D inputs and 1D kernels [48]

It can be seen that CNNs usually have a convolutional section, from the input until the last convolutional/pooling layer, and then a dense block, from the first dense layer until the output [62]. They are effectively performing different tasks, with the convolutional section extracting features and the dense block making predictions based on these [62, 30].

It is also common for CNNs to have dropout layers [44]. Even though they can be applied to any non-output layer, they are usually deployed between fully connected layers, as suggested in [20]. These layers randomly hide artificial neurons, and their connections, so that individual units do not rely on others [44, 20]. By efficiently mimicking the training process of several networks with various architectures, neurons become individually more robust [20]. These layers help avoid overfitting the training data and in turn, the model is usually able to generalise better [44].

2.3.1 Literature review

Makonin et al. [40] proposed a state-of-the-art super-state hidden Markov model which makes use of a modification of the Viterbi algorithm. This model preserves the load dependency between appliances and due to the sparse variant of the Viterbi algorithm, it is able to efficiently perform exact inference, making it suitable for real-time disaggregation tasks. It was shown to perform well on complex multi-state appliances, however, due to its super-state architecture and corresponding state exponentiality problem, it is limited when it comes to the number of appliances modelled and their number of operating states.

Kelly and Knottenbelt [32] were the first to apply deep neural networks to NILM and their approach is extensively described in this section. The UK-DALE data set was used in their study and the appliances chosen to perform the analysis on were the fridge, washing machine, dishwasher, kettle and microwave because they are high energy consumption appliances and were present in multiple houses within the data set, facilitating the training and testing of the models. By using these five appliances, the algorithms were also put to the test regarding different energy consumption signature types.

In order to train deep neural networks effectively, the training set has to be large and to achieve that artificial data was created by randomly combining real power consumption signatures from different appliances. It was found that the models generalised better to unseen households when trained with a joint set of both real and synthetic data and hence the authors decided to use a 50:50 ratio between these two types of data in the training data set. The testing set was exclusively populated with real data to better model real-world scenarios and it contained the last week's worth of data from every house, with the rest being used for training the algorithms.

Five networks were trained in total, one for each appliance, and due to computational limitations and in order to avoid the vanishing gradient problem [64], the input was 'split' into multiple windows which

have different sizes taking into consideration the appliance being modelled. All the trained networks had windows of aggregated energy consumption as input and the power drawn by the target appliance during that same sequence as output, an approach to perform NILM that is called sequence-to-sequence (seq2seq). It was discovered that longer windows usually deteriorate the performance of the networks. However, the window should be large enough to capture most of the appliance's cycles in their entirety. It is important to note that these time frames can overlap depending on the amount of shifting applied from window to window and thus, the overall output of the network will take into account the multiple predictions made by the model at each time step.

To train the models, the real data was first pre-processed by using NILMTK's [7] methods and the power activations of every appliance were obtained. Each training example was then determined to contain or not contain the appliance power signature with a 50% probability. If it was decided that it would not contain the appliance signature, a random window from the real aggregated energy consumption data not containing any activation of the appliance would be selected as the input and the target would also not contain any activation. On the other hand, if it was decided that the training example would contain the appliance signature, a randomly selected appliance activation would be randomly placed in the window shown to the algorithm as the target and the corresponding time frame of real total energy consumption data would be used as the input. The synthetic training data followed a similar procedure but with assigned probabilities as to whether or not the appliances' signatures should be comprised in a particular window. The target appliance had a 50% chance of being included and the other appliances, called distractors, had a probability of 25% each.

Three different deep neural network architectures were analysed in their paper: a long short-term memory (LSTM) recurrent neural network, a denoising autoencoder (dAE) and a regression algorithm. All of these approaches used the Mean Squared Error as the loss function that the models aim to minimise. The LSTM used backpropagation through time as the learning algorithm and was made up of six layers: an input layer which varied in length accordingly to the target appliance and its duration, a 1D convolutional layer which extracted features from the input over the time frame provided, two bidirectional LSTM layers and two fully connected layers. It might seem redundant to simultaneously use LSTMs and convolutional layers since it is expected that LSTMs can remember the entire context of the input, however, the model outperformed its counterpart without the convolutional layer.

The denoising autoencoder was an approach to energy disaggregation which aimed to learn how to clean the input, that is the 'noisy' signal that consisted of the total energy consumption data, so that the target appliance load could be extracted, the 'clean' signal. The dAE had six layers in total: an input layer, a 1D convolutional layer, three fully connected layers in which the middle one was the 'code layer' responsible for encoding the input into a more compact format and finally, another 1D convolutional layer which 'deconvolved' the results.

The regression algorithm focused on learning how to model the start and end time as well as the total load consumed by a target appliance activation in the window of aggregated energy load used as input to the network. This can be seen as modelling the activation through a rectangle, which has the left edge at the start time, the right edge at the end time and the height corresponding to the target appliance's average power consumption during that activation. The deep neural network used for this purpose had eight layers: an input layer, two 1D convolutional layers and five fully connected layers.

On unseen houses, the denoising autoencoder and the regression model showed better results than previous approaches to NILM, namely combinatorial optimisation (CO) and factorial hidden Markov model (FHMM) algorithms, on almost every performance metric used. On the other hand, the LSTM network consistently demonstrated worse results than the CO and FHMM on multi-state appliances, whilst performing better on two-state ones.

Zhang et al. [64] built upon the work carried out by Kelly and Knottenbelt and proposed a new approach to carry out NILM called sequence-to-point (seq2point), which is comprehensively described in this section. This alternative to the previously described seq2seq algorithm aims to predict the middle point of a window instead of having as the target output a continuous interval of power consumption by the selected appliance. It managed to avoid some of the problems related to the seq2seq technique, such as the smoothing of the results due to having to combine multiple predictions for each time step and not taking into consideration that time steps further to the sides of a specific window are likely to be predicted with less accuracy than the midpoint as a consequence of not having as much relevant past or future data to learn from.

The architecture of the neural networks used to perform both seq2seq and seq2point were extremely similar, only diverging on the last layer in order to get either the full window prediction (seq2seq) or a single time step prediction (seq2point), as shown in Figure 2.9.

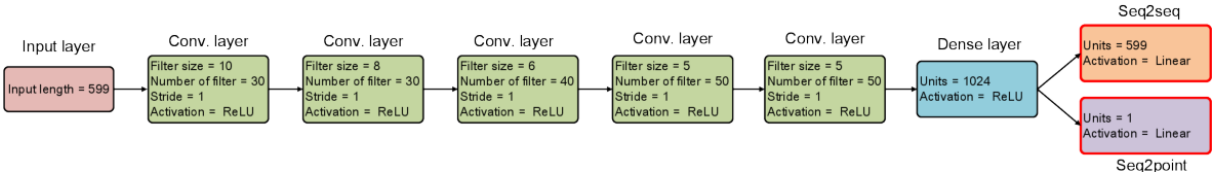


Figure 2.9: Architecture of the seq2seq and seq2point models proposed by Zhang et al. [64]

Both architectures contain the two typical CNN sections that were previously covered in this chapter. The convolutional block goes up to the 5^{th} convolutional layer and extracts features from the input data, whilst the dense section contains the two dense layers at the end and computes the actual predictions from the features extracted.

The models were trained and tested on the REDD and UK-DALE data sets, two of the most used data sets in the context of NILM research. As in Kelly et al. [32], the appliances studied were the fridge, microwave, washing machine, dishwasher and kettle on the UK-DALE data set and due to the REDD data set not having available data about the kettle, that appliance was not analysed on it. House 2 and house 1 were used as the testing data for the UK-DALE data set and REDD data set, respectively, while the remaining houses in each data set were used to train the models.

The networks were trained using the same method explained in [32]. The seq2seq network used selected windows, which were extracted at every time step, of the aggregated energy consumption data as input and the corresponding time frames of the target appliance power consumption as the target. On the other hand, the seq2point algorithm only used the midpoint of the target appliance energy consumption window as the target, instead of the whole interval.

Both models presented obtained substantially better results when compared to the denoising autoencoder proposed by Kelly et al. which was the best-performing seq2seq network out of the three suggested approaches. The mean absolute error and the normalised signal aggregate error, the two metrics used to compare models, were reduced by over 80% by the two models presented by Zhang et al. in their paper and the seq2point approach had the overall best performance in both data sets. It was also shown that both deep convolutional neural networks suggested by Zhang et al. were able to learn features from the data automatically, such as the target appliance energy consumption signature, instead of this step being done manually and subsequently fed to the algorithms like in previous studies.

Piccialli et al. [47] proposed a state-of-the-art algorithm to perform energy disaggregation, which is summarised in this section. The load disaggregation with attention (LDwA) model consists of a deep neural network with two distinct but cooperating components: the regression and classification subnetworks. The architecture of the seq2seq model used in their paper is illustrated in Figure 2.10.

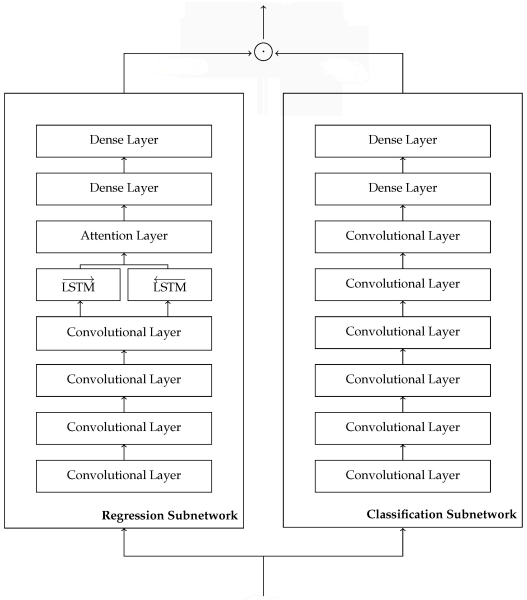


Figure 2.10: Architecture of the LDwA model proposed by Piccialli et al. [47]

The regression subnetwork is composed of four 1D convolutional layers, a bidirectional LSTM, an attention mechanism and two dense blocks. The attention layer employed has the purpose of finding the time steps in the aggregated signal where the most important changes in the target appliance's state happen. It assigns a weight to each observation of the input, with its magnitude representing the importance of the change at that time step. On the other hand, the classification subnetwork contains six 1D convolutional layers followed by two dense layers.

The output of the classification subnetwork at each time step represents a probability classification since it lies in the $[0, 1]$ range, with 0 indicating maximum confidence that the target appliance is OFF and 1 denoting maximum confidence that the appliance is ON. In this context, an output of 0.5 suggests that the model could not decide between the ON and OFF statuses at that time step and thus both states have a probability of 50%. The regression subnetwork output at each time step represents the model's initial energy consumption estimate. Finally, these two outputs are element-wise multiplied, an operation that is represented by $\odot$ in Figure 2.10, so that the final prediction of electricity consumption at each time step can be obtained.

