POLITECNICO DI TORINO

Master's Degree Thesis in Data Science and Engineering

Master's Degree Thesis

Investigating Deep Learning Generalization Capabilities for Non-Intrusive Load Monitoring from Synthetic to Real-World Datasets

Supervisors

Prof. Giuseppe RIZZO

Dott. Stefano BERGIA

Dott. Marco GALATOLA

Candidate

Marina BENEDETTO

October 2024

Summary

In recent years, the management and the monitoring of the energy consumption has gained more and more importance in different fields, such as industrial, commercial and residential ones. This thesis work investigates the capabilities of the Non-Intrusive Load Monitoring (NILM) technique of accurately reconstructing the active power consumption of individual domestic appliances and identifying the operational state ('On'/'Off'), by taking as input just the aggregated consumption of the household appliances. The aim of this work is to adapt and utilize a synthetic dataset containing both individual and aggregated active power consumption of different devices. The synthetic dataset will be used to train, validate and test the BERT model, which has been specifically adapted for NILM tasks. Afterwards, the model will be evaluated on the real and publicly available UK-DALE dataset in order to assess its ability to generalize.

Acknowledgements

I would like to express my gratitude to my supervisors Prof. Giuseppe Rizzo, Dott. Stefano Bergia and Dott. Marco Galatola for their professionalism, continuous support, and availability. I am truly thankful to them for providing me with the opportunity to carry out my thesis at the LINKS Foundation, a stimulating environment that encourages both intellectual development and professional growth.

Table of Contents

List of Tables

List of Figures

Acronyms

NILM
Non Intrusive Load Monitoring

ML
Machine Learning

AI
Artificial Intelligence

NLP
Natural Language Processing

SVM
Support Vector Machine

NB
Naive Bayes

K-NN
K-Nearest Neighbors

XGBoost
Extreme Gradient Boosting

DL
Deep Learning

CNN
Convolutional Neural Networks

RNNs

Recurrent Neural Networks

ReLU

Rectified Linear Unit

s2p

Sequence to Point

GRU

Gated Recurrent Units

LSTM

Long-Short Term Memory

BERT

Bidirectional Encoder Representations from Transformers

PFFN

Position-wise Feed-Forward Network

GELU

Gaussian Error Linear Unit

MLP

Multi-Layer Perceptron

MSE

Mean Squared Error

KL

Kullback-Leibler

MRE

Mean Relative Error

MAE

Mean Absolute Error

Chapter 1

Introduction

1.1 Problem statement

In the energy field, NILM (Non-Intrusive Load Monitoring) is an important service that is able to recognize and disaggregate the precise load consumption of a specific appliance from the aggregated load signature, by also classifying its operational status at each instance of time. It could happen that in the same house there are several different devices that could start to be used at different instance of time and for different durations. Typically, NILM approach takes as input a sequence of active power values, continously recorded over time in different apartments. NILM algorithm mainly contributes to monitor residents' consumption habits, helping them to potentially save energy; non-intrusive monitoring techniques are usually exploited in residential fields, however they are also implemented for industrial and commercial purposes.

The problem can be mathematically expressed as: $P_t = \sum_{i=1}^{n} p_{i,t} + \epsilon_t$
The total active power consumption at a given time t (P_t) is the sum of the power consumed by all individual appliances and the noise term (ϵ_t). The main objective of the NILM technique is to accurately reconstruct $p_{i,t}$ for each appliance i from the total power consumption P_t and potentially other features derived from P_t.

By analyzing individual load signatures, different categories of appliances can be identified, making the disaggregation process either simpler or more complex, depending on the appliance type:

- Type I: On/Off State. This type of device has only two operational states. Some examples are toaster, boiler, lamps.
- Type II: Finite State Machines. Some household appliances, like dishwasher or stove, register multiple operating states, which are characterized by a repeating

pattern. Thus, this type of devices' signature is easier to recognize.

- Type III: Continously Variable Devices. They do not have a finite number of operational status since their pattern tends to change constantly. Thus, the appliances that belong to this category are hard to identify.

- Type IV: Permanent Consumer Devices. Some appliances, such as TV receiver, remain active and consume energy at all times.

1.2 Current issues and limitations

The application of NILM task on real data would require to record and collect a huge amount of individual and aggregated active power signatures of different types of household appliances. Thus, for this purpose, it would be necessary to install in several houses one meter sensor for each device and one sensor to collect the aggregated consumption pattern. Obviously, this practice would require high costs and very long waiting times, since data would have to be collected for several years. Moreover, collecting information about the energetic consumption of houses' residents would lead to a high risk of lack of residents' privacy, mainly relating to their daily habits. Until now, there have been many attempts to collect and store individual and aggregated load signature of different household appliances. However, these real datasets often have limitations due to their very small size and few consumption patterns records of different devices; thus, these limitations could lead to overfitting during the training and testing phase of a learning model,which may not be able to generalize well. The overfitting problem is a very common condition we could face when we deal with training machine learning or deep learning models, and it refers to the fact that our model learns too well on the training data so that it will not be able to reach high performances on a new dataset.

1.3 Objectives

The first objective of the thesis work is to adapt and exploit a large synthetic dataset, that comprises all the necessary information, to train and test a deep learning model for the NILM tasks. For this purpose, the needed information are: the individual active power trace of different household appliances, operational status (on/off) at each instance of time for each appliance and the aggregated active power. Two are the tasks we aim to solve:

- Regression task. It concerns the reconstruction of the active power signature

of an individual appliance from the aggregated one; thus, it aims to accurately disaggregate the load signature and recognize the consumption of the considered device inside it.

- Classification task. It refers to assign a label 'on' or 'off' that represents the operational status of the device of interest at each instance of time. Obviously, the label 'on' means that, at that moment, the device is turned on and it is consuming power.

Both tasks fall under supervised learning, one of the most common branches of machine learning that exploits labeled training data to help models make accurate predictions. The main goal is to adapt and train a learning model on a big synthetic dataset and then test it on a small real dataset, in order to check whether the model generalizes well and achieves at least the same high performances that have been achieved by some models trained on small and real set of data.

1.4 Value proposition

This thesis work is part of a wider project of EU founded project DATA CELLAR, who aims to create an energy data space that will support the creation, development and management of local energy communities in the EU. Its activity would make possible the implementation of a collaborative platform with the scope of creating a dynamic and secure energy data space. Specifically, a NILM technique must be implemented to contribute and encourage the monitoring and the improvement of energy consumption habits, in order to avoid energy waste. The current project on NILM task has the main goal of exploiting the NILM service to have access to appliance specific signature, directly from whole house consumption measurements; thus, it will not be necessary to install different sensors, one for each appliance in each house, but only one meter sensor for recording the aggregated consumption. Moreover, the usefulness of exploiting a synthetic dataset rather than a real one is related to the fact of being able to reduce the data collection and processing times as well as the cost incurred. In this way, it would not be needed to gather real energetic consumption data but it would also be possible to generate them with characteristics and criteria specifically for our goal. Furthermore, having access to individual electric signature of each house's appliances, rather than the aggregated load one, introduces several benefits both for residents and companies. Firstly, focusing on the houses' inhabitants, the advantage is the opportunity to periodically monitor their energy consumptions, being also able to identify which appliances are more energy intensive. Thus, residents can make more thoughtful decisions and apply strategies to improve their energy habits, by achieving consistent energy and costs savings, simultaneously. Secondly, energetic companies could

also benefit from NILM service because they can better understand the usage and consumption of electric power; this information could help them to implement market segmentation, by clustering consumers based on their similar needs and energetic habits. Moreover, utility companies can also recommend to customers personalized services and products that fits their individual needs, giving also the opportunity to the operators to create a more accurate matching between power supply and demand. In addition, collecting information about how the power energy is consumed by residents could also be useful for load forecasting task. Lastly, the potential energy savings from load energy disaggregation can also have a positive impact on the environment, by reducing its waste.

Chapter 2

Background

2.1 Machine learning

Machine learning (ML) [1] is a branch of artificial intelligence (AI) and computer science that exploits data and algorithms to enable AI to learn human tasks, with the aim of reaching high accurate performances. ML approaches are able to automatically learn from large volume of data and historical information to identify and extract patterns and make predictions, by minimizing human intervention. So, these techniques are capable of extracting and transforming the information into useful knowledge that could be exploited in different real-word applications. By training on a set of data, machine learning techniques develop a model that can generate predictions; the accuracy of the model is then evaluated by comparing its predictions to actual outcomes. Based on this metric, the algorithm is iteratively trained using an augmented learning approach to minimize prediction errors, until the model achieves the desired level of accuracy.

In literature, NILM tasks have been addressed with the following ML approaches:

- Supervised learning: [2] it is one of the most widely used ML approach with the scope of learning a function that maps an input to an output based on input-output matching pairs previously seen. To define a function or a model, a supervised ML technique exploits already labeled training examples. The most common supervised tasks in real world applications are classification and regression analysis. The classification task separates input data and tries to assign to it the correct label. Moreover, classification analysis is characterized by a discrete target variable to predict. While, regression task aims at predicting a continuous target variable, that depends on one or more explanatory and independent variables.

- Unsupervised learning: [2] this task analyzes unlabeled datasets and extracts

hidden patterns, meaningful features and trends directly from input data, without human involvement. Unsupervised techniques are capable of discovering similarities and differences in input information.

In the years, NILM task has been approached with both supervised and unsupervised methods, whose choice typically depends on the amount of available information. Although supervised techniques might be the most immediate and intuitive to use in this field, their application is limited and hampered by the lack of a sufficient volume of labeled data with sub-metered appliances.

2.1.1 Supervised ML tasks

Since this thesis focuses on supervised ML approaches applied to NILM context, the current section will provide an overview of fundamental concepts of supervised classification and regression tasks we are mainly interested in.

Classification analysis

Classification is a supervised learning method in ML that involves predicting a class label for a given example. From a mathematical point of view, there is a mapping function

$$f$$

that links input variables X to output variables Y, assigning each example a specific target label or category [2].

We specifically focus on the binary classification, which involves assigning one of two possible labels to a given example, such as 'yes/no', 'true/false'; in the context of NILM classification task, this means labeling the operational status of a household appliance as either 'On' or 'Off'.

Regression analysis

Regression analysis aims at predicting a continuous target variable (Y) based on the value of one or more explanatory variables (X). Thus, the main difference between classification and regression is that the first one predicts a target label, while the second one predicts a continuous value. Regression techniques are usually used for time series estimation, cost predictions or forecasting, trend analysis, and others.

For the specific NILM task, a linear regression approach is used to break down the total power consumption into the power usage of individual appliances. Linear regression identifies a regression line that is fit by a linear relationship between the dependent variable (Y) and one or more independent variables (X). Mathematically, the total active power consumption at a given time t is represented by the sum of the power consumed by all individual appliances plus a noise term.

2.1.2 Shallow learning approaches for NILM

It is usual to refer to traditional ML approaches of classification and regression with the term 'shallow learning', that exploits the features that have directly been extracted from input data. Since these approaches do not belong to deep-learning ones, they typically assign a great importance to domain knowledge, which becomes extremely necessary to understand the task and reach the final goal. In this section, the most common ML approaches in literature that have been used for the current task will be briefly described.

Support Vector Machine (SVM)

It is a ML technique [1] that can be used for both classification and regression. It is mainly used for the first task for which the SVM algorithm creates an hyper-plane or a set of hyper-planes in high or infinite dimensional space, to better separate different groups. There could be different ways to separate points of different classes but this algorithm aims at choosing the one that maximize the margin, which means the distance between the considered hyper-plane and the nearest training data points in any class. The goal of SVM technique is to reach a strong separation among points of different categories, since the greater the margin, the lower the classifier's generalization error. The selected hyper-plane in N-dimensional space will be the one that distinctly classifies data points.

In [3], SVM algorithm was used to address the NILM task with a linear approach. The authors applied Wavelet Transforms to extract unique features from the power consumption signals of devices; these features were then classified using Support Vector Machines to recognize distinct appliance signatures.

Naive Bayes (NB)

Naive Bayes classifier is a supervised probabilistic classifier that assigns to the data object the most likely class. It exploits the Bayes' Theorem, which is a mathematical rule used to determine the conditional probability of events, based on the hypothesis' prior knowledge about relevant conditions to the considered event.
The Bayes' Theorem is expressed by the following formula:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{2.1}$$

where P(A|B) is the posterior probability, which is the probability of hypothesis A on the observed event B;
P(B|A) is the likelihood probability, which means the probability of the evidence given that the probability of a hypothesis is true;

P(A) is the prior probability of the hypothesis before observing the evidence. As previously stated, this theorem assumes that the probability of the outcome of event A does not depend on the probability of the outcome of event B.

In [4], the Naive Bayes classifier was used to classify individual appliances' status based on their power consumption patterns, assuming the independence condition among each of the appliance's states. This algorithm exploited the features extracted from the power usage data, such as the average power consumption and on/off switching events, and applied probabilistic reasoning to assign a status label to each appliance.