It was shown that it achieved better results in all appliances analysed when compared to previous approaches, such as the seq2point network presented by Zhang et al. [64] and other state-of-the-art energy disaggregation models. The training times of the networks were also studied and even though there are more efficient models, namely the ones that solely depend on convolutional layers instead of LSTM layers, the gains in performance demonstrated by this approach clearly outweigh the slightly longer training overhead when compared to these competing networks.

2.4 Energy Disaggregation through Other Common Approaches

Pattern matching consists of identifying predefined features in the data [41]. Dynamic time warping (DTW) is the most used technique within this category and involves finding the most compatible match between two temporal sequences. Wang et al. [59] proposed a DTW approach that performs load monitoring by disaggregating the mains signal one appliance at a time, starting with the highest energy-consuming one. This is achieved by carrying out sub-sequence searching with DTW, which extracts the load drawn by a single appliance throughout a specified window based on the usual energy consumption pattern shown by the target appliance.

Source separation, referred to as blind source separation (BSS) when the nature of the sources and the mixing process are unknown, consists of extracting signals from an aggregated input [53]. Independent component analysis (ICA) is the most prevalent method within this NILM research stream and aims to find the most independent set of sources that could be combined in order to recreate the original mixed input. Semwal et al. [55] carried ICA to extract individual appliances' loads from households' aggregated energy consumption signals and it was shown that it performed significantly better in appliances with only ON/OFF operating statuses.

Pattern matching and source separation seem plausible approaches to NILM due to the idea that the aggregated load drawn by a household is roughly the sum of the energy consumption patterns of each individual appliance monitored. However, with the noise inherent to power consumption measurement equipment, the significant differences in power signatures between appliances of the same type, the dependence between the usage of multiple appliances, the likelihood of not monitoring all the energy-consuming devices in a household, amongst other factors, their suitability when compared to other approaches quickly becomes arguable.

2.5 Performance Metrics in NILM

The most commonly used benchmarks can be divided into two categories [46]: event detection-based (ED) and energy estimation-based (EE). ED evaluation metrics measure how well the models perform energy disaggregation based on how precise their predictions identify changes in the operating states of the target appliance, namely ON/OFF transitions [34]. These states can be extracted by thresholding the energy consumption with a value that is thought to represent the minimum load drawn by the target appliance when it is ON [60]. Frequently used ED performance metrics, namely *accuracy*, *precision*, *recall* and $F1$-*score*, are obtained from the model's confusion matrix and are given by (2.16), (2.17), (2.18) and (2.19), respectively [34, 46, 60].

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.16}$$

$$precision = \frac{TP}{TP + FP} \tag{2.17}$$

$$recall = \frac{TP}{TP + FN} \tag{2.18}$$

$$F1\text{-}score = 2 \times \frac{precision \times recall}{precision + recall} \tag{2.19}$$

TP, TN, FP and FN represent the true positive, true negative, false positive and false negative terms, respectively. TP and TN stand for the number of correctly predicted ON and OFF events, correspondingly. FP and FN denote the number of times the model incorrectly estimated ON and OFF events, respectively.

The *accuracy* describes the proportion of correctly predicted events to the total number of observations. It indicates how precise the model is at classifying the operating status of the target appliance as ON or OFF. The *precision* is the proportion of ON events that were correctly estimated against the total number of times that they were predicted. It provides an insight into how confident we can be regarding high energy consumption values predicted by a model, with the threshold for being considered high values changing accordingly to the target appliance. The *recall* represents the ratio between the amount of correctly estimated ON events and the ground-truth number of times that they were observed. It gives an indication of how good a model is at identifying power activations of the target appliance. Finally, the $F1$-*score* is the harmonic mean of *precision* and *recall*, giving a better overview of the network's performance when it comes to its ability to correctly classify the status of the target appliance as ON or OFF. It is important to note that since the four evaluation metrics discussed above stand for ratios, they lie in the $[0, 1]$ range, with 1 indicating perfect performance.

On the other hand, energy estimation-based performance metrics rely on statistical analysis methods [46]. A variety of these techniques are prone to not accurately depicting the real performance of models, especially in scenarios where the target appliance does not consume energy for long periods of time since the prediction can always be no consumption and still yield excellent results [34]. Some of the most commonly used EE evaluation metrics in NILM literature are given by (2.20), (2.21) and (2.22) [34, 46, 60].

$$MAE = \frac{1}{T} \sum_{t=1}^{T} |target_t - predicted_t| \tag{2.20}$$

$$RMSE = \sqrt{\frac{\sum_{t=1}^{T} (target_t - predicted_t)^2}{T}} \tag{2.21}$$

$$SAE = \frac{1}{N} \sum_{n=0}^{N-1} \frac{1}{K} \left| \sum_{t=n\times K+1}^{(n+1)\times K} target_t - \sum_{t=n\times K+1}^{(n+1)\times K} predicted_t \right| \tag{2.22}$$

$target_t$ stands for the ground-truth energy consumption of the target appliance at time step t and $predicted_t$ is the model's prediction of the same variable at that time step. $t \in \{1, 2, 3, \ldots, T\}$, where T is the total number of time steps being considered. Additionally, $n \in \{0, 1, 2, \ldots, N-1\}$, where N represents the number of consecutive disjoint windows with K time steps within the sequence being evaluated.

The mean absolute error, denoted by MAE, indicates the average absolute error between the network's energy consumption predictions and the ground truth values. It can be seen as the expected absolute error for any prediction made by the model. The root-mean-square error is represented by $RMSE$ and is a reliable estimation of the standard deviation of the model's error distribution. The SAE stands for the signal aggregate error and it is the average, over the sequence being evaluated, mean absolute error within windows of K time steps. In contrast to what was mentioned about the ED evaluation metrics, 0 indicates perfect performance in energy estimation-based metrics since they capture the errors yielded by the assessed model.

2.6 Main Anticipated Challenges

A challenge intrinsic to NILM machine learning algorithms is their generally poor performance in disaggregating low-power appliances due to noise hindering the impact that these appliances have on the mains load [32]. Some algorithms also show limitations in modelling appliances that have multiple operating states [32], that is type II and III appliances, due to their more complex energy usage patterns.

The lack of state-of-the-art and widely accepted by the academic community metrics of performance is also one of the current biggest challenges regarding NILM research [34, 46, 7]. It makes the comparison between contending approaches harder and thus hinders the possibility of finding a widely recognised set of algorithms that are known to work best for this type of application and are seen as a reference for future work.

There is also a shortage in the availability of comprehensive, but under the same format, publicly-assessed data sets that can be used to train and test these disaggregation models [46, 7]. There are no standard approaches to collecting energy consumption data from households and hence these data sets usually have different durations, sampling rates and features measured. These factors contribute to the segmentation of the research in the field because most studies end up focusing on only one data set and do not extend their investigation to others.

2.7 Summary

In this chapter, load monitoring was introduced and the energy disaggregation problem was formalised. Prevalent NILM methods, namely HMMs, ANNs, DTW and ICA, were considered and relevant approaches within those research streams were reviewed. Commonly used performance metrics and some of their inherent limitations were also discussed. Furthermore, an overview of the main challenges anticipated to be encountered within the scope of this dissertation was presented. In the next chapter, a detailed explanation of the proposed NILM process is given, covering topics such as the model architecture and the pre- and post-processing steps.

Chapter 3

Project Execution

3.1 Overview

In this chapter, the adopted NILM workflow is thoroughly explained, including sections on the ANN architecture, the data set utilised, the pre- and post-processing stages, the data splits and the sliding windows used, as well as how batch processing was employed.

3.2 Proposed ANN Architecture

The source code of the deep learning workflow proposed in this dissertation is available here and it had as a starting point and main inspiration Zhang et al.'s [64] implementation of seq2point energy disaggregation. The proposed model is illustrated in Figure 3.1 and it is mainly composed of three one-dimensional convolutional layers, a bidirectional long short-term memory block and two dense layers.

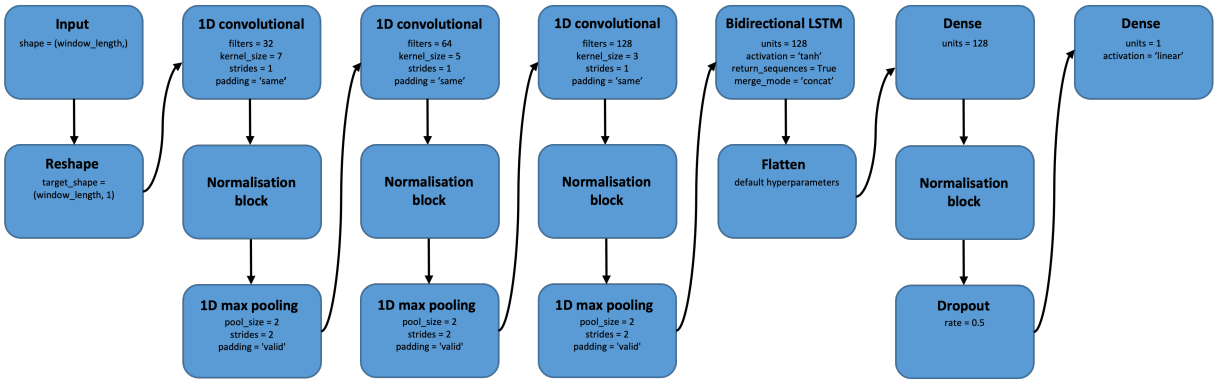


Figure 3.1: Architecture of the proposed energy disaggregation model

The 1D convolutional layers act as feature extractors, which are then passed on to the bidirectional LSTM which is able to better understand the context of the input due to selectively remembering information. The output of the bidirectional LSTM is fed to a dense layer in order to increase the expressibility of the model. Finally, this layer is connected to another dense layer with a single output node which will output the model's prediction at the middle point of the input sequence.

The reshape layer behaves as a connector so that the input can be passed on to the first 1D convolutional layer. It only adds one dimension to the data, without changing its intrinsic values. The model's flatten layer also serves a similar purpose, in that it allows the output of the bidirectional LSTM layer to be fed to the first dense one, by collapsing its dimensions.

Several regularisation techniques were used in order to avoid some of the issues that usually arise from the use of ANNs. 1D max pooling layers were introduced after each convolutional-normalisation block to downsample the input and thus speed up the training process. The network also gains to some extent input shift invariance which is particularly useful when it comes to temporal data. These layers had a pool size and stride of 2, which means that for every disjoint group of two consecutive time steps, only the maximum value was kept and the other was disposed of. If there are any remaining time steps

towards the end of the input which do not make up for a full pool window accordingly to the stride and pool size defined, these are dropped by the network because the padding was set to 'valid'. A dropout layer with a rate of 0.5 was added between the two dense layers to control overfitting and improve the generalisation ability of the model. In each training step, it hides units from the first dense layer with a probability of 50% so that the following dense layer does not overly rely on certain nodes to make predictions. This makes the neurons individually more robust, enhancing their capacity to learn abstract patterns from the data and hindering overfitting on the training set.

The batch normalisation layers receive as input the output from the layer that precedes them and transforms it, by using the mean and standard deviation over the current mini-batch, so that the result has a mean close to 0 and a standard deviation around 1. The normalisation block has the structure shown in Figure 3.2.

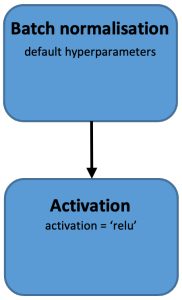


Figure 3.2: Structure of the proposed normalisation block

The mean and standard deviation values computed from each mini-batch are parameters of the network, although non-trainable since they are determined directly from the data and not through gradient descent. With the use of batch normalisation, the training of the network becomes faster due to feeding standardised data to the activation layers which in turn output stable distributions instead of sparse ones [28]. The output of the batch normalisation layers is then fed to the subsequent activation layers, which apply the rectified linear unit (ReLU) activation function to the inputs.

3.3 Data Set

The data set used to train and test the models was the reference energy disaggregation data set (REDD), the most commonly used data set to benchmark NILM algorithms. It is composed of power data, in Watts (W), collected from 6 distinct houses located in the state of Massachusetts, in the United States of America.

Each house had its mains electricity and up to 24 different circuits monitored. These individual circuits usually represent single appliances but can also describe the power consumption of a group of them [35], with the circuit of kitchen outlets being a practical example of that. The sampling rate of the mains was 1 Hz, whilst the individual circuits had their power consumption measured approximately every 3 seconds [64].

In order to benchmark the suggested approach across a variety of different power signature types, the target appliances used in this dissertation were the microwave, fridge, dishwasher and washing machine. The fact that they are all high-consuming appliances and are frequently utilised in NILM research, which facilitates comparisons with other studies, also had an impact on the decision to use data from these four appliances.

3.4 Pre-Processing Stage

Pre-processing steps were performed on the raw data so that it could be effectively used to perform energy disaggregation with the proposed model. Split-phase electric power distribution is common in North America and the houses monitored had their mains electricity coming from two distinct lines. These data points were merged by summing the values that were recorded at the same time so that a view of the whole house's electrical consumption could be formed. The same approach was taken when the target appliance had its load monitored across more than one channel. The mains and target appliance's information were combined at first by, once again, merging data points recorded at the same time.

These measurements were then resampled by using bins with a length of 8 seconds and taking their mean, which made the downsampled data, composed of 8 seconds time steps, more robust to outliers. The remaining, if any, invalid data points were imputed using backpropagation, with a limit of 1. This means that they were replaced with the next valid observation only up to 1 consecutive not applicable value, i.e. if there were two or more consecutive invalid values, only the last one would be imputed backwards. After this procedure, the remaining invalid observations were dropped from the data set.

The energy consumption data of the mains and the target appliance was subsequently standardised using the values proposed by Zhang et al. [64], which are described in Table 3.1.

	Mains	Microwave	Fridge	Dishwasher	Washing machine
Mean (Watts)	522	500	200	700	400
Standard deviation (Watts)	814	800	400	1000	700

Table 3.1: Values used to standardise the data for each target appliance network

The data was standardised by taking the mean and dividing it by the standard deviation of the associated distribution, which was either the target appliance or the mains. It is important to note that the data standardisation procedure performed with these values was applied to the whole data set, to ensure consistency between the training, validation and testing sets.

3.5 Data Splits

The training data was obtained from houses 1, 2, 4, 5 and 6, with house 3 being held out in order to be able to test the network's ability to generalise to houses not seen during training. This split was done in order to maximise the amount of available training data as well as the representativeness of the testing set. Some target appliances were not monitored in every training house and these houses were excluded from the training set for those specific networks. Table 3.2 describes the houses used in each of the networks trained.

	Microwave	Fridge	Dishwasher	Washing machine
Training houses	1, 2, 5	1, 2, 5, 6	1, 2, 4, 5, 6	1, 2, 4, 5, 6
Testing houses	3	3	3	3

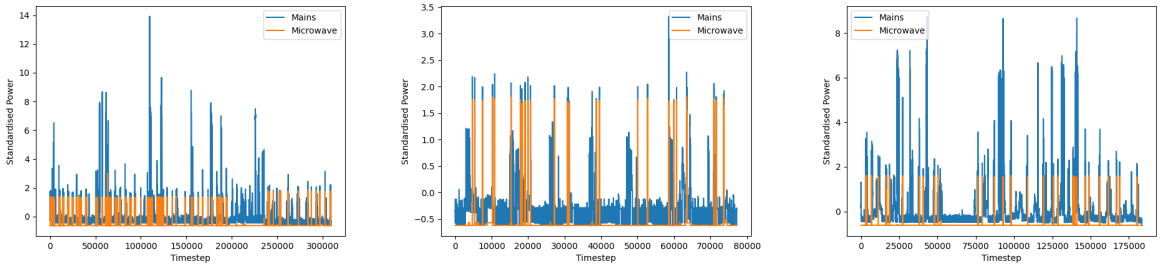
Table 3.2: Training and testing houses used in each target appliance network

To effectively implement early stopping and prevent overfitting, 20% of the training data was used as a validation set. It was used to check the generalisation performance of the model after every epoch and to stop training when the loss on the validation set did not decrease for more than 1e-6 for 5 consecutive iterations over the whole training set. The mean-squared error loss function was utilised to monitor the network's performance on the validation set since it penalises larger errors more harshly than other loss functions, such as the mean absolute error. The data splits utilised for the microwave, fridge, dishwasher and washing machine models can be seen in Figures 3.3, 3.4, 3.5 and 3.6, respectively. In all the splits, the standardised power consumption of the target appliance is shown in orange and the standardised load recorded from the mains is displayed in blue.

Within some of these sets of graphs, the magnitude of the target appliance's power activations appears to be noticeably changing from split to split, but this does not accurately represent reality. It is due to the change in scale of the y-axis, i.e. the change in the maximum power consumption value recorded in each split.

Contrary to what can be seen in the microwave data splits, the fridge training, validation and testing sets have recurrent activations of the target appliance. This corroborates our knowledge of how these appliances are normally used in real-life scenarios, with the fridge being constantly ON whilst the microwave only being sporadically used. The power activations of the dishwasher and the washing machine are even more sparse than the microwave ones, which also confirms our prior knowledge.

Moreover, the microwave activations present in the data set are somewhat stable regarding their power consumption, whilst the fridge, dishwasher and washing machine show more inconsistent values in load drawn during their activations.

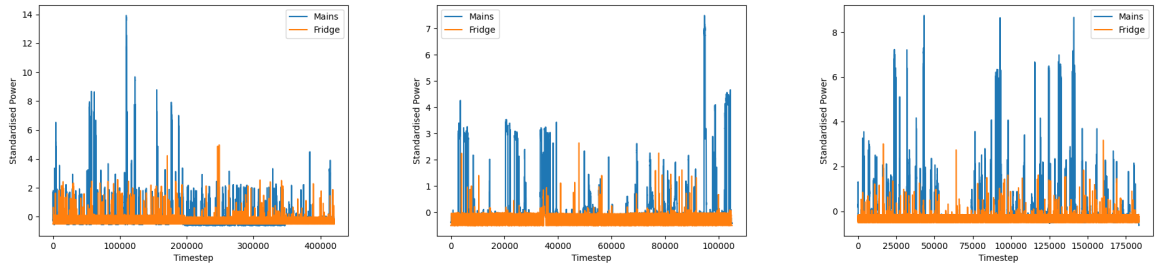


(a) Microwave network's training set

(b) Microwave network's validation set

(c) Microwave network's testing set

Figure 3.3: Data splits used for the microwave network

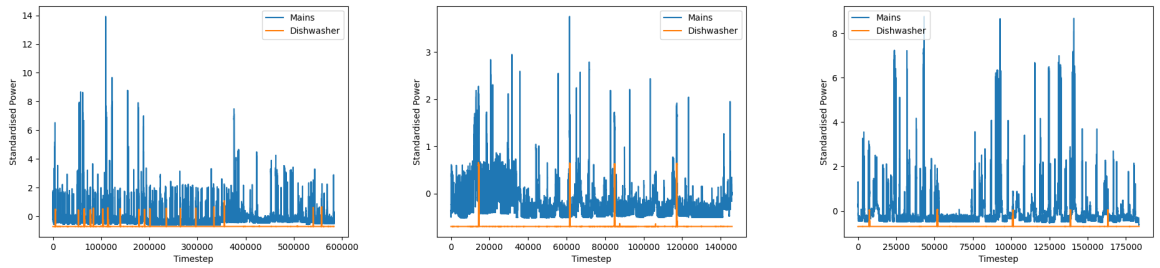


(a) Fridge network's training set

(b) Fridge network's validation set

(c) Fridge network's testing set

Figure 3.4: Data splits used for the fridge network

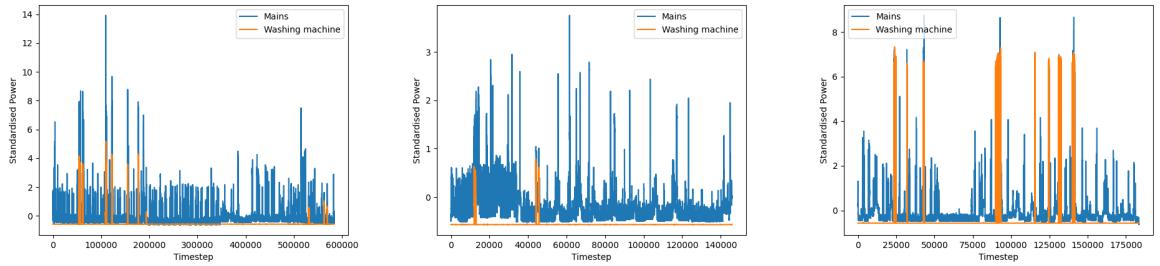


(a) Dishwasher network's training set

(b) Dishwasher network's validation set

(c) Dishwasher network's testing set

Figure 3.5: Data splits used for the dishwasher network



(a) Washing machine network's training set

(b) Washing machine network's validation set

(c) Washing machine network's testing set

Figure 3.6: Data splits used for the washing machine network

3.6 Sliding Window and Batch Processing Methods

In order to efficiently structure the problem as supervised learning, the data was segmented into sliding windows instead of feeding the entire time series to the model. It lowered the models' complexity as well as the pressure put on the machine's memory unit. The size of the windows received as input by each model can be found in Table 3.3.

	Microwave	Fridge	Dishwasher	Washing machine
Window length (time steps)	27	455	699	499

Table 3.3: Input window length used in each target appliance network

The window lengths used in each model greatly varied and it can be intuitively thought of as changing the amount of context each network had access to in order to predict the load of the target appliance at the middle point of the input sequence. These values were selected after a comprehensive search over the parameter space so that the best-performing ones could be chosen.