K-Nearest Neighbors (K-NN)

K-NN algorithm [5] is a supervised ML technique based on the concept that data points with similar features should belong to the same class. Usually, the Euclidean distance is computed to measure how much the data points differ each other:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{2.2}$$

Thus, given an unclassified point, it will more likely owned to the class its nearest neighbors belong to. So, after identifying the K nearest neighbors of the considered data point through the distance metric, the most probable label will be assigned to it by majority vote technique: the higher the number of neighbors that belong to a class, the higher the probability that the considered point considered also belongs to the same class.

Specifically for the NILM context, in [6] the authors investigated the capabilities of K-NN algorithm to distinguish between appliances with similar load signatures, in order to enhance the disaggregation process and recognize the individual appliance's usage pattern from the aggregated one.

Decision trees based algorithms

Ensemble methods leverage a collection of weak classifiers, such as decision trees and the final prediction is determined by the majority vote of multiple classifiers.

Random Forest is a machine learning technique that combines the outputs of multiple uncorrelated decision trees to produce a final prediction. This method leverages both the bagging technique and feature randomness, since different and random subsets of features are used to train each model in parallel. RF algorithm was used in [7] to address the NILM classification task, identifying the operational states of multiple household appliances from mixed energy consumption data. This novel approach leverages on a multi-label classification, which allows for identifying multiple active appliances simultaneously, rather than one at a time.
The authors in [8] investigated a method to identify household appliances' energy

consumption patterns, leveraging on key features extracted from their power signals; afterwards, these features were classified using the Extreme Gradient Boosting (XGBoost) algorithm for appliance disaggregation. Similarly to Random Forest technique, XGBoost combines weak models thanks to the boosting technique and creates a final stronger learner. It exploits the Gradient Boosting technique, an extension of the boosting one, which is applied as a gradient descent algorithm to minimize the prediction errors: each shallow decision tree is trained to correct the residual errors of the previous one, thus the final prediction is obtained as the weighted sum of all tree predictions. For this specific thesis' purpose, deep learning techniques were preferred to achieve the NILM tasks due to the complexity and the nature of the problem. Unlike shallow learning methods, deep learning ones are able to handle and extract complex patterns from data, being also capable of extracting discriminative features from time series data. Deep learning techniques such as RNNs, LSTMs, and Transformers are specifically designed to model sequential data and can capture long-term dependencies between power usage events, being able to reach high performances, as proved by results in table 2.1.

2.1.3 Deep Learning

Recent studies have also performed NILM tasks using Deep Learning (DL) approaches: while ML techniques exploit algorithms and statistical models to learn features from input data, DL approaches represent a subset of ML which uses artificial neural networks. Neural networks are able to extract complex patterns and features from large dataset, thanks to their deep structure of several layers. DL models [9] consists of multiple layers: starting from the raw input, each layers applies non-linear transformations to process data into increasingly abstract representations. Especially for classification tasks, high level representations of raw data allow to highlight and extract only the most discriminative features. During the training phase of deep neural networks, the weight vectors are iteratively adjusted by an optimization algorithm to minimize the error, which is the difference between the actual and predicted values. A gradient vector is computed, determining how much the error would increase or decrease with a small change in each parameter. After the backpropagation of the error, the weights are adjusted in the opposite direction of the gradient vector, to reach the local minimum of the objective function.

In recent studies, the following state-of-the-art deep learning architectures have been investigated in the NILM field:

- Convolutional Neural Networks (CNN)
- Recurrent Neural Networks (RNN)
- Transformers

Convolutional neural networks

Convolutional Neural Networks (CNN) are powerful deep learning models that allow to automatically extract features from input data. Its architecture is typically composed of multiple convolutional layers at different levels of depth: the former deal with extracting low level features, while the latter extract more abstract features. Since CNN are characterized by many layers and parameters, their training phase is complex and it usually requires time. In general, the convolutional layer is composed of three stages:

- convolutional, it processes input data as tensors (multi-dimensional matrices) and outputs a matrix of tensors, which represents the extracted features. Each convolutional layer comprises hundreds sliding filters, with multiple trainable weights; each filter continuously moves over the input matrix and operates transformations on small windows. The shallow filters recognize simple and general features, while the deeper filters extract more discriminative characteristics.

- activation, introduces non-linearity during the computation and activates input stymula. The activation function that is typically used in CNN is the Rectified Linear Unit (ReLU); it is the most common used activation function in deep learning models and it is mathematically defined as:

$$f(x) = \max(0, x), \tag{2.3}$$

  thus, it shrinks to 0 all negative input data. ReLU activation function reduces the vanishing gradient problem and does not saturate; moreover, its main advantage is related to the faster computation of loss's gradients during the backpropagation.

- pooling, it performs tensor downsampling by applying a sliding filter on a small window that returns a summary statistic computed on it; the most widely used is maxpooling, which outputs the maximum value over all small considered tensor's windows.

After the convolutional layers, there also are the flatten and the fully connected layers: the first one takes as input the multidimensional output of the last convolutional layer and reduces its dimension to transform it into a single vector; while the second one processes the flattened vector, performs on it a linear transformation, followed by an activation function, then, it outputs the final predictions.

Specifically for NILM, in [10] a sequence to point (s2p) model was proposed to disaggregate individual appliance's consumption pattern from the aggregated signal. The s2p architecture leverages on a sliding window approach over the aggregated

input data: instead of predicting the entire sequence of active power pattern for an appliance, the model predicts only the midpoint value of the window for a specific appliance. Thus, this approach reduces complexity and simplify the training of the model. The authors designed a CNN architecture for the s2p model, as it can handle sequential data and extracting relevant feature from input data. The s2p learning approach has been tested on two different real-world NILM datasets UKDALE and REDD.

Recurrent neural networks

Recurrent Neural Networks (RNNs) are a deep learning models designed to process sequential data, by retaining information from previous states, which evolve during time and represent the input for the current state. At each time step t, RNN receives an input vector x(t), along with the hidden state at the previous time step s(t-1). The network is composed by multiple cells with a set of weights, biases, activation function and feedback loop, that contribute to construct the final output at time t based on the output at the immediately preceding time step. Specifically, given an input vector, the previous state contributes to generate the current state at each step, then the output vector is obtained as a linear combination of the weights and the current state. The training of a RNN concerns the backpropagation of the error across multiple time steps, thus the weights are iteratively updated to minimize the loss, which is given by the difference between the final output and the expected one.
To address the limitations of RNNs, such as the vanishing gradient problem and the difficulty of retaining long-term dependencies, two deep recurrent neural network models were employed for the NILM task in [11]: Gated Recurrent Units (GRU) and Long Short-Term Memory (LSTM).

LSTM is a variation of a RNN; it is used to mainly overcome the vanishing gradient problem and to retain only the important information for a much longer time, by discarding the irrelevant ones. LSTM makes predictions by taking into account the past information too, thus it exploits two separate paths: one for retaining long term memories, one for short term memories. The LSTM cell has three inputs:

- sequential data x_t,
- short memory $h_t - 1$,
- long memory $c_t - 1$.

The generated outputs are:

- updated short memory h_t,

- updated long memory c_t.

To track information over time, LSTM relies on three different gates, which control the flow of information through the cell:

- Forget Gate, computes the percentage of the long term information from the previous cell that should be retained or forgotten; the most irrelevant past data for future predictions will be ignored. The amount of the long term memory to keep is computed by the sigmoid function that outputs for each value a number between 0 and 1: if it is equal to 0 it means that the considered information should be completely forgotten, if it is 1 it should be retained.

- Input Gate, decides how much of the short term information should be added to the retained long term memory; the computation is done with the tanh activation function. The result of the sum represents the updated long term memory.

- Output Gate, decides what information should be passed to the next step. It applies the tanh function to compute the potential information to keep from the updated long term memory, then it will be added to the amount of the short term memory to keep, which is computed by the sigmoid function. The final output is the updated short term memory, which is also known as "hidden state".

The LSTM computational block uses the same weights and biases at each step; furthermore, to manipulate longer sequences, multiple LSTM cells can be stacked sequentially.
While the GRU is a slightly simpler variant of LSTM, which often outperforms LSTM network in terms of convergence time and generalization ability. Similarly to the LSTM, also the GRU relies on gating mechanisms to control the flow of information and retain dependencies of different time scale, but without having a separate memory cells. The GRU is composed of two distinct parts:

- Update Gate computes a linear combination between the amount of the previous hidden state to keep and the percentage of the new information, thus it combines the functions of the forget and input gates in LSTM network.

- Reset Gate, it computes how much of the irrelevant previous hidden state to forget.

Specifically, the authors of [11] introduced regularization methods, like dropout and L2 regularization, to improve the performances of LSTM and GRU models in the NILM task. Both of them were evaluated on UK-DALE and REDD datasets.

Transformers

RNNs and their slight variations, like LSTM and GRU, have been widely used to address sequence to sequence problems and manipulate sequential data. Even if LSTM and GRU architectures have been able to partly overcome the vanishing gradient problem, however the training of the network still suffers from the rapid changes of the loss' gradient; moreover, its computational inefficiency is due to the networks' incapacity of carry out multiple parallel operations, especially when recurrent networks deal with long sequences.

Transformers, presented in [12], are encoder-decoder architectures designed to overcome these limitations. They rely on the attention mechanism to capture and model global dependencies between input and output sequences, regardless of their positional distance Through this mechanism, transformers focus only on the most discriminative tokens of input data to generate the output sequence.
The transformer structure, shown in Figure 2.1, is composed of two parts: the encoder, which maps and process an input sequence to generate a set of high-level representations, and the decoder, which generates the corresponding output sequence, one token at a time.
As shown in the left column of Figure 2.1, the encoder block is composed of 6 stacked identical layers; the overall structure is characterized by:

- Word embedding, it represents each input word as a vector of numbers in a high dimensional space; the basic principle is that the words with a similar semantic meaning tends to present a similar representation in the vector space. This mechanism enables the model to identify semantic relationships among words of the input sequence.

- Positional encoding. Since the Transformer processes input data simultaneously, rather than sequentially as LSTM or RNN does, the order of token in the input sequence is lost. Thus, positional encoding incorporates additional information about the position of each token in the considered sequence. It's crucial to keep track of the positional information of tokens because the order of words in a sentence determines its meaning.

- Multi-Head Self-Attention mechanism, it consists of multiple self-attention layers in parallel, which allows the model to focus on on different positions of the sequence simultaneously. Unlike the RNN that process data sequentially, self-attention technique enables each token in a sequence to focus on all the other tokens, by capturing relationships between pairs of words regardless of their position. For each token, three different vectors are computed: Query (Q), Key (K), and Value (V). These three representations are obtained by multiplying each word embedding with the corresponding weights matrix. Then, the attention score of the i-th token with respect to the j-th token is

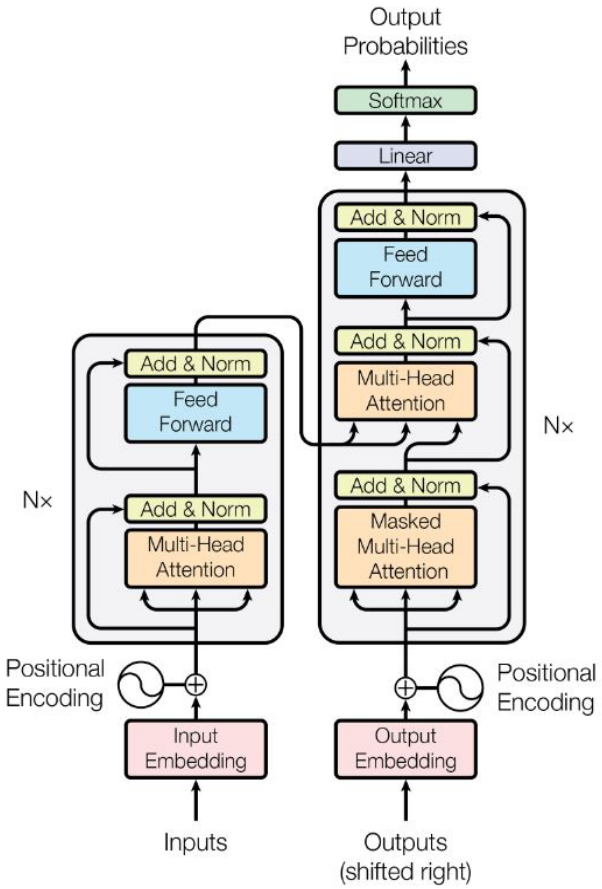

Figure 2.1: The Transformer architecture - Source: "Attention Is All You Need [**12**]"

computed with the dot product between the query vector of the i-th token and the key vector of the j-th one:

$$\text{Score}(Q_i, K_j) = Q_i \cdot K_j \tag{2.4}$$

. The obtained dot product represents the similarity of the pair of considered tokens. Before normalizing the scores with the softmax function and obtaining probabilities, the attention score of each pair of input tokens are scaled by the square root of the dimensionality of the key vectors to prevent extremely large values:

$$\text{Attention score} = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) \tag{2.5}$$

. The final token representation is given by the weighted sum of its value vectors where the weights are the attention scores obtained at the previous

step:

$$\text{Attention Output} = \text{Attention scores} \cdot V \tag{2.6}$$

.