During the training process, mini-batches of 64 windows were used to update the weights of the connections between units. The model would predict the energy consumption of the target appliance at the middle point of 64 distinct windows of aggregated load, compute the associated errors, and then update the weights of its connections accordingly to the mean over these errors. Therefore, in each round of weight updates, the model was effectively trained with a small subset of the whole training set.

When compared to using the whole training set as the batch size, this approach led to a decrease in computational expenses during the training process, although it increased training times due to the much larger number of weight updates and the inherent overhead. This technique also reduced the likelihood of the model being stuck in a local minimum of the loss function during training, due to introducing noise between weight updates.

3.7 Post-Processing Stage

The model was trained with standardised data and thus, to get insights into the performance of the network on the original scale, the test data and the model's predictions were subject to the opposite workflow carried out in the pre-processing stage. The target appliance's consumption values, both the ground truth data and the model's predictions, were multiplied by the standard deviation of the target appliance's training data and then the target appliance's training data mean was added. The same procedure was applied to the aggregate energy consumption values, but with the standard deviation and mean of the mains' training data. The values used in the post-processing stage were the same as the ones displayed in Table 3.1, which were used in the pre-processing.

To reduce the impact of the inaccuracies inherent to the measurement equipment on the model's performance, a threshold of 10 Watts was used and every ground truth value of the target appliance's drawn load below or equal to that amount was mapped to 0 Watts. The predictions made by the networks were also thresholded and the values used for this purpose are shown in Table 3.4.

	Microwave	Fridge	Dishwasher	Washing machine
Predictions threshold (Watts)	815	45	25	40

Table 3.4: Predictions threshold used in each target appliance network

The thresholds account for scenarios in which the models were not particularly confident, by mapping these predictions to 0 Watts. They were used to calculate the confusion matrix of each target appliance network, with energy consumption values below them indicating that the appliance was OFF and values above denoting that it was ON.

3.8 Summary

In this chapter, the energy disaggregation approach proposed in this dissertation and the rationale behind some of the decisions made were explained in depth. The model consists of three one-dimensional convolutional layers, followed by one bidirectional long short-term memory and two dense layers, with normalisation and regularisation units in between. The REDD data set went through various pre-processing steps, such as resampling and standardisation, before being fed to the network. The splits of data and the sliding windows used were also thoroughly discussed. Moreover, mini-batches of 64 windows were employed during training and post-processing was used to convert the model's predictions back to the original scale and mitigate the negative impact of factors inherent to the suggested approach. In the next chapter, the results obtained by applying the previously described workflow are comprehensively discussed and compared to other NILM research.

Chapter 4

Results

4.1 Overview

In this chapter, the outcomes produced by employing the methodology previously outlined are examined in relation to a variety of performance metrics, the visual relationship between the predictions and the actual values, the loss curves seen during training, as well as the training and inference times. Additionally, the results are contrasted with the ones obtained in related studies.

4.2 Microwave Network Results

The microwave artificial neural network, which was trained to predict microwave energy consumption from the mains load, yielded the results displayed in Table 4.1.

Microwave	Precision	Recall	Accuracy	F1-score	RMSE	MAE	SAE
Before post-processing	47.07%	58.20%	99.60%	52.04%	92.18W	12.39W	9.64W
After post-processing	47.07%	58.20%	99.60%	52.04%	92.72W	6.64W	5.36W

Table 4.1: Results of the microwave network on performance metrics

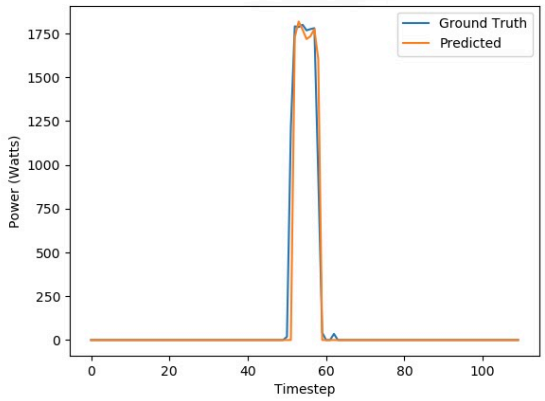
An *accuracy* of 99.60% means that the proportion of correctly classified events, i.e. true positives and negatives, to the total number of time steps is exceptionally high. However, this score can be misleading when it comes to assessing the network's performance, especially in scenarios where the data sets are imbalanced. There are many more observations in which the appliance is OFF and thus, the model could potentially have a large *accuracy* value just by consistently predicting low energy consumption values. The *precision* and *recall* values, and hence the $F1$-*score* which consists of the harmonic mean between both, are more reliable in such cases. A *precision* of 47.07% means that the model correctly classifies approximately half of its ON predictions. In other words, the model, roughly, incorrectly predicts a microwave activation for every correctly predicted one. On the other hand, a *recall* value of 58.20% indicates that the network correctly classified around 60% of the total number of activations of the target appliance.

It is also important to note that the effect of post-processing varies across the different performance metrics used. The event detection metrics were not impacted, which was expected considering that the thresholds used to post-process the data and calculate these performance results are the same. Conversely, the energy estimation-based metrics were influenced by the post-processing stage. The mean absolute and signal aggregate errors nearly halved, whilst the root mean squared error practically remained the same. Therefore, it can be deduced that the post-processing steps do not heavily affect the performance of the model in observations where the error is significant because these are the ones that greatly impact the value of the $RMSE$. However, they do have a positive impact on data points where the errors are smaller, which can be seen through the reduction of the MAE and the SAE. The same pattern is observed in the other networks studied under the scope of this dissertation.

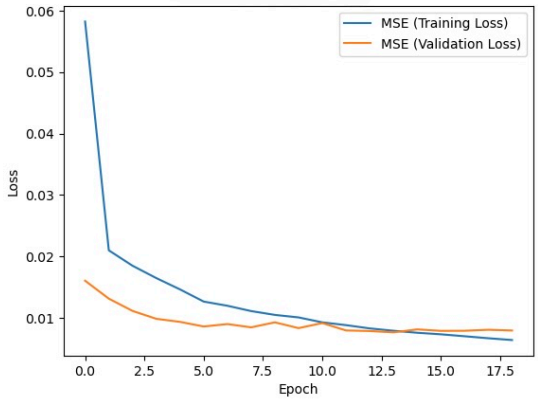
All the EE metrics are expressed in the same units as the data, which enhances their interpretability. A mean absolute error of 6.64W means that the network has, on average, an absolute error of 6.64 Watts at any time step in the testing set. This score can also be misleading, because, as mentioned above, the testing set has many more time steps with OFF states and the much larger errors yielded by

the model in ON observations could potentially be cloaked. Therefore, it is important to analyse the model through other energy estimation-based performance metrics, such as $RMSE$ and SAE. The root mean squared error penalises more severely larger errors, giving a better baseline method to compare competing approaches. An $RMSE$ of 92.72 Watts, although harder to intuitively understand, can be thought of as a good approximation of the standard deviation of the distribution of the errors yielded by the network. In order to compute the signal aggregate error, a window length of 450 time steps was used, which corresponds to approximately one hour of data. Hence, an SAE of 5.36W indicates that the model has, on average, a mean absolute error of 5.36 Watts between the predicted and the ground truth energy consumption values of the target appliance within hour-long periods in the testing set.

In Figure 4.1a it can be seen that the model remarkably identifies microwave activations and that the estimates of power consumption are extremely close to the ground truth observations. Furthermore, in Figure 4.1b the training and validation loss curves seen during the training process are shown. The values shown on the y-axis of Figure 4.1a are in the original scale of the data, whilst the ones displayed in Figure 4.1b are not, since these losses were computed from the standardised data during training.



(a) Microwave network's predictions on a single activation of the testing set



(b) Microwave network's training and validation loss curves

Figure 4.1: Results on the testing set and training history of the microwave network

The model had relatively small loss values and reduced the training loss at the end of every epoch, which means that the model complexity was adequate for the problem. If the complexity was too small, then the model would not be able to learn, thus having large loss values and these would not steadily decrease over time. To account for the fact that the model's complexity might have been too high, early stopping was implemented and it can be seen that the network started overfitting the training data after the 14^{th} epoch. The validation loss began to increase and after 5 epochs without reducing the loss on the validation set, the training was stopped and the best model was restored and saved. It is also worth mentioning that there is not a significant difference between the two loss curves and that the validation loss does not show an unusually noisy trajectory around the training curve. These two factors hint that both the training and validation splits derive from the same data distribution, and thus are likely to represent the problem well.

The training and inference times of the microwave model are displayed in Table 4.2. Training the model on the whole training data set, which consisted of energy consumption data collected over almost one month, took around 14 minutes. This value includes the training process itself and the time taken to evaluate the performance of the model at the end of every epoch on the validation set, which consisted of roughly a week of data. The inference time of the model was only 5.85 seconds, the best out of the four different models trained, which is the time that the network took to make predictions over the whole testing set. This testing set was made up of data collected over a period of approximately 17 days, which is also the time frame of the testing sets of the remaining models due to all being obtained from house 3.

Microwave	Training	Inference
Duration	14min	5.85sec

Table 4.2: Training and inference times of the microwave network

The inference time on the testing set can give an indication of whether the model would be adequate for real-time load monitoring, in the sense that recurrent offline energy disaggregation is performed. Due to showing the anticipated delay in response by the model in a conceivable real-life scenario, the time taken to predict the load drawn by the target appliance over 1 second of aggregated data was used to analyse this suitability. The microwave network would be able to predict microwave usage over 1 second of aggregated energy consumption measurements in under 4.0×10^{-6} seconds. The delay inherent to the channel of communication that would be used to transfer data so that it could be fed to the model would also have to be considered. However, it is unlikely to have a significant impact on the suitability of the model, as the amount of data being sent and received at any one time would correspond to a small time frame. Therefore, the network would be appropriate for real-time energy disaggregation due to the human brain not being able to detect such small delays in response times.

4.3 Fridge Network Results

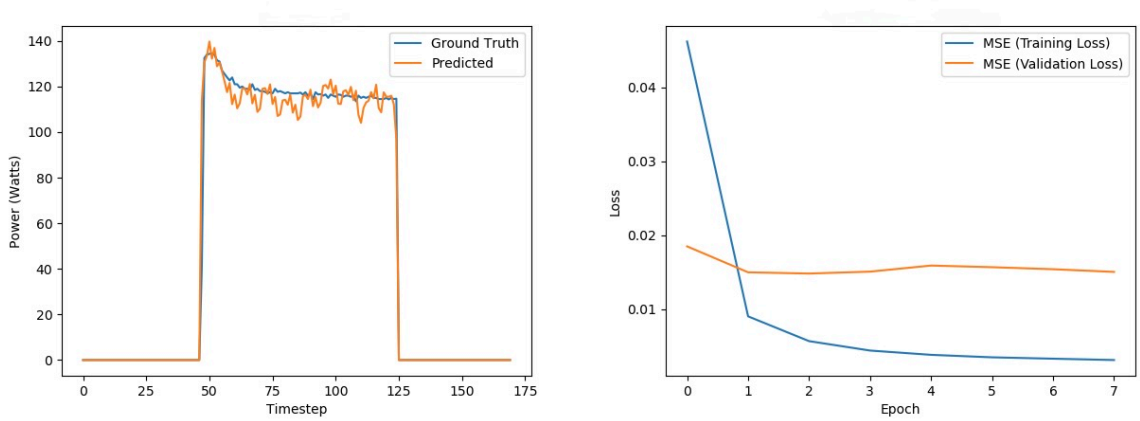
The fridge network results on the chosen performance metrics, both before and after the post-processing stage, are shown in Table 4.3.

Fridge	Precision	Recall	Accuracy	F1-score	RMSE	MAE	SAE
Before post-processing	76.10%	90.19%	85.70%	82.55%	50.34W	32.23W	16.93W
After post-processing	76.10%	90.19%	85.70%	82.55%	51.01W	25.92W	14.03W

Table 4.3: Results of the fridge network on performance metrics

The network correctly identified over 90% of the fridge activations present in the testing set and around 3 in every 4 ON predictions made by the model were accurate. The $F1\text{-}score$ of 82.55% was computed by taking the harmonic mean between *precision* and *recall*, as mentioned earlier. Furthermore, it correctly classified more than 85% of the time steps in the test set with either ON or OFF states. At any time step, the model is expected to have an absolute error of 25.92 Watts and within one-hour-long periods, the network showed, on average, a mean absolute error of 14.03 Watts. Finally, the fridge model showed an $RMSE$ of 51.01 Watts after the post-processing steps.

The model notably captures the activations of the fridge in the testing set, as can be seen in Figure 4.2a, and both training and validation loss curves are displayed in Figure 4.2b.



(a) Fridge network's predictions on a single activation of the testing set

(b) Fridge network's training and validation loss curves

Figure 4.2: Results on the testing set and training history of the fridge network

During training, the network showed continued improvement in the error relative to the training set, whilst the loss yielded on the validation set decreased until the 3rd epoch and then started increasing. However, it should be noted that the gap between the two curves is significant and that the validation loss seems to remain roughly constant. This might indicate that the data splits are unrepresentative of the problem at hand and that it is plausible that they come from different data distributions.

The training and inference times yielded by the fridge model are shown in Table 4.4. For this network, the training data was equivalent to around 39 days of records, the validation to almost 10 days and the test, as previously mentioned, to approximately 17 days.

Fridge	Training	Inference
Duration	59min	41.81sec

Table 4.4: Training and inference times of the fridge network

An inference time of 41.81 seconds on the testing set, means that the model would predict the energy consumed by the fridge over 1 second of aggregated load in under 2.8×10^{-5} seconds.

4.4 Dishwasher Network Results

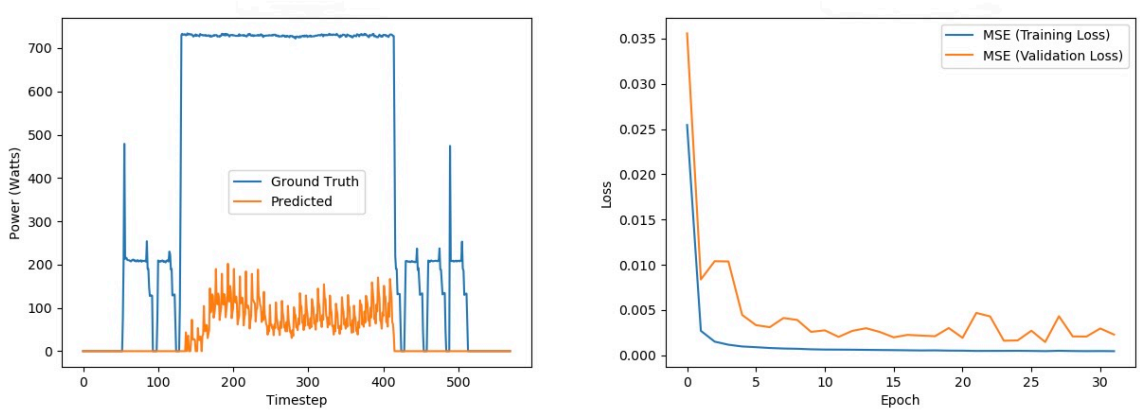
The results obtained by the dishwasher network, which predicts the load drawn by the dishwasher from the aggregated energy usage, are displayed in Table 4.5.

Dishwasher	Precision	Recall	Accuracy	F1-score	RMSE	MAE	SAE
Before post-processing	24.06%	38.78%	97.84%	29.70%	67.07W	11.15W	10.98W
After post-processing	24.06%	38.78%	97.84%	29.70%	67.35W	8.06W	8.00W

Table 4.5: Results of the dishwasher network on performance metrics

Around 1 in every 4 ON predictions made by the model were correct, approximately 40% of the activations in the testing set were accurately classified and roughly 98% of the time steps were correctly assigned to ON or OFF states. Additionally, the network had a mean absolute error of 8.06 Watts on the entire testing set. It also displayed, on average, a mean absolute error of 8.00 Watts between the predicted and the actual energy usage throughout hour-long periods. Finally, the model obtained an $F1\text{-}score$ of 29.70% and an $RMSE$ of 67.35 Watts after post-processing.

Figure 4.3a shows the predictions made by the dishwasher model on a single activation in the test set against the ground truth values and Figure 4.3b displays the training and validation loss curves seen during the training process.



(a) Dishwasher network's predictions on a single activation of the testing set

(b) Dishwasher network's training and validation loss curves

Figure 4.3: Results on the testing set and training history of the dishwasher network

It is apparent that the model identifies the part of the dishwasher activation signature with the largest power consumption values, whilst not demonstrating noticeable recognition of the load drawn in the remaining segments. The results obtained for this target appliance are not as visually appealing as the others, however, it is worth mentioning that the dishwasher model achieved state-of-the-art results, which is discussed in more detail in section 4.6. Additionally, it can be seen that the validation error has a downtrend up to the 27^{th} epoch, the epoch in which the network yielded the best validation loss,

and then starts increasing, showing signs of overfitting. It is also worthy of note that the validation loss curve seen in the dishwasher model training process was the noisiest out of the four trained networks. Nonetheless, both curves had a mostly descending trajectory and did not have a significant difference between them, which hints that the training and validation data splits used may be representative and come from the same distribution, in spite of the validation loss noise.

Table 4.6 displays the training and inference times shown by the dishwasher network. The model took almost 9 hours to train on data monitored throughout approximately 54 days while using roughly 14 days' worth of data as the validation set. Furthermore, it took about 1 minute to predict the dishwasher energy consumption from the mains load on the testing set.

Dishwasher	Training	Inference
Duration	8h 54min	61.68sec

Table 4.6: Training and inference times of the dishwasher network

It would take around 4.2×10^{-5} seconds to predict the load drawn by the dishwasher in 1 second of data from the mains, being the worst inference time seen across the four trained networks.

4.5 Washing Machine Network Results

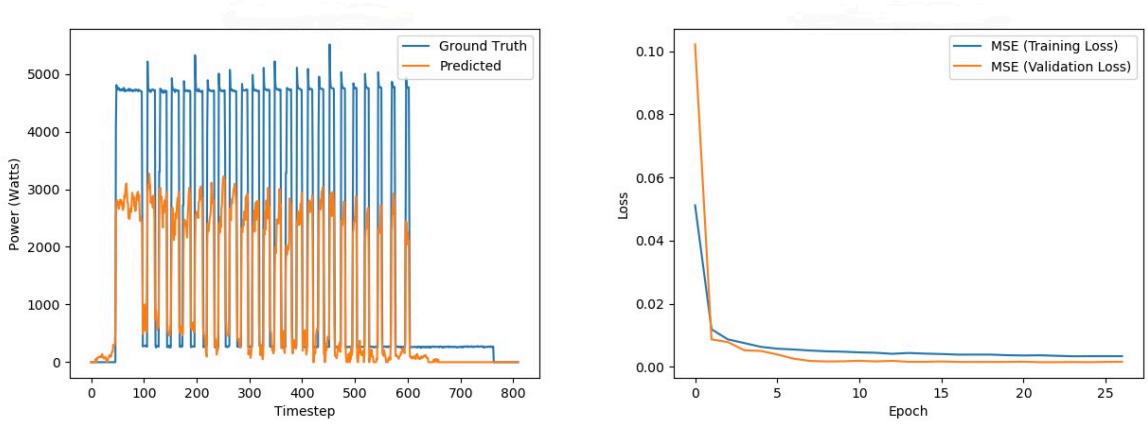
Lastly, the washing machine network, which was trained to model washing machine energy consumption, obtained the results on performance metrics shown in Table 4.7.

Washing machine	Precision	Recall	Accuracy	F1-score	RMSE	MAE	SAE
Before post-processing	89.12%	68.49%	98.32%	77.45%	307.56W	44.95W	42.00W
After post-processing	89.12%	68.49%	98.32%	77.45%	307.53W	43.83W	41.05W

Table 4.7: Results of the washing machine network on performance metrics

Around 90% of the predicted activations were accurate, approximately 70% of the ON states present in the testing set were captured by the model and roughly 98% of the observations were correctly classified. The washing machine model showed a mean absolute error of 43.83 Watts on the whole test set and an average mean absolute error of 41.05 Watts during hour-long intervals. At last, the network obtained an $RMSE$ of 307.53 Watts, after the post-processing stage, and an $F1\text{-}score$ of 77.45%.

The ground truth and predicted power values of a washing machine activation present in the testing set are shown in Figure 4.4a, whilst the training and validation loss curves of the washing machine model are displayed in Figure 4.4b.



(a) Washing machine network's predictions on a single activation of the testing set

(b) Washing machine network's training and validation loss curves

Figure 4.4: Results on the testing set and training history of the washing machine network

It can be seen that the network identifies the ON state of the washing machine and sensibly estimates the power consumption of the target appliance. It is interesting to note that the model shows difficulties in recognising the load drawn during the low energy-consuming segment of the activation, which most likely represents the rinsing and drying stage of a washing cycle. During training, the validation loss had a downtrend until the 22nd epoch and then did not decline for 5 consecutive epochs, dictating the termination of the training process. Both loss curves showed a descending trend and there was not a noteworthy gap between the two, which again suggests that the data splits utilised came from the same distribution and were representative of the problem.

The washing machine model's training and inference durations can be found in Table 4.8. The network took almost 5 hours to train on data recorded across 54 days, with a validation set that had data spanning over two weeks. The training and validation sets of the washing machine network had the same duration as the ones used to train the dishwasher model, as they contained data obtained in the same houses.