The encoder block exploits multiple self-attention heads to focus on different parts of the input sequence, simultaneously. Each single attention head is associated to a set of weights matrices, one for each query, key and value vector, which are used for the computation of the self-attention score; then, the scores of all the heads attention are concatenated and linearly transformed to produce the final output:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{2.7}$$

$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.8}$$

.

- Position-wise feed-forward network, FFN is typically composed of two fully connected layers and a ReLU activation function; it is called 'position-wise' since it operates on each token of the sequence, independently.
- Residual connections. After each sub-layer a residual connection is added to directly combine both initial positional encoding information and self-attention representation; furthermore, it is also followed by a layer normalization to guarantee a more stable training and to improve model's performances. The normalization also contributes to reduce the covariance shift, by avoiding too high changes in weights adjustment during the gradient descent optimization, and to facilitate a faster convergence of the network.

The attention mechanism establishes multiple pathways that directly link the encoder to the decoder within the transformer architecture; these connections enable the decoder to access to the encoder's outputs which will become its input values. Thus, the decoder's predictions are obtained by considering the most relevant encoder's input data, whose importance is directly proportional to its attention score.
The right column of Figure 2.1 depicts the decoder, whose structure is similar to the encoder one with 6 stacked layers; each of them is characterized by three main components:

- Masked multi-head attention, it is composed of multiple masked self-attention heads that focus on different positions in the input sequence, simultaneously. Unlike the self-attention mechanism of the encoder block, this one applies an additional masking step to prevent the decoder from attending to future

tokens. Thus, at each step, it predicts the token by considering the previously generated words only.

- Multi-head attention. Its attention mechanism is similar to the one used in the encoder block; specifically, it is also known as cross-attention since the self-attention is computed by using the key and the value matrices that come from the encoder and the query which comes from the decoder itself.

- Position-wise feed-forward neural network. It is a fully connected feed-forward network applied to each position, independently, which consists of two linear transformations with a ReLU activation in the middle.

Similar to the encoder, at the beginning, the word embedding creates a vector space representation for each token of the expected output sequence, then the relative positional encoding information is added. Even in the decoder block, each sub-layers is followed by residual connection and layer normalization.
Once the output has passed through all the layers of the decoder block, it is projected into a final linear layer, followed by a softmax function that converts the values into probabilities, for each possible token in the vocabulary.

To address the energy disaggregation task for NILM, recent studies combined the attention mechanism with Transformer architecture.

BERT model

BERT (Bidirectional Encoder Representations from Transformers) is a deep learning model architecture presented in [13]; it is a novel model developed by Google in 2018 that analyzes the text in both directions, across all its layers. This bidirectional mechanism allows it to fully understand the context of a word by considering the words before and after it simultaneously, leading to more accurate language representations. BERT exploits only the self-attention mechanism of the Transformer, which allows it to learn contextual relations between words (or sub-words) in a text; indeed, since BERT aims at understanding and analyze text inputs, only the encoder block is exploited. BERT is pre-trained on large datasets and it can be easily fine-tuned to perform a wide range of tasks. Thus, there are two main steps:

- Pre-training phase, in which the BERT model is trained on a large amount of unlabeled text data according to two different unsupervised tasks: Masked Language (Masked LM) and Next Sentence Prediction.

  - Masked LM. It is an unsupervised task that allows to train a deep bidirectional representations model, by first randomly replace the 15% of input

tokens with the [MASK] token, secondly learning to predict only those missing words. The aim is to predict the original word of the masked token based on the context provided by the other words which are located on its left and on its right. Moreover, as it is shown in Figure 2.2, a classification layer is added on top of the encoder output; then, the output vectors are multiplied by the embedding matrix to transform them into the vocabulary dimension. Finally, the probability of each word in the vocabulary is computed using the softmax function.

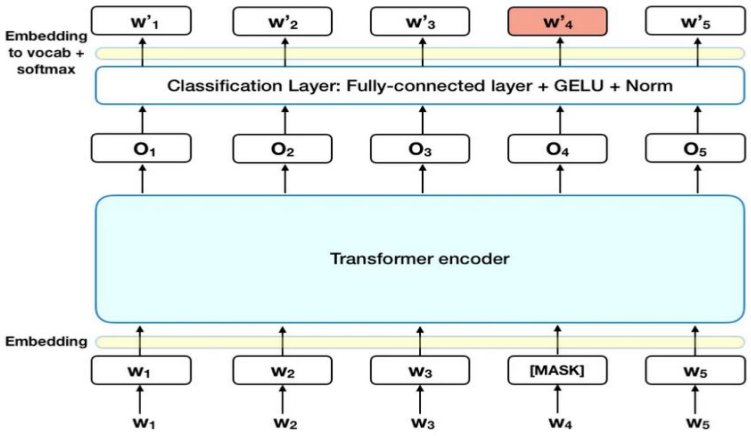

Figure 2.2: Masked ML - Source: "BERT Explained: State of the art language model for NLP" [14]

However, since the [MASK] token is not used for the fine-tuning phase, to avoid this mismatch BERT model does not replace all the random selected tokens with [MASK], but only the 80% of them. While, the 10% of the remaining part is replaced with a random word, leaving the other 10% unchanged.

– Next sentence prediction. Through this downstream task, the model learns how to predict the semantic relationships among sentences, which means understanding if two sentences are sequential or randomly sampled. The model is trained through a self-supervised technique: the 50% of the training input sentences is followed by its real subsequent sentence in the original document, while the remaining 50% is followed by a randomly selected sentence, which is not semantically correlated with the previous one. A [CLS] special token is added before each input sentence, while at the end the [SEP] token is used to distinguish two separate sentences which will be processed by the model during the training phase (Figure 2.3). After the generation of token embeddings, each of them is marked with 'A' or 'B', to identify the first or the second sentence of training pairs,

respectively. Then, the transformer positional encoding adds the positional information of each token in the sequence. Finally, the probability that the second sentence of the pair is connected to the first one is computed by the softmax function of the classification layer.

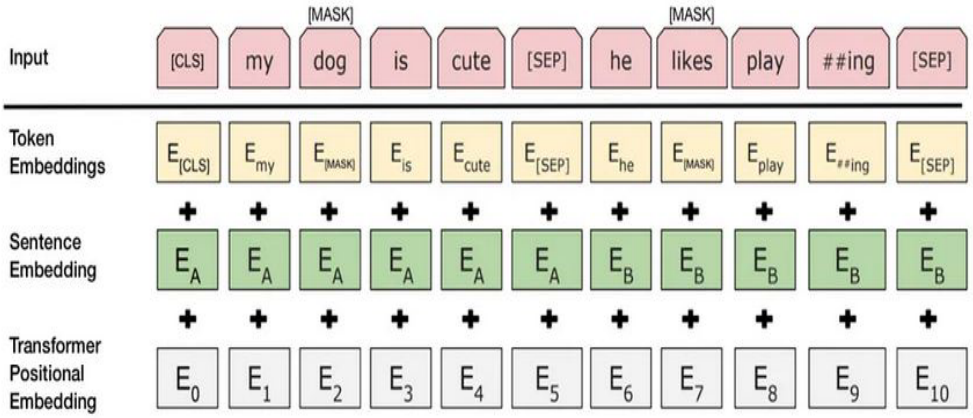

Figure 2.3: Next sentence prediction - Source: "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding [**13**]"

The simultaneous training of these two unsupervised tasks aims at minimizing the combined loss function.

- Fine-tuning phase, in which the parameters of the pre-trained model are exploited to initialize models for different down-stream tasks; then these parameters are fine-tuned on labeled data, so the main advantage is that only few parameters need to be learned from scratch. Usually, a new layer, specifically designed for the downstream task, like a classification head, is added on top of the pre-trained BERT model.

In [15], the authors introduced a novel architecture called BERT4NILM, inspired by the BERT model. This architecture has been tailored for the energy disaggregation tasks by exploiting a specific loss function designed to improve performance in NILM. BERT4NILM model will be discussed in detail in the next section.

Recently, the growing use of attention layers and Transformers has also been registered in NILM field, due to the achievement of high accuracy in performances. However, the adaptation of these techniques for the energy disaggregation and event detection tasks is still challenging. Transformers, typically used in NLP tasks, can be adapted and exploited for NILM tasks due to their ability to capture complex temporal dependencies and relationships in sequential data. Energy consumption data mainly concern time series where, to identify the contribution of individual

appliances to the aggregated pattern, the analysis of long-term dependencies across time should be carried out. For this purpose, the self-attention mechanism in transformers is exploited to allow the model to focus on the most relevant portion of the data, enabling it to efficiently disaggregate energy signals.

[15] provides a comparison among the performances of CNN, GRU, LSTM and BERT4NILM on a real-world dataset, UK-DALE, whose results are reported in Table 2.1.

Device	**Model**	**Acc.**	**F1**	**MRE**	**MAE**
Kettle	GRU+	0.993	0.425	0.008	23.22
	LSTM+	0.994	0.531	0.007	21.26
	CNN	0.997	0.850	0.003	9.64
	BERT	**0.998**	**0.907**	**0.002**	**6.82**
Fridge	GRU+	0.636	0.401	0.901	39.54
	LSTM+	0.573	0.174	0.956	43.74
	CNN	0.772	0.718	0.758	29.20
	BERT	**0.813**	**0.766**	**0.732**	**25.49**
Washer	GRU+	0.342	0.018	0.662	68.65
	LSTM+	0.938	0.150	0.067	15.66
	CNN	0.913	0.173	0.094	11.90
	BERT	**0.966**	**0.325**	**0.040**	**6.98**
Microwave	**GRU+**	**0.996**	0.266	0.014	6.41
	LSTM+	0.995	0.060	0.014	6.55
	CNN	0.995	**0.341**	**0.014**	**6.36**
	BERT	0.995	0.014	0.014	6.57
Dishwasher	**GRU+**	**0.977**	0.639	0.035	38.42
	LSTM+	0.976	0.605	**0.033**	36.36
	CNN	0.947	0.560	0.069	25.43
	BERT	0.966	**0.667**	0.049	**16.18**

Table 2.1: Performance comparison of deep learning models on UK-DALE

As shown in Table 2.1, the BERT4NILM model consistently outperforms other deep learning architectures on the UK-DALE dataset in terms of accuracy and overall performance for the NILM task. This is the reason why this thesis work will focus on the BERT4NILM model to investigate its generalization capabilities when trained on a synthetic dataset and applied to a real-world dataset.

2.2 NILM datasets

2.2.1 Real-World Residential Dataset - UK-DALE

The energy disaggregating technique aims at reconstructing the consumption pattern of an individual household appliance from the whole house aggregated signal. Thus, it would be necessary to train the model on a dataset which includes, for each considered building, both the aggregated active power consumption and the individual active power demand of all household devices. In [16] one of the first real and public dataset for NILM task was presented: UK-DALE. In this dataset the Domestic Appliance-Level Electricity in United Kingdom (UK) buildings has been recorded and collected with two different sampling rates:

- Whole-House Electricity Consumption: it is recorded with a high frequency rate of 16 kHz, which means 16.000 samples per second, by providing a more detailed view of the overall household consumption.
- Individual Appliance Consumption: it is acquired at a low sampling rate of 6 kHz, so one sample is recorded every six seconds.

UK-DALE collects energetic data from 5 houses only, located in UK, constrained by high costs, limited time, and the availability of residents. House 1 is the one in which the recordings lasted longer for 655 days from 54 different channels, one per individual device; while the energetic consumptions from the remaining houses were acquired for months and include less channels. The recorded data cover a period that ranges from 2012 to 2014, differently for each of the 5 houses. Moreover, in three houses the household voltage and current were also recorded and stored at 16 kHz.

Figure 2.4 shows a daily active power consumption in House 1 for the top five appliances, according to the energy usage; each device is represented by a different color line, while the fine grey line displays the aggregated active power demand. Within each residence, the appliances to be monitored were selected in advance, usually those that consume the most energy.