Washing machine	Training	Inference
Duration	4h 44min	45.38sec

Table 4.8: Training and inference times of the washing machine network

An inference time of 45.38 seconds indicates that the model would predict the washing machine energy usage from one second of aggregated readings in less than 3.1×10^{-5} seconds.

4.6 Comparison with Other NILM Research

In order to situate this dissertation within the wider research community, a comparison between the proposed network and state-of-the-art approaches was performed. The findings are shown in Table 4.9, with the results on performance metrics of other NILM models retrieved from [47] and the best approach in each scenario highlighted in bold.

Approach	Performance metric	Target appliance			Overall
		Microwave	Fridge	Dishwasher	
Seq2point [64]	F1-score (%)	17.43	75.12	47.66	46.74
	MAE (Watts)	27.13	26.01	24.44	25.86
	SAE (Watts)	18.89	16.24	22.87	19.33
	Training time (hours)	NA	NA	NA	NA
SCANet [10]	F1-score (%)	57.43	83.12	69.21	69.92
	MAE (Watts)	13.75	21.77	10.14	15.22
	SAE (Watts)	9.97	14.05	8.12	10.71
	Training time (hours)	2.68	3.24	6.44	4.12
LDwA [47]	F1-score (%)	**69.01**	**86.76**	**74.41**	**76.73**
	MAE (Watts)	11.17	**19.81**	8.65	**13.21**
	SAE (Watts)	7.12	**13.23**	**6.94**	**9.10**
	Training time (hours)	1.88	3.07	**5.37**	3.44
Proposed	F1-score (%)	52.04	82.55	29.70	54.76
	MAE (Watts)	**6.64**	25.92	**8.06**	13.54
	SAE (Watts)	**5.36**	14.03	8.00	9.13
	Training time (hours)	**0.23**	**0.98**	8.90	**3.37**

Table 4.9: Comparison of the results obtained by the proposed approach with other NILM research

The proposed model outperformed the previous state-of-the-art approach, the load disaggregation with attention (LDwA) network [47], in nearly all the performance metrics when it comes to microwave energy disaggregation. The MAE was reduced by around 40.56%, the SAE by approximately 24.72% and the training time was roughly 12.23% of the time taken by the previous best approach. Moreover, the proposed network yielded close results to the ones shown by the state-of-the-art model regarding fridge load monitoring, with less than $\frac{1}{3}$ of the training time required. Finally, the proposed approach also showed an improvement of almost 7% in MAE on dishwasher energy consumption predictions when compared to the previous best network.

It is important to note that the proposed approach showed significantly better results than the one on which it was based, the seq2point network proposed by Zhang et al. [64], in all performance metrics besides the $F1$-*score* seen on dishwasher load monitoring.

The LDwA model and the scale- and context-aware network (SCANet), proposed by Piccialli et al. [47] and Chen et al. [10] respectively, were not tested on washing machine data. Zhang et al. [64] tested their seq2point model on the washing machine, albeit with a substantially different approach than the one used in this dissertation. Therefore, the washing machine results were not covered in this comparison with state-of-the-art approaches.

4.7 Summary

In this chapter, the results obtained with the NILM approach proposed in this thesis were exhaustively presented and analysed. All the trained networks but the dishwasher model appeared to notably recognise and estimate the energy consumption of single target appliance activations in the test set. The suggested model showed state-of-the-art results on microwave energy disaggregation, with a reduction of around 88% in training time when compared to the previous best. It also achieved state-of-the-art MAE results on dishwasher load monitoring and near state-of-the-art results on fridge energy disaggregation with less than a third of the training time. In the next chapter, the suggested procedure is critically evaluated and future research streams are discussed.

Chapter 5

Critical Evaluation

5.1 Overview

In this chapter, future research directions are outlined along with a critical evaluation of the key components of the proposed approach. Some of the decisions made throughout the project are discussed, as well as consequent strengths, weaknesses and potential improvements in the suggested method.

5.2 Proposed ANN Architecture

Various architectures combining different numbers of convolutional, bidirectional-LSTM and dense layers were tested to find the best-performing one. The depth of the network was shown to have a significant impact on the results obtained, with deeper networks yielding signs of overfitting early on in the training process and shallower models having difficulties in learning from the training set and displaying large validation losses. The proposed model exhibited the best performance out of the ones tested, which suggests that a good balance was found between the expressibility and complexity of the network.

Hyperparameter tuning was performed by analysing previous NILM research and by trial and error. As previously mentioned, in initial convolutional layers, the features identified are simpler and more generic and as we go deeper into the network they become more abstract, being able to model complex patterns. As we go further into the architecture, the convolutional layers have more filters (32, 64 and 128) and the size of these kernels decreases (7, 5 and 3) to allow the model to represent progressively more complex features and take into account smaller regions in the data to detect these patterns. This was the approach that showed the best results on the validation set. The hyperparameter 'return_sequences' was set to 'True' to allow the model to make use of information from every hidden unit and not only from the last one, which also significantly increased the network's ability to learn from data and remarkably improved the performance on the validation set. Manual hyperparameter tweaking was done, but there are techniques that could potentially uncover improvements to the network, such as scikit-learn's GridSearchCV method [8]. It was not used due to how time-consuming the process would be taking into account the large amount of training and validation data, but it would be interesting to explore its impact on performance.

Powers of two were used as the number of filters in the convolutional layers and as the number of units in the bidirectional-LSTM and dense layers to follow the convention used across the wider academic community. This controlled the amount of hyperparameter tuning needed during the conception of the model, which also helped to ensure that the network was generic enough to be useful in other scenarios.

Max-pooling layers were shown to significantly decrease training times whilst maintaining performance and thus were used after each convolutional layer. Moreover, normalising the inputs to the activation functions also showed to decrease the training times and stabilise the learning process of the network. Batch normalisation was performed before feeding the results to ReLU activation functions, however, it would be interesting to check the impact of inverting the process and applying this normalisation after the activation function. The ReLU function maps all negative values to 0 and does not change non-negative values, which means that by performing normalisation before applying the activation function, the benefits might be diminished by destabilising the distribution of the values. It would also be worth researching the impact of using different activation functions, such as tanh, even though ReLU makes intuitive sense in the context of energy disaggregation because all negative values of power consumption are not applicable and should be mapped to 0 Watts.

5.3 Data Set

As previously mentioned, the data set used in this dissertation was the reference energy disaggregation data set (REDD), which although provides a good baseline to train and test load monitoring approaches, is not comprehensive enough to accurately describe the models' ability to generalise to unseen scenarios. It would be beneficial to increase the data available to train and test the model by using other data sets, potentially with data from houses in different countries, consumption behaviours, and target appliances' energy efficiency, amongst other factors. This would increase the robustness of the results presented and increase our confidence in the ability of the model to disaggregate energy in houses not seen during the training process. It is worth mentioning, however, that the majority of NILM research is carried out on single data sets.

The proposed model was trained and tested on data from four distinct appliances to check its performance on a broad enough range of power signatures. More appliances could have been used and some of them, likely appliances with only ON/OFF operating statuses like the microwave and kettle, could yield state-of-the-art energy disaggregation results.

The REDD data set is considered to be a low-frequency data set because the sampling rate of the aggregated load drawn is monitored at 1 Hz. Another potential improvement to the suggested approach would be increasing the sampling rate of the data so that finer features could be learnt by the model to predict the target appliances' energy consumption from the mains, which would likely improve the results obtained.

5.4 Pre-Processing Stage

Pre-processing was carried out on the raw data to efficiently train the model, reduce the impact of outliers, and stabilise and reduce the duration of the training procedure. However, some steps could be taken that would likely improve the performance of the trained networks.

Similarly to what was proposed in the post-processing stage, raw values below a certain threshold could be mapped to 0 Watts in order to reduce the impact of noise, such as the errors inherent to the measurement equipment used to monitor power consumption. As mentioned before in Chapter 3, the data was standardised with the values suggested by Zhang et al. in their seq2point paper [64]. This could have caused the mean and standard deviation of the data to slightly deviate from the expected values of 0 and 1, respectively. Therefore, another potential improvement would be to use the mean and standard deviation of the training data to standardise the data.

5.5 Data Splits

As previously mentioned, the data splits used to train, validate and test the proposed approach were done in order to maximise the amount of data available for training and at the same time ensure that all the splits were representative of the problem at hand, i.e. included a reasonable number of activations and these were diversified.

It would likely improve the model's performance if the sets used were balanced so that they have a similar amount of ON and OFF observations. However, it would also be interesting to see whether this would negatively impact performance by changing the ground truth nature of utilisation of some of the target appliances, such as the washing machine, which is usually not used very often in a single day.

5.6 Sliding Window Method

The sliding window size showed a significant impact on the performance of the trained networks. The idea that the best results are achieved by having an input window length that is big enough to capture a whole target appliance activation but at the same time not too large to avoid capturing other activations, suggested by Kelly et al. [32], was confirmed by the experiments done under the scope of this dissertation.

The microwave model was extensively studied in this regard, with several window lengths used and compared to find the best possible model. However, due to constraints related to the training times of the other models, fridge, dishwasher and washing machine, these were not as thoroughly researched. A more extensive study on the best possible window length used for these networks would be beneficial and it might yield clear state-of-the-art results, such as the microwave model did.

Interestingly, the input window lengths showed a strong positive correlation with the number of trainable parameters of the models. Table 5.1 breaks down the number of parameters in each target appliance network.

	Microwave	Fridge	Dishwasher	Washing machine
Trainable parameters	397,697	2,134,401	3,150,209	2,331,009
Non-trainable parameters	704	704	704	704
Total number of parameters	398,401	2,135,105	3,150,913	2,331,713

Table 5.1: Number of parameters in each target appliance network

The relationship between the number of trainable parameters and the input window size was uncovered by noticing that the relative order of the models regarding these two variables is the same. Such is the case due to the model's flatten layers collapsing the dimensions of their inputs and outputting a reshaped version of them, with the same size as their original number of elements. The last dimension of the input to the flatten layer is not affected by the window size, as it always has a value of 256, corresponding to the number of units in the bidirectional LSTM layer. However, the first dimension of the input changes in the same direction as the change in window length, and hence, so does the number of elements in the input and the size of the flatten layer output. Every output of the flatten layer is connected to every unit in the following dense layer, and each one of these connections is considered a trainable parameter because their weights are updated through the gradient descent algorithm. Therefore, the number of trainable parameters rises if the models' input window length increases, and decreases otherwise.

The number of non-trainable parameters remains the same across all the networks because it is characteristic of the proposed architecture. It depends on the sizes of the last dimension of the inputs to the batch normalisation layers, which are fixed. It corresponds to the sum of the number of outputs in the layers preceding the normalisation blocks, namely the three convolutional and the first dense layers.

At last, even though the training times cannot be compared, across all the models, in relation to the input window length due to differences in the size of the training sets used, the inference times can. The duration of the testing sets was the same across all the trained networks and it is interesting to see that larger input window sizes lead to slower inferences. This is likely caused by the greater number of computations needed by models with a larger number of parameters in order to make predictions. Additionally, within a particular target appliance, networks with larger input window lengths also showed longer training times when compared to the ones using smaller windows as input. This is also expected to be related to the number of parameters in the models, as with larger input windows, there will be more parameters to update and thus the training process will take longer.

5.7 Batch Processing Method

Batch sizes of 16, 32, 64, 128, 256, 512 and 1024 windows were used in order to check the impact on performance and training times. These experiments were only done with the microwave network, due to being the model with shorter training times. Because of time constraints, the other models were not examined within this scope, but the findings are likely to generalise.

Larger batch sizes, although had much shorter training times, demonstrated considerably worse performance on the validation set than smaller batch sizes. Small batch sizes took longer to train, and had noisier loss curve trajectories but yielded the best results on the validation set. 64 windows in each batch was the approach used based on the trade-off between training times and generalisation ability and it was also adopted in all the other networks.

5.8 Training and Testing Processes

The loss function used in the training process was the mean squared error to heavily penalise larger errors, characteristic of observations in which the target appliance is ON. It would be interesting to see the impact on the performance of using a user-defined loss function during training, which could be chosen depending on the priorities of the desired network. Cross-validation would also be useful to increase our confidence in the ability of the model to generalise and make sure that the model we are selecting during training is the one that is most likely to perform well on unseen houses. However, the use of cross-validation would have significant negative implications on the training times of the models.

The networks were trained and tested using the lab machines in the Merchant Venturers Building, home of the University of Bristol's Computer Science department. These computers have a 12th Generation Intel® Core™ i7-12700K central processing unit (CPU) and 32 gigabytes of random access memory (RAM). All 12 physical cores were used, which added up to a maximum of 20 threads running at any one time due to Intel® Hyper-Threading Technology [27]. Nonetheless, it would be worth researching the effect on training and inference times of using the Intel® Ultra High Definition (UHD) Graphics 770 and the NVIDIA GeForce RTX 3080 graphics processing units (GPUs) available. They are designed for concurrent multi-threading for high-throughput tasks [27] and thus would likely improve the times yielded by the networks significantly.

5.9 Post-Processing Stage

Different thresholds were used in the post-processing stage to dictate when the predictions made by the model were not significant enough to be considered an activation and thus mapped to 0 Watts. The impact on performance was measured through an extensive search over multiples of five, from 0 Watts up to the maximum value of power drawn by the target appliance.

The microwave model showed the best performances with large thresholds which nudges us to the idea that the model might be overly confident in predicting that the microwave is ON and consuming energy in time steps that in reality, it is not. On the other hand, the remaining trained networks achieved the best results with small thresholds, which indicates that the models do not benefit from these somewhat large modifications. In these cases, the thresholds are simply adjusting small inaccuracies of the model in predictions near the 0 Watts value.

It would be interesting to explore the impact on the performance of having expert knowledge to define these thresholds. They would likely be set to the minimum known energy usage of the target appliance during an activation across all its different types.

5.10 Summary

In this chapter, the proposed approach was critically analysed regarding its core aspects, namely the data set, data splits and sliding windows utilised, the pre- and post-processing stages, the batch processing technique used, and the model's architecture. Some promising research opportunities, such as the balancing of the data set and more extensive hyperparameter tuning of the network, were also considered. In the next chapter, the main achievements and outcomes of this dissertation are summarised.

Chapter 6

Conclusion

Residential energy consumption significantly contributes to worldwide electricity usage and its magnitude has been increasing. It has been shown that energy consumption feedback, especially at the appliance level, is capable of reducing the energy consumed in households and non-intrusive load monitoring is a viable approach to achieve it. Smart meters can be the single point of energy consumption measure and through energy disaggregation, the load drawn by individual appliances can be predicted. Deep learning generally yields better results than traditional approaches and overcomes some important challenges, such as the need for expert knowledge.

A deep learning workflow was proposed in this thesis, with a deep neural network composed of convolutional, long short-term memory and dense layers with normalisation and regularisation units being suggested. Data from the REDD data set, the most widely used within the NILM academic community, was utilised to test the proposed approach. The raw data was pre-processed through several methods, namely merging, resampling, imputation and standardisation so that it could be in a convenient format to be used by the proposed ANN. Subsequently, the models were trained with a sliding window technique, being structured to receive as input a window of aggregated energy usage and predict the load drawn by the target appliance at the middle point of the input sequence. The windows used in each of the trained networks greatly varied, from 27 time steps in the microwave model to 699 time steps in the dishwasher network. Mini-batches of 64 windows were used during the training process of the models to avoid getting stuck in local minima of the training loss and reduce the associated computational expenses. Thereafter, in the post-processing stage, the predictions made by the models were mapped to 0 Watts if they fell below a predetermined threshold, which takes into account the fact that these networks were not particularly confident at these time steps. The ground truth energy consumption values of the target appliances were also thresholded, with observations below 10 Watts being mapped to 0 Watts to factor in the error inherent to the measurement equipment. Finally, the results were compared with other energy disaggregation research to situate this dissertation within the wider non-intrusive load monitoring literature.

The suggested approach achieved state-of-the-art results in microwave energy disaggregation with an improvement in MAE and SAE when compared to the previous best approach of approximately 41% and 25%, respectively. Moreover, these results were obtained with a reduction in training time of around 88%. The model also performed remarkably well on all the remaining appliances studied, with near state-of-the-art results on fridge load monitoring with less than a third of the training time and state-of-the-art MAE on dishwasher energy disaggregation. The performance of the washing machine network could not be compared to previous research due to incompatibilities in the way the models were tested, but it appeared to perform well.

Potential improvements to the proposed workflow were suggested in the critical evaluation chapter. More extensive tuning of the model's hyperparameters, the use of other publicly-accessed data sets for energy disaggregation and additional pre-processing steps were some of the recommendations provided for future non-intrusive load monitoring research based on this dissertation.

Bibliography

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

[2] Isiyaku Abubakar, Saifulnizam Khalid, Mohd Mustafa, Hussain Shareef, and Mamunu Mustapha. Application of load monitoring in appliances' energy management - A review. *Renewable and Sustainable Energy Reviews*, 67:235–245, 2017. doi:10.1016/j.rser.2016.09.064.

[3] Aiman Albatayneh, Adel Juaidi, and Francisco Manzano-Agugliaro. The Negative Impact of Electrical Energy Subsidies on the Energy Consumption - Case Study from Jordan. *Energies*, 16(2):981, 2023. doi:10.3390/en16020981.

[4] Anaconda. Anaconda Software Distribution. URL: www.anaconda.com.

[5] Georgios-Fotios Angelis, Christos Timplalexis, Stelios Krinidis, Dimosthenis Ioannidis, and Dimitrios Tzovaras. NILM applications: Literature review of learning approaches, recent developments and challenges. *Energy and Buildings*, 261:111951, 2022. doi:10.1016/j.enbuild.2022.111951.

[6] Carrie Armel, Abhay Gupta, Gireesh Shrimali, and Adrian Albert. Is disaggregation the holy grail of energy efficiency? The case of electricity. *Energy Policy*, 52:213–234, 2013. doi:10.1016/j.enpol.2012.08.062.

[7] Nipun Batra, Jack Kelly, Oliver Parson, Haimonti Dutta, William Knottenbelt, Alex Rogers, Amarjeet Singh, and Mani Srivastava. NILMTK: An Open Source Toolkit for Non-Intrusive Load Monitoring. In *Proceedings of the 5th International Conference on Future Energy Systems*, page 265–276. Association for Computing Machinery, 2014. doi:10.1145/2602044.2602051.

[8] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. *Computing Research Repository (CoRR)*, 2013. arXiv:1309.0238.

[9] Arunav Chakraborty and Diganta Goswami. Prediction of slope stability using multiple linear regression (MLR) and artificial neural network (ANN). *Arabian Journal of Geosciences*, 10, 2017. doi:10.1007/s12517-017-3167-x.

[10] Kunjin Chen, Yu Zhang, Qin Wang, Jun Hu, Hang Fan, and Jinliang He. Scale- and Context-Aware Convolutional Non-Intrusive Load Monitoring. *IEEE Transactions on Power Systems*, 35(3):2362–2373, 2020. doi:10.1109/TPWRS.2019.2953225.

[11] Fabrizio Ciancetta, Giovanni Bucci, Edoardo Fiorucci, Simone Mari, and Andrea Fioravanti. A New Convolutional Neural Network-Based System for NILM Applications. *IEEE Transactions on Instrumentation and Measurement*, 70:1–12, 2021. doi:10.1109/TIM.2020.3035193.

[12] George Dahl, Tara Sainath, and Geoffrey Hinton. Improving deep neural networks for LVCSR using rectified linear units and dropout. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8609–8613, 2013. doi:10.1109/ICASSP.2013.6639346.

[13] Sanket Desai, Rabei Alhadad, Abdun Mahmood, Naveen Chilamkurti, and Seungmin Rho. Multi-State Energy Classifier to Evaluate the Performance of the NILM Algorithm. *Sensors*, 19(23):5236, 2019. doi:10.3390/s19235236.

[14] Pedro do Nascimento. Applications of Deep Learning techniques on NILM. Master's thesis, Universidade Federal do Rio de Janeiro, 2016. URL: www.pee.ufrj.br/index.php/pt/producao-academica/dissertacoes-de-mestrado/2016-1/2016033126-applications-of-deep-learning-techniques-on-nilm/file.

[15] Anthony Faustine, Nerey Mvungi, Shubi Kaijage, and Michael Kisangiri. A Survey on Non-Intrusive Load Monitoring Methodies and Techniques for Energy Disaggregation Problem. *Computing Research Repository (CoRR)*, 2017. arXiv:1703.00785.

[16] Will Gans, Anna Alberini, and Alberto Longo. Smart meter devices and the effect of feedback on residential electricity consumption: Evidence from a natural experiment in Northern Ireland. *Energy Economics*, 36:729–743, 2013. doi:10.1016/j.eneco.2012.11.022.

[17] Zoubin Ghahramani and Michael Jordan. Factorial Hidden Markov Models. In *Machine Learning*, volume 29, page 245–273, 1997. doi:10.1023/A:1007425814087.

[18] George Hart. Nonintrusive appliance load monitoring. *Proceedings of the IEEE*, 80(12):1870–1891, 1992. doi:10.1109/5.192069.

[19] Taha Hassan, Fahad Javed, and Naveed Arshad. An Empirical Investigation of V-I Trajectory Based Load Signatures for Non-Intrusive Load Monitoring. *IEEE Transactions on Smart Grid*, 5(2):870–878, 2014. doi:10.1109/TSG.2013.2271282.