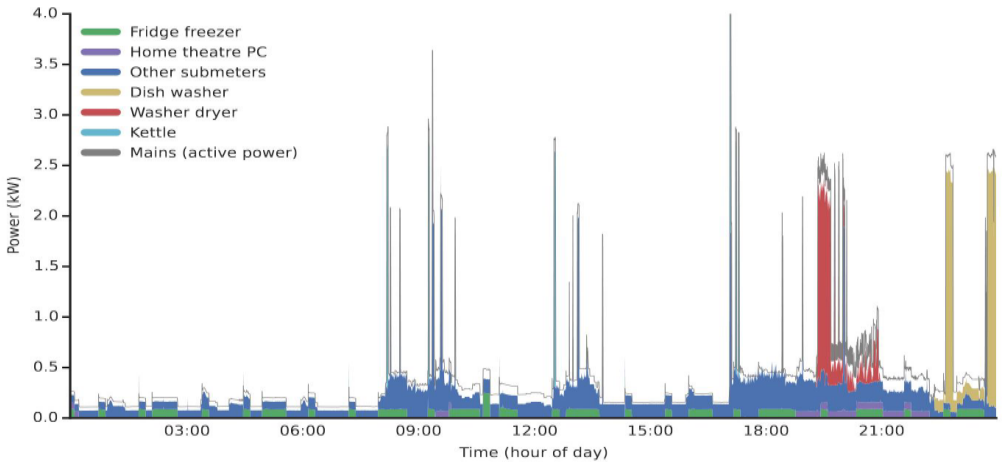

Figure 2.4: House 1 - A typical power demand of a day (Sunday 2014-12-07) - Source: "UK-DALE [16]"

Data collection system

To collect the UK-DALE dataset, a cheap and open-source wireless system was accomplished.
The structure of UK-DALE dataset is organized in five directories, one for each house; each of them contains a set of channels, that are CSV files, one file per meter. Moreover, as stated in a separate labels file, each channel is represented by a positive integer number which is associated with a specific appliance, except for one CSV file that contains the aggregated active power. In each CSV file there are two columns: the first one is UNIX timestamp, whose format is the UTC (Coordinated Universal Time) which represents the number of seconds elapsed since 00:00:00 UTC on January 1, 1970 ; the second column contains the active power data, measured in units of Watts. The particularity of this dataset concerns some metadata files which contains specific properties of each recorded household appliance, like which measurements were acquired by the sensor, in which room the device was used, etc. Other useful information was defined for each appliance to help the identification of On and Off operational status events:

- Max power. It is the maximum power that an appliance can reach when it is fully operational.
- On power threshold. It is the minimum power level at which an appliance is considered to be "on", thus if the active power value is greater than this

threshold, the device is turned on. There are some devices that draw more than 0 Watts when they do not work. Usually, the standard value is 5 Watts, however this threshold strictly depends on the type of appliance.

- Min on duration. It is the minimum amount of time that an appliance must be associated to the 'on' state before it can be considered as truly 'On'.
- Min off duration. It is the minimum amount of time that an appliance must remain in the 'Off' state before it can be considered as truly 'Off'.

Moreover, the authors in [16] observed that gaps lower than 2 minutes are usually associated to radio transmission errors, thus when the gaps last more than that threshold they are filled with zeros and represent appliances when are turned off.

The UK-DALE dataset is one of the most widely used to train and test the NILM model. However, recording and collecting real energy consumption data from each house requires a lot of time and economic resources to get all the necessary equipment.

2.2.2 Synthetic residential dataset - SynD

The goal of this thesis work is to test the generalization capabilities of a ML model for NILM task on the real dataset UK-DALE, after being trained and validated on a synthetic dataset that collects energy data exclusively from residential buildings.

For this purpose, an open-access synthetic dataset SynD, released by the authors in [17], has been tailored. NILM is a ML problem that requires a large amount of data for the training and the validation phase, so a synthetically generated dataset could be a valid alternative to collect a huge quantity of information, by reducing data both the collection time and the equipment costs. Moreover, unlike real world scenarios, synthetic datasets do not suffer from missing records, transcription and miscalculation problems, incorrect data, not aligned timestamps. SynD exploits a dataset generation system which simulates the active power of appliances' consumption for 180 days, by relying on real energy consumption traces of 21 appliances, recorded in two Austrian households. To increase the similarity with other real residential datasets, it also takes into account the type of category the device belongs to and the daily consumption habits of individuals or young couples. SynD dataset includes both individual appliances' active power consumptions and aggregated load signature, which is given by their sum.

Dataset generation approach

The first step of the dataset generation approach concerns the measurement campaign, in which 21 electrical devices was monitored in two Austrian houses; although

the appliances belong to the same category, they register different power consumption patterns in terms of shape and active power values, depending on the type of use or type of brand. Indeed, two brands of fridge can belong to different energy levels or dishwasher programs can consume energy differently, as it is shown in Figure 2.5.

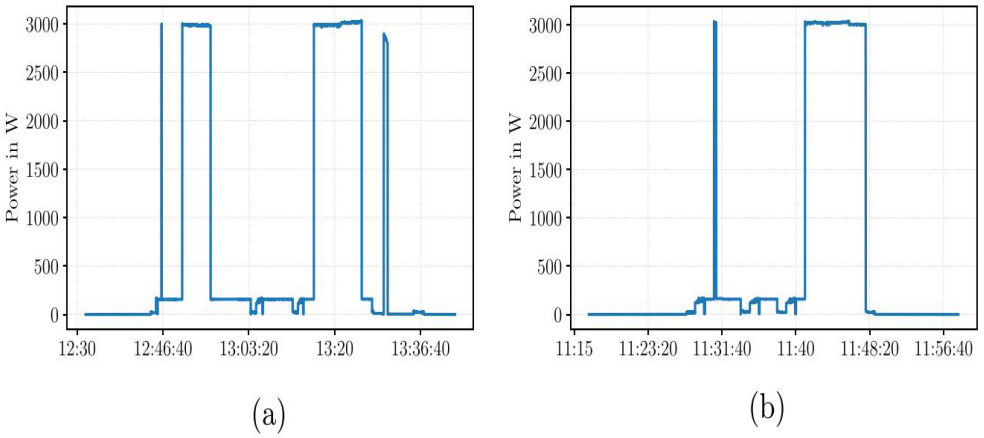

Figure 2.5: Power consumption patterns of two different dishwasher programs - Source: "SynD" [**17**]

Depending on the devices' active power signatures, observed over a sufficiently meaningful time window, and on their difficulty level of reconstruction, the authors of [17] identified four different categories:

- Constantly-On. It includes all electrical appliances which always are turned on, like WiFi router.
- Periodical. This category refers to household devices that are characterized by regular and recurring energy consumption patterns, like fridge.
- Single pattern. It refers to all appliances that start the activity when the resident manually turns them on until the completion of the task. They follow one single pattern, which is always quite similar, thus it should be easy to predict. An example is the water kettle, whose energy consumption pattern usually covers very short periods.
- Multi pattern. This category includes household devices like dishwashers and washing machines with different types of patterns, which differ by length, shape and process steps. Multi pattern appliances are characterized by predefined programs of operation with a predictable end time.

Table 2.2 highlights some of the most widely used household appliances in Synd with the corresponding category they belong to. In this thesis, the analysis will be focused on these five devices, as they are common to both the SynD and UK-DALE datasets.

ID	Household appliance	Category
2	Fridge	Periodical
3	Dishwasher	Multi pattern
5	Washing machine	Multi pattern
8	Microwave	Multi pattern
22	Water kettle	Single pattern

Table 2.2: Categories of some of household appliances in SynD

The second phase of the synthetic dataset generation method involves a dynamic process of consumption signatures interpolation; after the definition of the sampling rate (0.2 s), the duration (180 days) and the power type (active power in Watts), for each considered appliance, the system simulates the active power patterns day by day. For the sake of simplicity, an assumption of independence among consumption patterns of different days was considered, thus the load signatures of one day does not influence the ones of the next days.
The simulation of each device's daily consumption pattern involves the following key steps:

- Select and extract the active power consumption pattern from the real traces, acquired during the collection campaign.
  The choice of the pattern depends on the category the appliance belongs to:
  - If the device belongs to the constantly-on category, the system continuously extends the recorded consumption pattern for the entire day.
  - For the appliances that register a periodical operation, multiple activity patterns were recorded and repeatedly inserted until the end of the day.
  - For single-pattern appliances, the recorded active power usage is associated to different time windows over the entire day.
  - If the appliance belongs to multi-pattern category, the acquired patterns have the same probability of being randomly chosen.

To increase the similarity with a real daily consumption, only for single and multi pattern devices, each selected pattern is associated to a random variable with a uniform probability distribution. Thus, depending on this value, the selected pattern will be inserted or substituted with a null vector, which means that the considered appliance was not turned on that day.

- Interpolate or resample the extracted pattern, if necessary. The interpolation process will be applied only for household appliances which are not characterized by predefined and predictable patterns. Thus, it generates a random number from a uniform distribution, whose parameters differ for each device since they rely on the minimum and the maximum duration of its usage. So, the considered consumption signature will be adapted according to the length of the generated samples, which represent the duration that appliance usage should have.

- Associate the considered consumption pattern to the corresponding time of usage. From specific and predefined time windows, a technique was implemented to randomly select the time at which the considered single or multi pattern appliance is turned on. In this way the active power consumption of the device is randomly spread over different time periods of the day, to better simulate the daily habits of residents. Specifically, according to the predefined time intervals, a uniform distribution is generated for each device; then, a sample of time is obtained which is used as a mean together with the corresponding standard deviation in order to parameterize a normal distribution. Then, from this distribution, the final power-on time of the considered appliance is obtained, by ensuring the independence assumption during the attribution of the starting time for different devices during the day.

The main steps of the simulation process for the SynD dataset are synthesized in Algorithm 1.

Algorithm 1 Simulation process of SynD dataset

Input: Real power consumption traces for each device
Output: Daily consumption pattern for each device
for each device d **do**
    Select consumption pattern:
    if d is constantly-on **then**
        Extend recorded consumption pattern for the entire day.
    else if d has periodical operation **then**
        Repeatedly insert multiple activity patterns until the end of the day.
    else if d has a single pattern **then**
        Assign the recorded usage to different time windows throughout the day.
    else if d has multi-pattern **then**
        Randomly choose a pattern with equal probability for each pattern.
    end if
    Random insertion for single and multi-pattern devices:
    if d is single or multi-pattern **then**
        Draw a random variable $u \sim \mathcal{U}(0, 1)$
        if $u <$ threshold **then**
            Insert the selected pattern into the daily schedule.
        else
            Replace the pattern with a null vector (device not turned on).
        end if
    end if
    Interpolate or resample pattern:
    if d does not have predefined predictable patterns **then**
        Generate random duration $t \sim \mathcal{U}(t_{min}, t_{max})$.
        Resample the consumption pattern to match the duration t.
    end if
    Assign time of usage:
    if d has predefined time windows **then**
        Generate random time t_s from uniform distribution over time windows.
        $t_s \sim \mathcal{U}(T_{start}, T_{end})$
        Use t_s as the mean and generate the final power-on time from $\mathcal{N}(t_s, \sigma^2)$.
    end if
    Ensure independence assumption for starting times across different devices.
end for
Return Daily consumption pattern for all devices.

The active power traces of the measurement campaign were collected and stored in CSV files with a sampling interval of 100 ms. While, the simulated consumption readings for 180 days (which covers a period from year 2019 to 2020) were stored in CSV files, one for each of the 21 household appliances, while the aggregated power consumption series of one device at a time is collected into a separate CSV file. Each file is associated to an integer number which represents a specific appliance, as it is shown in the 'labels' file. Each of them provides the timestamps in human-readable format in the first column and the corresponding measurements in the second one.

As synthetic dataset, we chose SynD, an open-access dataset providing synthetic consumption patterns of different residential appliances, since it has been shown that it reflects a very similar structure of real-world data effectively.
To evaluate the performances of the NILM model, we selected the UK-DALE dataset, the largest publicly available real-world dataset, widely used for experiments. The comparisons with other models is facilitated by its extensive use in research studies.

Chapter 3

Method

3.1 Deep learning NILM approach

Recently, deep learning models have gained more and more importance in the NILM field, especially for the power consumption reconstruction task. In [12] the authors proposed BERT4NILM, a novel architecture inspired to the Bidirectional Encoder Representations from Transformers (BERT), which has been accurately adapted for sequence-to-sequence NILM learning, by also introducing a specific and adequate objective function. After training and testing BERT4NILM model on real datasets, it has been shown that it outperforms state-of-the-art models on UK-DALE. Thus, this novel architecture based on BERT has been chosen and further adapted for this thesis purpose: it has been trained and validate on the synthetic dataset Synd, in order to test its ability to generalize on the real dataset UK-DALE.

3.1.1 BERT4NILM model

BERT4NILM model is based on the BERT architecture, which was originally developed for NLP tasks, and exploits the self-attention mechanism of the Transformer encoder to better capture the sequential patterns in energy consumption data. Thus, BERT4NILM model was specifically designed for NILM task, to disaggregate the total load signature and reconstruct the individual consumption patterns of its components. For this purpose, the authors of [12] proposed a novel loss function that boosts the performances, which are evaluated through a sequence-to-sequence benchmark evaluation for prediction and classification. It is a combination of four key components: a Mean Squared Error (MSE) which computes the average squared difference between predicted and actual power consumption; the Kullback-Leibler (KL) Divergence that compares the predicted and actual probability distributions of appliance states, improving the state classification by focusing on rare events;

soft-margin loss that penalizes relevant misclassifications of appliance states and L1 regularization technique to reduces prediction errors in critical situations.

Moreover, both real datasets REDD and UK-DALE were used in [12] to test the performances of BERT4NILM model.