[20] Geoffrey Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *Computing Research Repository (CoRR)*, 2012. arXiv:1207.0580.

[21] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997. doi:10.1162/neco.1997.9.8.1735.

[22] Patrick Huber, Alberto Calatroni, Andreas Rumsch, and Andrew Paice. Review on Deep Neural Networks Applied to Low-Frequency NILM. *Energies*, 14(9):2390, 2021. doi:10.3390/en14092390.

[23] Anders Huss. Hybrid Model Approach to Appliance Load Disaggregation: Expressive appliance modelling by combining convolutional neural networks and hidden semi-Markov models. Master's thesis, KTH Royal Institute of Technology, 2015. URL: https://kth.diva-portal.org/smash/get/diva2:881880/FULLTEXT01.pdf.

[24] International Energy Agency (IEA). Global Energy and CO2 Status Report, 2019. URL: www.iea.org/reports/global-energy-co2-status-report-2019.

[25] International Energy Agency (IEA). Electricity consumption: Overview, 2021. URL: www.iea.org/reports/electricity-information-overview/electricity-consumption.

[26] International Energy Agency (IEA). Global Energy Crisis, 2022. URL: www.iea.org/topics/global-energy-crisis.

[27] Intel®. Intel® Core™ i7-12700K Processor Specifications. URL: www.intel.co.uk/content/www/uk/en/products/sku/134594/intel-core-i712700k-processor-25m-cache-up-to-5-00-ghz/specifications.html.

[28] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 448–456, 2015. URL: https://proceedings.mlr.press/v37/ioffe15.html.

[29] Anil Jain, Jianchang Mao, and Moidin Mohiuddin. Artificial neural networks: a tutorial. *Computer*, 29(3):31–44, 1996. doi:10.1109/2.485891.

[30] Andreas Kamilaris and Francesc Prenafeta-Boldú. Deep learning in agriculture: A survey. *Computers and Electronics in Agriculture*, 147:70–90, 2018. doi:10.1016/j.compag.2018.02.016.

[31] Maria Kaselimi, Nikolaos Doulamis, Athanasios Voulodimos, Eftychios Protopapadakis, and Anastasios Doulamis. Context Aware Energy Disaggregation Using Adaptive Bidirectional LSTM Models. *IEEE Transactions on Smart Grid*, 11(4):3054–3067, 2020. doi:10.1109/TSG.2020.2974347.

[32] Jack Kelly and William Knottenbelt. Neural NILM: Deep Neural Networks Applied to Energy Disaggregation. In *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, page 55–64, 2015. doi:10.1145/2821650.2821672.

[33] Jack Kelly and William Knottenbelt. Does disaggregated electricity feedback reduce domestic electricity consumption? A systematic review of the literature. *Computing Research Repository (CoRR)*, 2016. arXiv:1605.00962.

[34] Christoph Klemenjak and Peter Goldsborough. Non-Intrusive Load Monitoring: A Review and Outlook. *Computing Research Repository (CoRR)*, 2016. arXiv:1610.01191.

[35] Zico Kolter and Matthew Johnson. REDD: A Public Data Set for Energy Disaggregation Research. *Proceedings of the SustKDD workshop on Data Mining Applications in Sustainability*, 25, 2011. URL: https://people.csail.mit.edu/mattjj/papers/kddsust2011.pdf.

[36] Christopher Laughman, Kwangduk Lee, Robert Cox, Steven Shaw, Steven Leeb, Leslie Norford, and Peter Armstrong. Power signature analysis. *IEEE Power and Energy Magazine*, 1(2):56–63, 2003. doi:10.1109/MPAE.2003.1192027.

[37] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, 2015. doi:10.1038/nature14539.

[38] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi:10.1109/5.726791.

[39] Gregory Ledva, Laura Balzano, and Johanna Mathieu. Real-Time Energy Disaggregation of a Distribution Feeder's Demand Using Online Learning. *IEEE Transactions on Power Systems*, 33(5):4730–4740, 2018. doi:10.1109/TPWRS.2018.2800535.

[40] Stephen Makonin, Fred Popowich, Ivan Bajić, Bob Gill, and Lyn Bartram. Exploiting HMM Sparsity to Perform Online Real-Time Nonintrusive Load Monitoring. *IEEE Transactions on Smart Grid*, 7(6):2575–2585, 2016. doi:10.1109/TSG.2015.2494592.

[41] Djauhar Manfaat, Alex Duffy, and Byung Lee. Review of pattern matching approaches. *The Knowledge Engineering Review*, 11(2):161–189, 1996. doi:10.1017/S0269888900007815.

[42] Tom Mitchell. *Machine Learning*. McGraw-Hill, 1997. URL: www.cs.cmu.edu/~tom/files/MachineLearningTomMitchell.pdf.

[43] Christoforos Nalmpantis and Dimitris Vrakas. Machine Learning Approaches for Non-Intrusive Load Monitoring: From Qualitative to Quantitative Comparation. *Artificial Intelligence Review*, 52(1):217–243, 2019. doi:10.1007/s10462-018-9613-7.

[44] José Naranjo-Torres, Marco Mora, Ruber Hernández-García, Ricardo Barrientos, Claudio Fredes, and Andres Valenzuela. A Review of Convolutional Neural Network Applied to Fruit Image Processing. *Applied Sciences*, 10(10):3443, 2020. doi:10.3390/app10103443.

[45] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *Proceedings of the 30th International Conference on Machine Learning*, 28(3):1310–1318, 2013. URL: https://proceedings.mlr.press/v28/pascanu13.html.

[46] Lucas Pereira and Nuno Nunes. Performance evaluation in non-intrusive load monitoring: Datasets, metrics, and tools - A review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(6), 2018. doi:10.1002/widm.1265.

[47] Veronica Piccialli and Antonio Sudoso. Improving Non-Intrusive Load Disaggregation through an Attention-Based Deep Neural Network. *Energies*, 14(4):847, 2021. doi:10.3390/en14040847.

[48] Miguel Pérez-Enciso and Laura Zingaretti. A Guide for Using Deep Learning for Complex Trait Genomic Prediction. *Genes*, 10(7):553, 2019. doi:10.3390/genes10070553.

[49] Antonio Ridi, Christophe Gisler, and Jean Hennebert. A Survey on Intrusive Load Monitoring for Appliance Recognition. In *22nd International Conference on Pattern Recognition*, pages 3702–3707, 2014. doi:10.1109/ICPR.2014.636.

[50] Hannah Ritchie. How have the world's energy sources changed over the last two centuries? *Our World in Data*, 2021. URL: www.ourworldindata.org/global-energy-200-years.

[51] Antonio Ruzzelli, Nicolas Cimetiere, Anthony Schoofs, and Gregory O'Hare. Real-Time Recognition and Profiling of Appliances through a Single Electricity Sensor. In *7th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 1–9, 2010. doi:10.1109/SECON.2010.5508244.

[52] Nasrin Sadeghianpourhamami, Joeri Ruyssinck, Dirk Deschrijver, Tom Dhaene, and Chris Develder. Comprehensive feature selection for appliance classification in NILM. *Energy and Buildings*, 151:98–106, 2017. doi:10.1016/j.enbuild.2017.06.042.

[53] Hiroshi Sawada, Nobutaka Ono, Hirokazu Kameoka, Daichi Kitamura, and Hiroshi Saruwatari. A review of blind source separation methods: two converging routes to ILRMA originating from ICA and NMF. *APSIPA Transactions on Signal and Information Processing*, 8(1):e12, 2019. doi:10.1017/ATSIP.2019.5.

[54] Pascal Schirmer and Iosif Mporas. Non-Intrusive Load Monitoring: A Review. *IEEE Transactions on Smart Grid*, 14(1):769–784, 2023. doi:10.1109/TSG.2022.3189598.

[55] Sunil Semwal, Deepak Joshi, Rai Prasad, and Dogga Raveendhra. The practicability of ICA in home appliances load profile separation using current signature: A preliminary study. In *Proceedings of 2013 International Conference on Power, Energy and Control (ICPEC)*, pages 756–759, 2013. doi:10.1109/ICPEC.2013.6527756.

[56] Ajay Shrestha and Ausif Mahmood. Review of Deep Learning Algorithms and Architectures. *IEEE Access*, 7:53040–53065, 2019. doi:10.1109/ACCESS.2019.2912200.

[57] Mark Stamp. A revealing introduction to hidden Markov models. *Science*, pages 1–21, 2004. URL: www.cs.sjsu.edu/~stamp/RUA/HMM.pdf.

[58] Daniel Svozil, Vladimír Kvasnicka, and Jirí Pospichal. Introduction to multi-layer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems*, 39(1):43–62, 1997. doi:10.1016/S0169-7439(97)00061-0.

[59] Huijuan Wang and Wenrong Yang. An Iterative Load Disaggregation Approach Based on Appliance Consumption Pattern. *Applied Sciences*, 8(4):542, 2018. doi:10.3390/app8040542.

[60] Min Xia, Wan'an Liu, Ke Wang, Wenzhu Song, Chunling Chen, and Yaping Li. Non-intrusive load disaggregation based on composite deep long short-term memory network. *Expert Systems with Applications*, 160, 2020. doi:10.1016/j.eswa.2020.113669.

[61] Baran Yildiz, Jose Bilbao, Jonathon Dore, and Alistair Sproul. Recent advances in the analysis of residential electricity consumption and applications of smart meter data. *Applied Energy*, 208:402–427, 2017. doi:10.1016/j.apenergy.2017.10.014.

[62] Liang Yu, Binbin Li, and Bin Jiao. Research and Implementation of CNN Based on TensorFlow. *IOP Conference Series: Materials Science and Engineering*, 490(4), 2019. doi:10.1088/1757-899X/490/4/042022.

[63] Michael Zeifman and Kurt Roth. Nonintrusive appliance load monitoring: Review and outlook. *IEEE Transactions on Consumer Electronics*, 57(1):76–84, 2011. doi:10.1109/TCE.2011.5735484.

[64] Chaoyun Zhang, Mingjun Zhong, Zongzuo Wang, Nigel Goddard, and Charles Sutton. Sequence-to-Point Learning with Neural Networks for Non-Intrusive Load Monitoring. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2018. arXiv:1612.09106.

[65] Tehseen Zia, Dietmar Bruckner, and Adeel Zaidi. A hidden Markov model based procedure for identifying household electric loads. In *37th Annual Conference of the IEEE Industrial Electronics Society (IECON)*, pages 3218–3223, 2011. doi:10.1109/IECON.2011.6119826.

[66] Ahmed Zoha, Alexander Gluhak, Muhammad Imran, and Sutharshan Rajasegarar. Non-Intrusive Load Monitoring Approaches for Disaggregated Energy Sensing: A Survey. *Sensors*, 12(12):16838–16866, 2012. doi:10.3390/s121216838.