BERT4NILM architecture

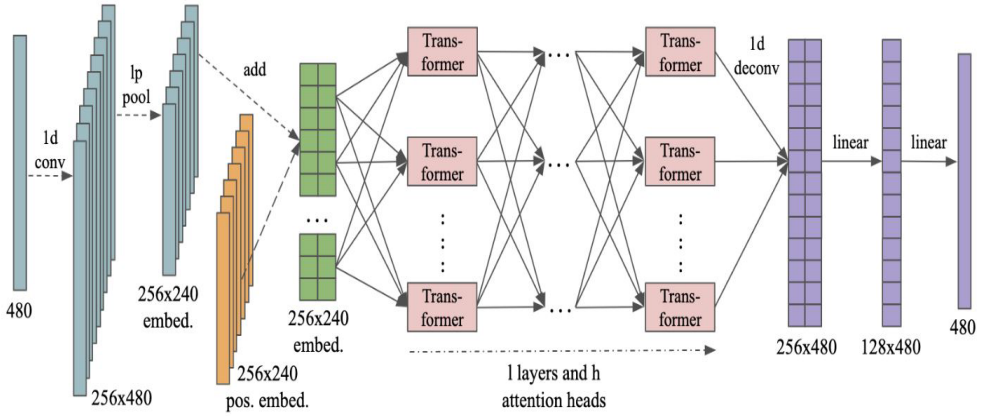

Figure 3.1: BERT4NILM architecture - Source: "BERT4NILM: A Bidirectional Transformer Model for Non-Intrusive Load Monitoring" [12]

The architecture of the model is composed of three main parts, as depicted in Figure 3.1:

- Embedding module.
  The model takes as input the one-dimensional sequential data, which is first processed by a convolutional layer with the aim of extracting key features and increasing its hidden size. Then, the generated output is fed into a pooling layer where the L_2 norm is applied to aggregate features: the L_2 norm, which is mathematically defined as

$$y_i = \sqrt{\sum_{j=1}^{k} x_{i,j}^2} \tag{3.1}$$

  , is applied to a subset of the input sequence, within a pooling window of size k. This step aggregates the values of each window by computing the square root of the sum of their squared values, so that the sequence length is reduced by half while preserving important features. The positional encoding of the sequence is then computed and added to the pooling output. The steps

that are carried out by the embedding module are expressed by the following equation:

$$\text{Embedding}(X) = \text{LPPooling}(\text{Conv}(X)) + E_{\text{pose}} \tag{3.2}$$

- Transformer layers.
  The generated embedding matrix is processed by a bidirectional transformer with l layers and h attention heads per layer. To compute the self-attention, each attention head generates the Query(Q), Key(K) and Value(V) matrices by multiplying the input with the corresponding matrix of weights. So, the attention score is calculated as follow:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.3}$$

  To determine how much attention we should keep on each part of the input sequence, the similarity between Q and K is computed by multiplying the Query matrix with the transpose of the Key one. Then, the similarity score is normalized by dividing it by the square root of the dimension of the Key vectors $\sqrt{d_k}$. Finally, this score is also scaled by the softmax function, which computes the probabilities, that represent the final attention weights. Indeed, thanks to these weights, the model computes a weighted sum of the Value matrix in order to understand which are its most relevant parts for the generation of the final output.
  This self-attention computation is executed multiple times on different subspaces of the input thanks to multi-head attentions, whose results are concatenated and combined together, as shown in the following expression:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W^O \tag{3.4}$$

$$\text{where} \quad \text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{3.5}$$

  Each transformed layer, after the multi-attention head, is followed by a Position-wise Feed Forward Network (PFFN), expressed by:

$$\text{PFFN}(X) = \text{GELU}(XW_1 + b_1)W_2 + b_2 \tag{3.6}$$

  where:
  - x is the input.
  - W_1 and W_2 are weight matrices. They contribute to add complexity to the model: the first one projects the token embeddings into a high-dimensional space, while the second one brings back these features to the original input

size and gets more complex representations.
- b_1 and b_2 are bias vectors.
- GELU is the Gaussian Error Linear Unit activation function. It is widely used in Transformer because its non-linear transformations is particularly effective in capturing complex patterns in data. GELU [14] activation function is expressed by:

$$\text{GELU}(x) = x \cdot \Phi(x) \tag{3.7}$$

, where $\Phi(x)$ is the Cumulative Distribution Function (CDF) of the standard normal distribution.

The PFFN operates on each token of the sequence, independently, and consists on two fully connected layers with a GELU function in the middle.
It precedes residual connections and applies non-linearity transformations on each token's embedding, allowing the model to better capture and transform complex features.
Furthermore, after the feed-forward module, the input features are preserved by the residual connections, followed by layer normalization.

- Multi-Layer Perceptron (MLP) output layer.
  It is composed of:

  – A deconvolutional layer: it is a transposed convolutional layer which restores the length of the output sequence back to its original length, so that the model is capable of making predictions for each time step in the input sequence.

  – Two MLP layers with a Tanh activation function in the middle.
  The output of the deconvolutional layer is passed through a linear layer that applies a linear transformation, by multiplying it by a weight matrix W_1 and adding a bias vector b_1.
  The next step consists of applying the Tanh function to scale the data in the range [-1,1].
  The transformation executed by the second linear layer adjusts the data to the desired output size, which corresponds to the power predictions of the individual appliance.

  The construction of the final output consists of three steps:

  1. Scaling.
     The output predictions assume values in the range [0,1], so they are multiplied by the maximal device power to scale the predictions to realistic power consumption values.

2. Clamping.
   This step is often used to ensure that the outputs of a neural network remain within a realistic or desired range. Indeed, when the model predicts the active power values of a household appliance, usually it is a good practice to limit the prediction within the range of values between 0 and the maximum power value that the considered device can consume.
3. Appliance Status Detection.
   The predictions of the appliance's power consumption values are compared against the predefined thresholds to also determine the status operation of the considered device, which can assume only one of two labels 'On' or 'Off'.

Objective function

In DL models the objective function, also known as loss function, is exploits during the learning process to quantify the error between the predicted outputs and the ground truth, by helping the model to improve its prediction skills at each optimization step. So, the cost function measures the model's performance and how well its predictions match the truthful data. The learning process is optimized by an algorithm that, in our context, has the aim of minimizing the loss function, in order to reduce the predictions errors of the model at each step. Thus, the choice of the objective function is a crucial aspect because it significantly impacts and affects the behavior and performances of the model.

Since the BERT4NILM model was adapted to achieve two tasks simultaneously, which are the individual active power reconstruction of the appliance and the relative status classification, the authors of [12] specifically designed a novel loss function for achieving both tasks.
It is expressed by:

$$\begin{aligned} L(\mathbf{x}, \mathbf{s}) = & \frac{1}{T}\sum_{i=1}^{T}(\hat{\mathbf{x}}_i - \mathbf{x}_i)^2 \\ & + D_{KL}\left(\text{softmax}\left(\frac{\hat{\mathbf{x}}}{\tau}\right) \middle\| \text{softmax}\left(\frac{\mathbf{x}}{\tau}\right)\right) \\ & + \frac{1}{T}\sum_{i=1}^{T}\log\left(1 + \exp(-\hat{\mathbf{s}}_i \mathbf{s}_i)\right) \\ & + \frac{\lambda}{T}\sum_{i \in O}|\hat{\mathbf{x}}_i - \mathbf{x}_i| \end{aligned} \tag{3.8}$$

where x_i,$\hat{x}_i \in [0,1]$ are the ground truth and the predicted active power values of

the device, respectively; while, s_i,$\hat{s}_i \in$ -1,1 represent the actual operational status of the appliance and the predicted labels, respectively.
T represents the total number of time steps in the power usage sequence, so it denotes the length of the input sequence to the model. O stands for the time steps where either the appliance is on (based on actual state labels) or the prediction is incorrect; thus, it considers all the time steps that refers on the most relevant situations, which can have a more impact on the model's performances, for example when the appliance is supposed to be on, or the model's predictions deviate significantly from the truth.
While, τ and λ are two hyperparameters that are introduced to control specific aspects of the model's behavior:

- τ is used to adjust the temperature of the Softmax function applied to the predicted and actual consumption power sequences. In NILM field, a Softmax with a temperature parameter is widely exploited to better capture the variability in different active power usage patterns, which can lead to more accurate energy predictions. Indeed, the hyperparameter τ affects the final probabilities generated by the Softmax function in the following way:
  - high temperature ($\tau > 1$) makes the output distribution smoother, so that the probabilities values are closer together. This means that the model is less confident in its predictions.
  - low temperature ($\tau < 1$) makes the output distribution sharper, so that the largest logits are associated to higher probabilities. Thus, the model is more confident in its predictions.
- λ is a regularization hyperparameter for the absolute error reduction; it can be tuned to reduce the impact of the outliers and to increase the robustness of the model in making accurate predictions.

The novel loss function of the BERT4NILM model can be broken down into four different components:

- Mean Squared Error (MSE) Term. It computes the average squared difference between the predicted and actual active power consumption values over all time steps. This is the most widely used loss function to compute the error of the model in the regression task. This term aims at penalizing the large deviations between the predictions and the ground truth, in order to improve the ability of the model in predicting accurate energy consumption values. The MSE loss function is expressed by:

$$\text{MSE} = \frac{1}{T}\sum_{i=1}^{T}(\hat{x}_i - x_i)^2 \tag{3.9}$$

- Kullback-Leibler (KL) Divergence Term. It quantifies the difference between the softmax-normalized probability distribution of predictions and the actual one, scaled by a temperature parameter τ. Since for the most of timestamps the devices are turned off, this hyperparameter was set to 0.1 to allow the model to focus more on mismatches about 'On' and 'Off' status, expecially for rarely used household appliances like kettle. Its mathematical expression is:

$$D_{KL} = D_{KL}\left(\text{softmax}\left(\frac{\hat{\mathbf{x}}}{\tau}\right) \middle\| \text{softmax}\left(\frac{\mathbf{x}}{\tau}\right)\right) \tag{3.10}$$

- Soft-Margin Loss. This loss penalizes more large incorrect classifications of the operational status of the household appliance, so that the model learns how to correctly assign the label 'On' or 'Off' to the appliance's state. The Soft-Margin Loss is expressed as:

$$\text{Soft-Margin Loss} = \frac{1}{T}\sum_{i=1}^{T}\log\left(1+\exp(-\hat{s}_i s_i)\right) \tag{3.11}$$

- L_1 Regularization Term. To reduce the mismatch between the predicted and the actual energy consumption values, the L_1 term is exploited, especially in cases where appliances are turned on or when there is a misclassification. Its aim is to reduce prediction errors mainly in the most critical situations, fine-tuning the hyperparameter λ to improve the model's performances. This regularization term is expressed by:

$$\frac{\lambda}{T}\sum_{i\in O}|\hat{x}_i - x_i| \tag{3.12}$$

3.1.2 Data preprocessing and loading

Preprocessing of SynD dataset

As previously stated in Section 3.1, in the synthetic dataset SynD the data points are sampled every 200 milliseconds; so, the sampling frequency is 5 Hz, which means 5 measurements are taken per second:

$$\text{Sampling frequency} = \frac{1}{200\,\text{ms}} = \frac{1}{0.2\,\text{s}} = 5\,\text{Hz} \tag{3.13}$$

Thus, for experiments purpose, before splitting the entire dataset into three parts (train, validation and test), it was necessary to resample the synthetic data every six seconds in order to align them to the same sampling frequency of the real UK-DALE dataset. For this purpose, the 'resample' method was used, which is a function provided by the pandas library in Python; typically, this method is exploited to manipulate time-series data and change their frequency, by upsampling or downsampling them, like in our specific case. After resampling the data into 6-second intervals, we exploited a method called "first" to manage any multiple data points that might fall within the same interval: this technique only selects the first data point that appears in each resampled interval, allowing that time period to be represented by a single value. In the context of NILM, the choice of the right aggregation method after downsampling can significantly impacts the quality of analysis and results; however, it usually depends on the specific goals of the work. By keeping the first value, the exact state of the signal at the beginning of each interval is preserved, which can allow to detect sudden changes. These information can be crucial for identifying when appliances are turned on or off, especially those that show sudden transitions in power usage. Moreover, unlike the 'mean' aggregated method, this one retains more detailed data, by keeping the most of the original signal's characteristics and peaks. Furthermore, since in standard residential settings the active power is consumed, its values are positive. Thus, in this specific context, any negative values in individual consumption patterns of each device represent outliers and they must be substituted with zero values before the downsampling technique. The detailed steps of the pre-processing phase are explained by the Algorithm 2.

Algorithm 2 Preprocessing of NILM Data for SynD Dataset

Input: Folder of CSV files (`csv_folder`) and aggregated file (`aggregated_file`)
Output: Processed and downsampled training, validation, and test CSV files
Initialize paths for input CSV files
Load the aggregated data into `df_aggregated`
Convert `timestamp` column to DateTime format
Downsample `df_aggregated` to 6-second intervals
Define and create empty CSV files for training, validation, and test sets
for each device data file in `csv_folder` **do**
    Load device data into `df_device`
    Replace negative values in the data with zero
    Downsample the device data to 6-second intervals
    Merge device data with the aggregated data on `timestamp`
    Split the merged data into training, validation, and test sets
    Append each set to the corresponding CSV file
end for

As already stated, the SynD dataset contains one CSV file for each individual household appliance, which provides timestamps and the relative power usage values, and one single CSV file which collects timestamps and the aggregated active power consumption. According to the column of timestamps, the indices to split the appliance's file into training (60%), validation (20%), and test (20%) sets were calculated. The next step was adding the relative column of aggregated active power pattern by merging these three sets of dataset with the aggregated file, using the timestamps as key. These two operations are repeated for each CSV file of appliance, after the downsampling step. The final output consists of three separated CSV files which will be exploited for training, validation and testing the model; each of them provides both individual and total usage data for all considered appliances, which are identified by the corresponding label.

Dataloader adaptation

Due to the large dimension of the synthetic dataset, loading the entire dataset into memory can be very memory-intensive, especially on systems with limited RAM. Thus, unlike the method proposed by the authors in [12], the dataloader was adapted to separately create train, validation and test loader directly from the three corresponding data subfolders of SynD dataset; each of these folders contains the relative CSV file of training, validation and testing dataset. Since the BERT4NILM model is trained on a single household appliance at a time, chosen by the user, the dataloader provides methods for filtering, loading, processing, and normalizing time-series data that only concern the selected device. Specifically, if BERT model is used, the 'BERTDataset' is applied on training data, which is a specialized dataset class specifically designed for transformer-based models, while the 'NILMDataset' is used for processing validation and testing data.
These two dataloader classes mostly follow the same steps:

- Initialization of attributes for data processing:
  - The path to data folders and the name of the selected appliance.
  - The size of the sliding window and the stride length, which are critical parameters for slicing the data into smaller and manageable chunks for time-series analysis. They allow the model to learn from distinct but potentially overlapping parts of the data. The window size defines the length of each segment of data used for model input, while the stride refers to the number of data points by which the window moves forward to create the next segment, so it controls the overlap between segments and the total number of samples created.
  - Thresholds, minimum on/off durations, and cutoff values, which play critical roles in defining the operational status (on/off) of the considered

appliance and in the management of outliers.

- Loading CSV files: for the purpose, the 'glob' module was used to search for and list all CSV files within the specified directory. The file paths are then stored for subsequent data loading. This technique efficiently loads and handles multiple CSV files.

- Filtering data and computing appliance status: this function reads data from the previously loaded CSV file in chunks, filters the data based only on the specified appliance name, and all the individual consumption values that exceed the cutoff parameter are set to 0. The cutoff value represents the maximum expected power consumption that the considered appliance can reach, so all the values above it are considered outliers. Finally, it computes the status (on/off) of the device at each instance of time and combines these filtered and processed chunks into a single DataFrame.

  To align the data processing with the one of UK-DALE in [12], the computation of the appliance status follows the same basic principle, which exploits threshold, cutoff, min on/off durations parameters, specific for the considered appliance.
  This method analyzes the data to detect when appliances are turned on or off, depending on whether the power consumption exceeds or falls below a certain threshold. The main steps are:

  – The status array is initialized with zero values of the same shape of input data. This array will be used to store the on/off status (0 or 1) for each data point.

  – The initial status is calculated according to the threshold value. So, a boolean array is created, whose elements indicate whether the corresponding data point is above (1) or below (0) the threshold.

  – The difference between consecutive elements in the array of initial status is computed to find the change points. So, the indices of non-zero elements are identified, which represent the points where the status changes from on to off.

  – The events are filtered according to the minimum on and minimum off durations. The off duration is calculated as the time between off and subsequent on events; so, the on events and the off events are extracted only if they meet the minimum duration criteria for being on or off, in order to retain only the meaningful events.

  – According to the detected on events and off events indices, each data points in the array of the initial status is equal to 1 (if on) or 0 (if off).

Thus, the threshold value is the predefined power consumption boundary that an appliance must exceed to be considered "on", so it is used to distinguish between the active (on) and inactive (off) states of an appliance. While, the minimum on/off durations make the state detection operation more stable and realistic by requiring minimum durations for each state, so that false positives are avoided.

- Standardization of the aggregated active power consumption.
  The aim of standardization is scaling the data to get a similar distribution, in order to improve the accuracy of the learning process and speed up the model convergence.
- The next step concerns the length calculation, which takes into account the length of the filtered data, the size of the sliding window, and the stride length. This method computes the total number of data samples generated from the filtered dataset on which the corresponding training, validation and testing pipeline will be iterated.
  To efficiently load the data into the model in manageable chunks, specific windows of data are retrieved by an index, which extracts the relevant portion of the window size, and pads the data if it is shorter than the window size. This technique ensures that all data samples have a uniform length.

  The only difference between the 'BERT Dataset' class, used for training data, and the 'NILM Dataset' class, that was applied on validation and testing data, concerns the way the BERT model accesses to the data samples. In fact, after the extraction of the windowed sequences, 'BERT Dataset' class also implements a masking technique:

  – For each element in the sequence, a probability value is used to randomly generate numbers between 0 and 1; based on the masking probability, the current element will be masked or not.
  – A second probability is generated to determine how to mask the value: 80(%) of chance to replace and hidden the value with -1; 10(%) of chance to replace the value with a random noise generated from a normal distribution, in order to introduce variability into the model; 10(%) of chance to leave the value unchanged. Thus, the model receives the lists of masked tokens, the corresponding ground truth values and status labels that it should be able to predict.

Standardization

Standardization is one of the most widely used scaling technique that transforms the data so that they have a zero mean and a unit standard deviation. Firstly, the

mean and the standard deviation of the data are computed; secondly, by subtracting the mean, each value is centered around it with a unit standard deviation, as shown in 4.1:

$$X_{\text{standardized}} = \frac{X - \text{mean}(X)}{\text{std}(X)} \tag{3.14}$$

Thanks to this technique, data observations rely on a comparable scale, which is a necessary requirement to train machine learning algorithms and obtain reliable results.

In our specific NILM context, the mean and the standard deviation are calculated on the total usage pattern in the filtered training data. These two values will be used to standardize the aggregated power consumption of the validation and test datasets, too.

Chapter 4

Experiments

In this chapter the evaluation metrics will be briefly discussed, as the obtained experiments results of the BERT4NILM model, both on SynD and UK-DALE datasets. The model's performances will be presented and compared to the baseline results of the state-of-art.

4.1 Evaluation metrics

To evaluate the performances of the BERT4NILM model, four metrics were adopted: Accuracy and F1 Score for the classification experiments, while Mean Relative Error (MRE) and Mean Absolute Error (MAE) to evaluate the regression task.

4.1.1 Accuracy

The accuracy metric measures the proportion of correct model's predictions out of all predictions; so, it quantifies how well the model correctly predicts the outcomes. The formula for accuracy is:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{total number of predictions}} \tag{4.1}$$

This metric is the most widely used to evaluate classification models because it is intuitive and reliable when the classes are balanced in the dataset. However, the accuracy metric has some limitations:

- It is not suitable for imbalanced datasets. For example, in our specific case, if the 'Off' class is significantly more frequent than the 'On' one, the model will always tend to predict the off class more easily. So, although the accuracy value is high, the model fails to correctly predict the less frequent class.

- It does not reflect the difference between the false positive and false negative errors, which can have a different cost. The false positive is the number of negative instances incorrectly classified as positive, while the false negative represents the number of positive instances incorrectly classified as negative.

4.1.2 F1 Score

Due to the limitations of the accuracy metric when the model deals with an unbalanced dataset, the F1 score is simultaneously used to evaluate its classification performances. This measure is computed as the harmonic mean of precision and recall metrics, expressed as:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.2}$$

The computation of the harmonic mean allows the F1 Score to penalize extreme values and to balance precision and recall metrics:

- if the precision is high and the recall is low, the F1 score is low, so it means that the model predicts many false negatives
- if the recall is high and the precision is low, the F1 score is low, so it means that the model predicts many false positives

Precision

Precision metric is computed as the ratio of the number of correct positive predictions (true positives) out of the total number of instances the model predicted as positive (both true and false positives). Precision is particularly important in situations where the cost of a false positive is high. The formula of precision is:

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}} \tag{4.3}$$

Recall

Recall is calculated by dividing the number of of correct positive predictions by the number of positive instances. So it measures the model's ability to find all the relevant positive cases in the dataset; it is also known as sensitivity rate. The formula of recall is:

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}} \tag{4.4}$$

4.1.3 MRE

Mean relative error is one of the most widely used metric in ML to evaluate regression problems. For each prediction, the relative error is computed by dividing the difference between the ground truth and the relative predicted value (the absolute error) for the actual value. Finally, for all the samples, the average of the relative errors is calculated. The formula of the MRE is:

$$\text{MRE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{|y_i|} \tag{4.5}$$

where:

- n is the number of observations
- y_i is the actual value of the i-th observation
- $\hat{y}_i$ is the predicted value for the i-th observation

A low value of MRE suggests that the predictions of the model are close to the ground truth values, while if the MRE is high, it means that there is a large difference between the predicted and actual values, indicating poor model's performances.

4.1.4 MAE

Mean absolute error metric quantifies the average magnitude of the absolute errors, which are simply the difference between predicted values and actual values. Its formula is expressed by:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{4.6}$$

The interpretation of this metric is equal to the one of the MRE. The MAE is simpler to compute and interpret than the previous metric, however it does not provide information about the size of the errors compared to the ground truth values.

4.2 Experiments on Fridge appliance

For our experiments, to align and make comparable the results of the BERT4NILM model on the synthetic dataset with the ones on UK-DALE, only the five most commonly used appliances were considered: fridge, washing machine, dishwasher, kettle and microwave.

4.2.1 Dataset exploration and parameters settings

UKDALE

In Figure 4.1, part of the active power consumption of the fridge is shown; for visualization purpose, only a small portion of the energy consumptions of House 1 is considered, specifically the day 23rd January 2013. The fridge is characterized

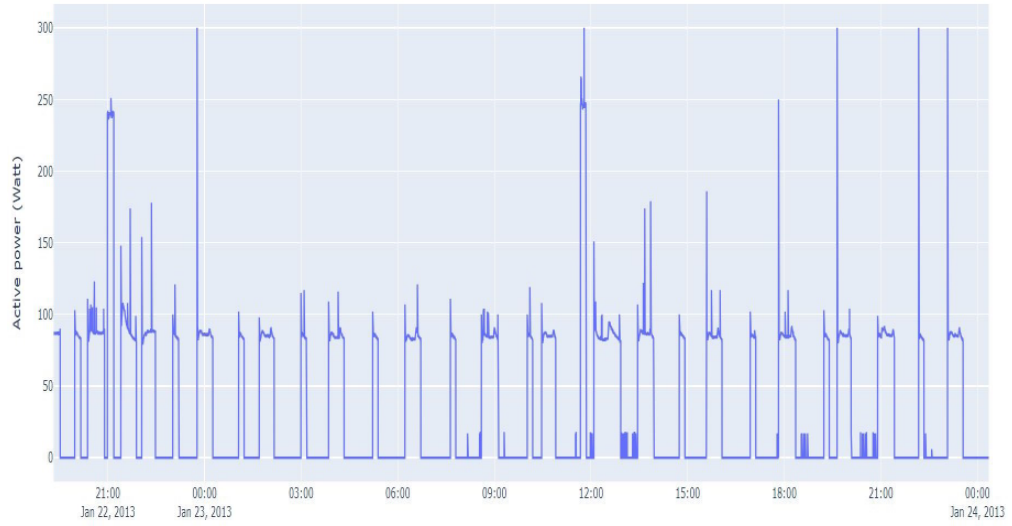

Figure 4.1: Active power consumption of Fridge -UKDALE - 23rd January 2013

by a periodical consumption pattern, with quite regular and recurrent usage cycles, which are usually influenced by the surrounding temperature and by its thermostat settings. From Figure 4.1 it is also clear that its cyclical behavior presents several peaks during the daily consumption; usually, they are caused by the periodical activity of the compressor, which is characterized by on cycles and off cycles. The compressor is one of the most important component of a refrigerator because its aim is to keep the internal temperature stable by removing the heat from inside the fridge and releasing it outside. Thus, when the compressor of a traditional fridge is turned on, the active power consumption significantly increases between a range of values from 100 to 300 Watts; this happens more frequently during the

summer season or every time the fridge's door is opened.

To also visualize and analyze the distribution and the spread of power consumptions data of the fridge, its distribution of frequency was generated and displayed in Figure 4.2; it analyzes the frequency of different values in the dataset. The obtained output is a histogram, a bar chart which divides the data into intervals (or bins), whose height represents the number of data points that fall into each interval.

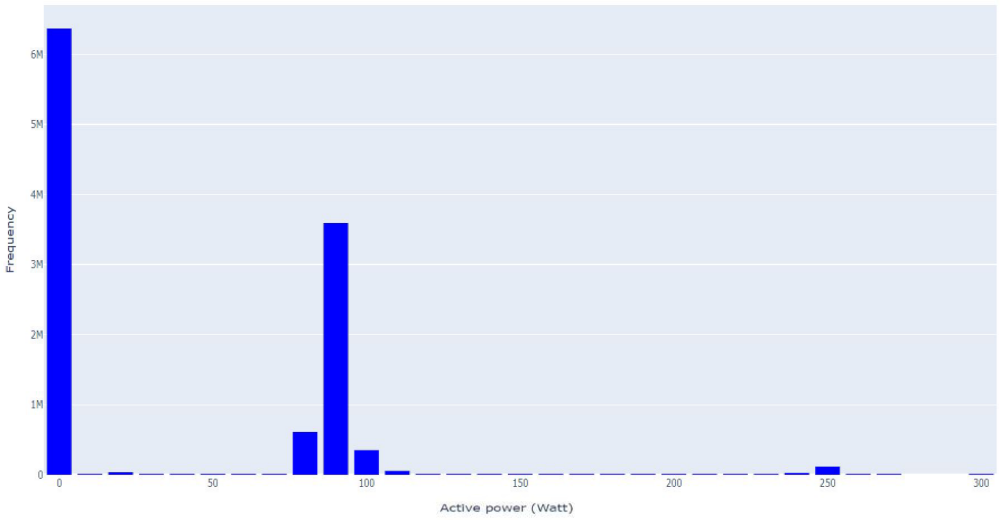

Figure 4.2: Fridge's distribution of occurrences as a function of frequency (Hz) - UKDALE

In Figure 4.2 it is possible to notice that, except for zero values, the most frequent values are mainly concentrated in the range [75W, 115W], while the maximum registered consumption value is 300W, which represents the cutoff.

For training and evaluating the BERT4NILM model, the authors of [12] set the values of some parameters (cutoff, threshold, min on, min off) which are necessary for the computation of the operational status of the appliance. Usually, these settings depends on the type of the appliance, the energy class it belongs to and the residents' consumption habits. Therefore, an adequate domain expertise can be useful to set them correctly.
A brief reminder of the role of these parameters:

- The cutoff is the maximum active power value that an appliance can consume.
- The threshold is the lower limit for the device to be considered turned on.
- The min on represents the minimum duration the appliance must remain on.

- The min off is the minimum duration the device must stay off.

For the real dataset UKDALE, the following values were set: cutoff: 300, threshold: 50W, min on: 60s, min off: 12s. Thus, consumption data of fridge were truncated at 300W, since all the active power values above this cutoff represent outliers, while all the values below the threshold 50W are associated to a fridge off.

SynD

A subset of the active power consumption of the fridge is shown in Figure 4.3, which displays its individual usage pattern, specifically for the day 23rd November 2019. By comparing this consumption pattern with the one sampled from UK-DALE,

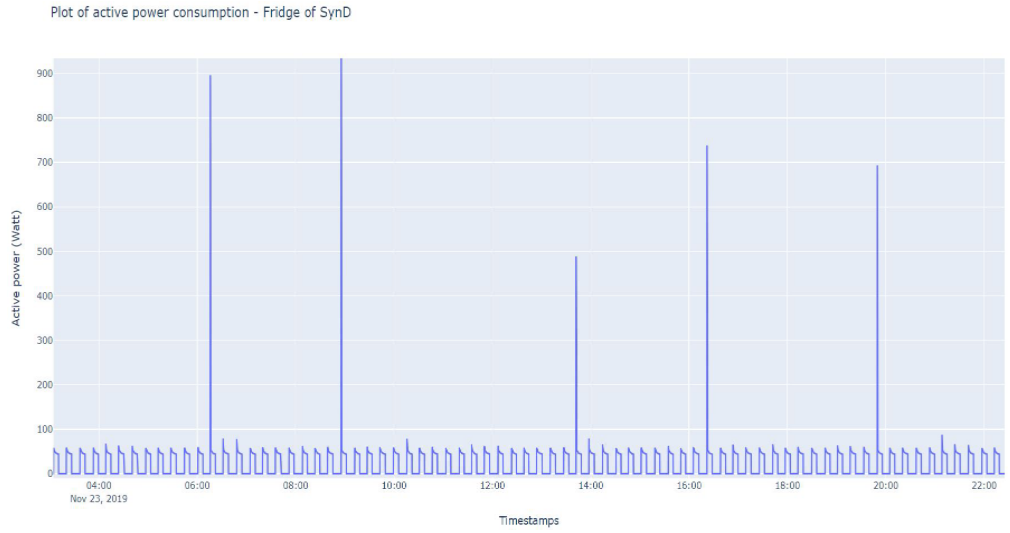

Figure 4.3: Active power consumption of Fridge - SynD - 23rd November 2019

some similarities come out: both of them are characterized by a cyclical behavior, with periodical peaks. However, the synthetic usage patterns seem to have a more regular conduct, which recurs itself with higher peaks of consumption, in fact, the maximum registered active power value is 900W, as it emerges from the distribution of frequency in Figure 4.4.

Usually a traditional domestic fridge reaches a maximum power consumption between the values 100W and 300W, even if the values it can reach often depend on the characteristic of the device, its energetic class, the year of production. Since in the synthetic dataset the data is distributed on a lower scale of values, we decided to reduce the values of the cutoff and the threshold parameter and assign them 100W and 40W, respectively. This choice is mainly supported by the fact that in SynD dataset the most frequent consumption data are concentrated in the interval

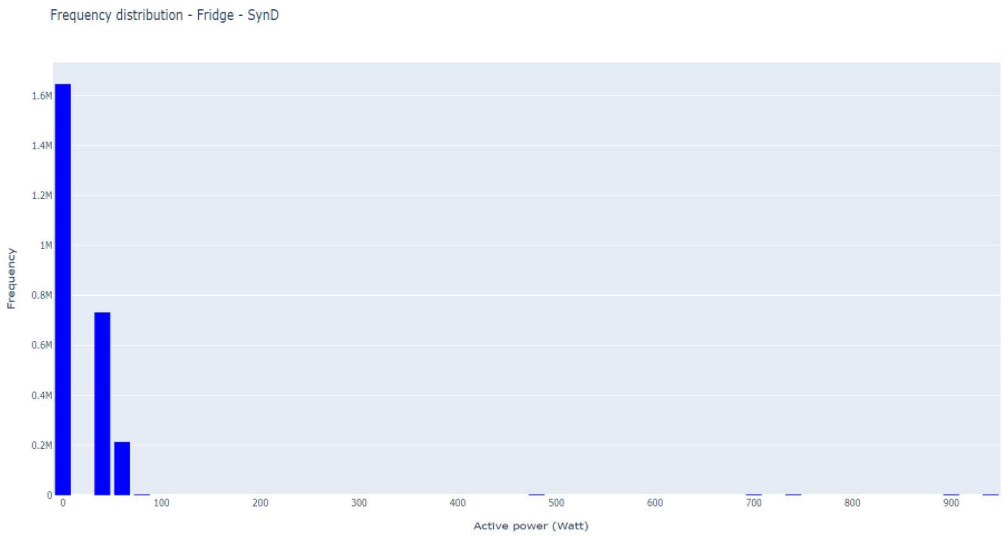

Figure 4.4: Fridge's distribution of occurrences as a function of frequency (Hz) - SynD

[35W, 90W], without recording any values near 300W. While, the min on and the min off parameters remain unchanged: 60s and 12s, respectively.

The parameters settings for both UK-DALE and SynD, real and synthetic datasets, respectively, are summarized in Table 4.1:

Dataset	**Cutoff**	**Threshold**	**Min on**	**Min off**	λ
UKDALE	300	50W	60s	12s	10^{-6}
SynD	100	40W	60s	12s	10^{-6}

Table 4.1: Parameters setting for Fridge appliance in UKDALE and in SynD datasets

The model's hyperparameters, used for all the experiments of this work, are reported in Table 4.2.

Specifically for the analysis of the BERT model performances, three different scenarios were considered:

- the model was trained on the 80% of the SynD dataset and it was tested on the remaining 20%.

Parameter	Value
Input length	480
Masking portion p	0.25
Number of transformer layers l	2
Number of attention heads h	2
Maximum hidden size	256
Convolution kernel size	5
Convolution padding length	2
Deconvolution kernel size	4
Deconvolution stride	2
Deconvolution padding length	1
$L2$ norm pooling kernel size	2
$L2$ norm pooling stride	2
Dropout rate	0.1
Learning rate	10^{-4}
Optimization method	Adam
Betas	0.9, 0.999
Weight decay	0

Table 4.2: BERT4NILM model's parameters settings

- the model was trained and validated on the 80% and 20% of the synthetic dataset,respectively; then, it was tested on all five houses of the UK-DALE dataset.
- the model was trained on Houses 1,3,4,5 and validated on House 1; then, it was tested on House 2.

In all the experiments, the BERT4NILM model was trained for 30 epochs.

4.2.2 Analysis

Table 4.3 shows the BERT model performances for the fridge in all the three considered scenarios.

Model	Train data	Test data	Acc.	Prec.	Recall	F1	MRE	MAE
BERT	SynD	SynD	0.95	0.99	0.88	0.93	0.06	2.87
BERT	SynD	UKDALE	0.64	0.54	0.69	0.59	0.46	39.05
BERT	UKDALE	UKDALE	0.81	0.81	0.73	0.77	0.73	25.49

Table 4.3: Model performances for Fridge

Focusing on the first scenario, in which the model was trained and tested on different portion of SynD dataset, BERT4NILM architecture performs well and reaches very high performances both in the classification and in the regression tasks. The low value of the MRE (0.06) suggests the effectiveness of the model to make accurate predictions when evaluated on a new set of the synthetic dataset, never seen during the training phase. The BERT model also achieves high score in F1 score, meaning that it is able to correctly classify most of positive class ('On' operational status). The results are graphically illustrated in Figure 4.5, which shows the model's ability to fairly reproduce quite well most of the fridge's usage pattern, despite it does not accurately reconstruct the high peaks in consumption.

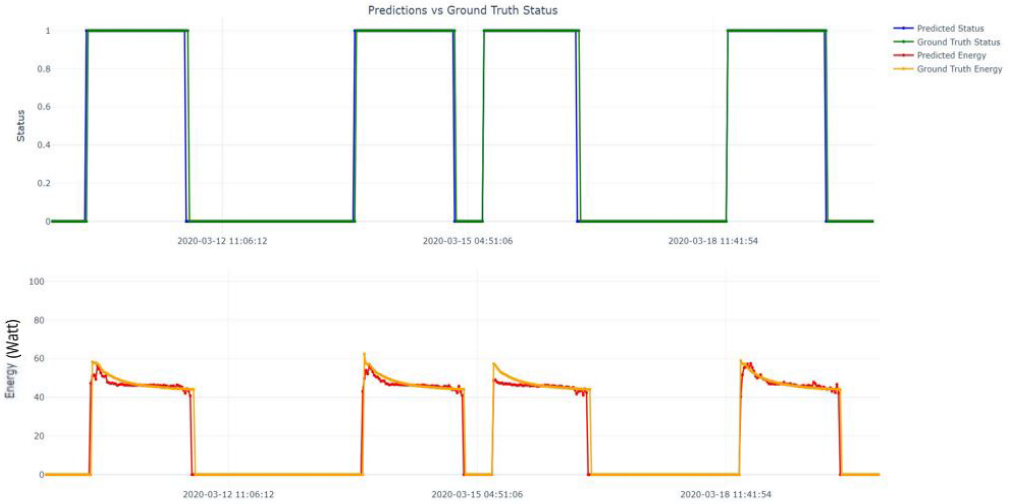

Figure 4.5: Predictions vs ground truth on Fridge - First scenario (train on SynD and test on SynD)

Exploring the second scenario, when the model is trained on SynD dataset and then evaluated on the real one, there is a fairly evident drop in performances. Concerning the classification task, both accuracy and F1 score fall to 0.64 and 0.59, respectively, meaning a reduction of the model's ability to generalize on real-world data. The performances register a precision of 0.54, reflecting that the model does not succeed in accurately classify true positives, with a recall of 0.69, which indicates that, despite an high number of false positives ('On' status is detected when an 'Off' operational status is registered in the real scenario), the model is able to identify most of positive events. Furthermore, compared to the first scenario, the higher MRE and MAE values indicate the difficulty in correctly recognize and reconstruct the usage pattern of the refrigerator, as it is shown in Figure 4.6.

The third scenario involves the training and the evaluation of the BERT4NILM model's performances on the UK-DALE dataset; in Table 4.3 this experiment

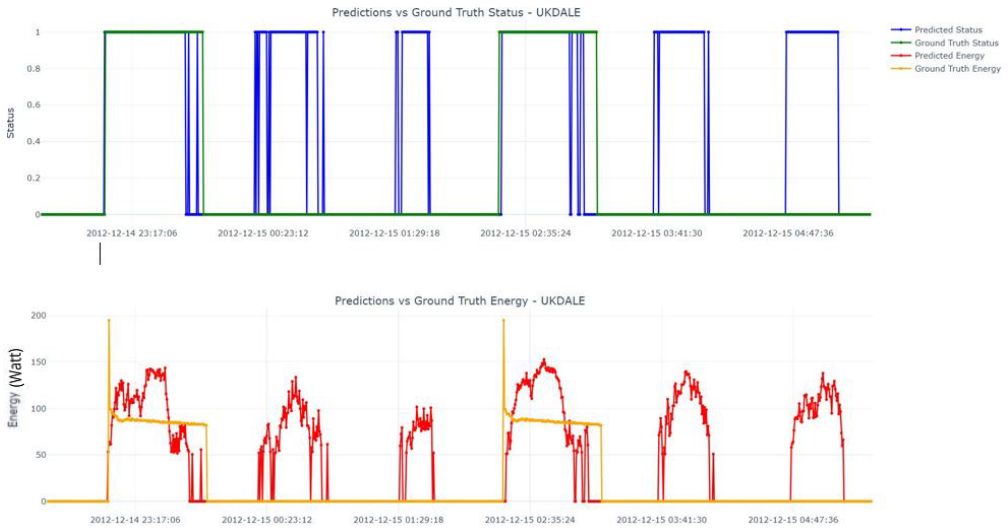

Figure 4.6: Predictions vs ground truth on Fridge - Second scenario (train on SynD and test on UK-DALE - House 1)

registers better results than the previous scenario, with an accuracy of 0.81 and a F1 score of 0.77. However, the MRE and MAE register high values, meaning that the model still faces difficulty in accurately predicts the individual power signal; moreover, it also wrongly identifies a usage power signal even during the inactivity period of the refrigerator, consequently registering a very high number of false positives. Comparing these results with the ones of the previous two experiments, we notice that training and evaluating the BERT4NILM model on a synthetic dataset lead to achieve better performances, both in classification and regression tasks. Specifically in our case, if we compare the synthetic and the real fridge's consumption patterns emerges that in the first one the active power values are mainly concentrated on a lower scale around 60W, while in the realistic pattern around 100-200W, making the generalization of the model more difficult.
The drop of performances on UK-DALE dataset could be caused by the differences between the synthetic and the real datasets: the synthetic data may not reflect the complexity and the variability of real energy consumptions, which are usually characterized by noisy and irregular patterns. A stable and highly regular synthetic dataset can make it challenging to transfer knowledge to real-world one, leading to the model's poor generalization performance.

4.3 Experiments on washing machine appliance

4.3.1 Dataset exploration and parameters settings

UKDALE

Figure 4.7 shows a subset of a real active power consumption of the washing machine of the UKDALE dataset, specifically for the consumption during the day 21 January 2013, in House 1. Typically, the energy consumption of the washing machine depends on multiple factors, such as the type, the model, the energy class, the specific wash cycle selected. It is possible to notice that, at the beginning of the wash cycle, very high active power values are registered until the consumption becomes quite stable. However, multiple peaks often characterize the usage pattern of this household appliance, according to the different phases of the wash cycle. In fact, the most energy intensive process is typically executed at the beginning, especially during the hot wash cycles, in which the washing machine needs to heat the water to the desired temperature, by causing high peaks of consumption. Once the water reaches the required temperature, the pattern becomes more stable, although peaks in consumption could occur, for example when the washing machine requires or absorbs water for its activity.

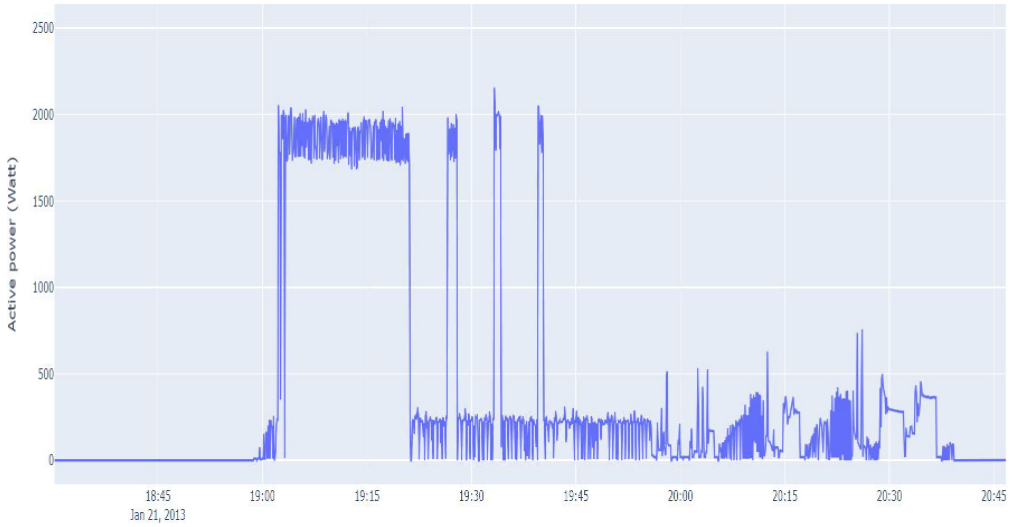

Figure 4.7: Active power consumption of Washing machine - UKDALE - 21th January 2013

From the distribution of frequency of the washing machine, in Figure 4.8, it is observed that the most frequent active power data, in quite stable condition, register a value between 100W and 300W, while the peaks of consumption are

mainly concentrated in the range [1600W,2000W]. For the UKDALE dataset, the maximum limit was set to 2500W, the minimum threshold to consider the washing machine turned on is 20W, while the min on and the min off durations were set to 1800s and 160s, respectively.

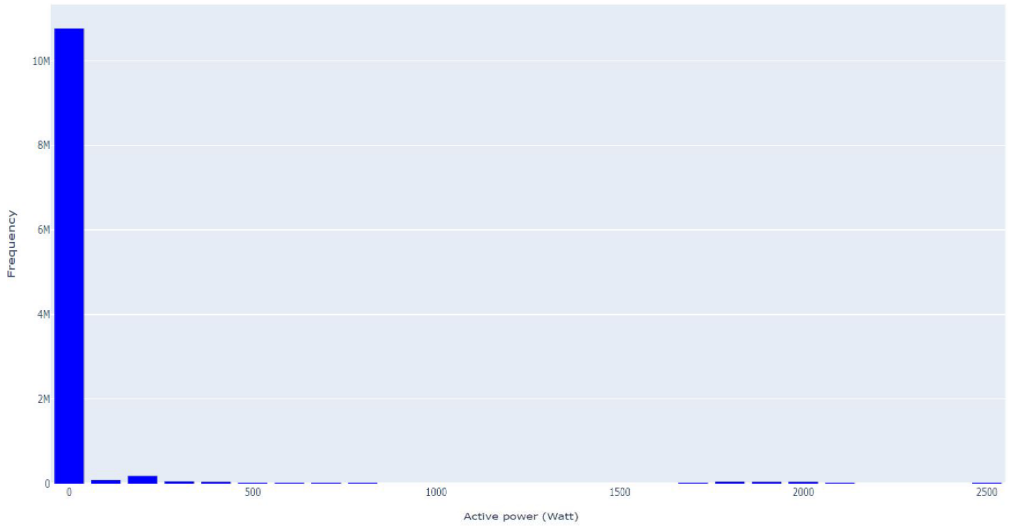

Figure 4.8: Washing machine's distribution of occurrences as a function of frequency (Hz) - UKDALE

SynD

Concerning the synthetic dataset, in Figure 4.9 the usage pattern of the washing machine, during the day 23rd November 2019, is displayed. It is possible to notice that its consumption shape follows the one in the real dataset quite similarly.

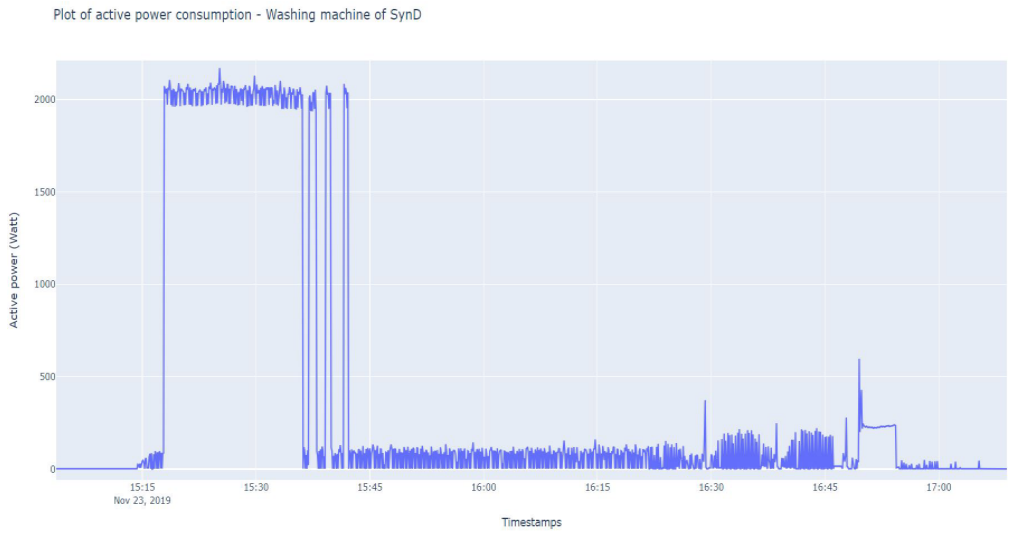

Figure 4.9: Active power consumption of Washing machine - SynD - 23rd November 2019

The similarity in the values of consumption is also proved by the frequency distribution shown in Figure 4.10.

Thus, also for the experiments on the washing machine of the synthetic dataset, the parameters used in UKDALE are kept unchanged. The parameters settings are summarized in Table 4.4.

Dataset	Cutoff	Threshold	Min on	Min off	λ
UKDALE	2500	20W	1800s	160s	10^{-2}
SynD	2500	20W	1800s	160s	10^{-2}

Table 4.4: Parameters setting for Washing machine appliance in UKDALE and in SynD datasets

4.3.2 Analysis

Table 4.5 shows the BERT model performances for the washing machine in all the three considered scenarios.

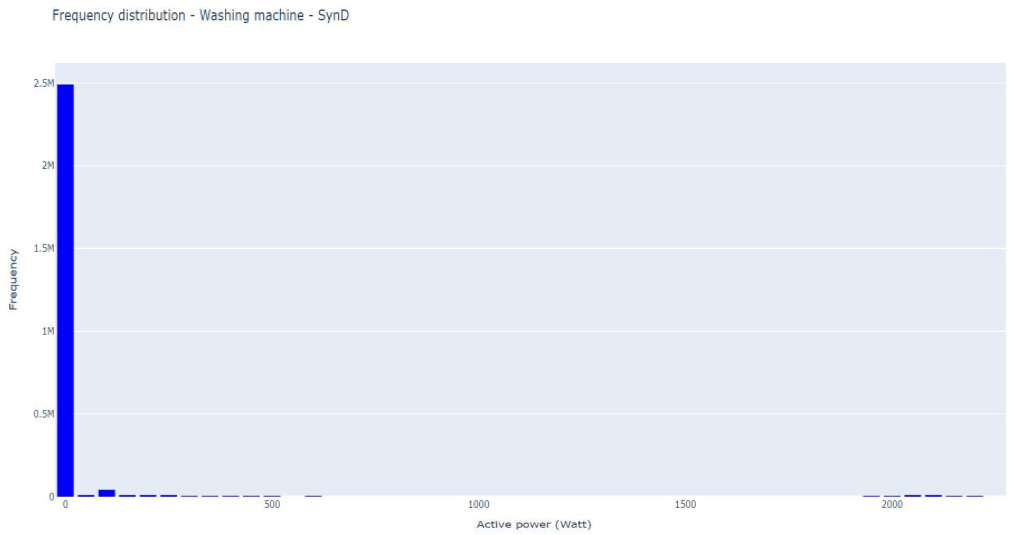

Figure 4.10: Washing machine's distribution of occurrences as a function of frequency (Hz) - SynD

Model	Train data	Test data	Acc.	Prec.	Recall	F1	MRE	MAE
BERT	SynD	SynD	0.94	0.70	0.37	0.46	0.08	11.97
BERT	SynD	UKDALE	0.87	0.11	0.15	0.12	0.15	31.01
BERT	UKDALE	UKDALE	0.97	0.40	0.28	0.33	0.04	6.98

Table 4.5: Model performances for Washing machine

Analyzing the numerical results in the first case, the high accuracy (0.94) suggests that the model performs very well on SynD dataset, correctly learning patterns and features from the data. However, the F1 score of 0.46 indicates that, even if the model is quite good on overall status predictions, its performances drop when it deals with imbalanced classes. It seems that the model predicts the majority class ('Off' status) better than the minority one ('On' status), which negatively affects the precision and recall metrics. In terms of error rate, the MRE registers a very low value (0.08), meaning that the model is able to capture relative differences in predictions quite well. However, Figure 4.11 shows that the model does not correctly reconstruct the pattern, leading to register a quite high number of false negative, as the low value of recall indicates. Moreover, the classification of the operational status strictly depends on the value of the threshold, thus if the predicted value is quite different from the true one, consequently the model will wrongly classified the relative status. Another aspect that should be take into account during the analysis of results is that BERT4NILM model assigns the status to input data considering the threshold, min and min off durations, while to assign

the operational status on top of the predicted data only the threshold value is considered.

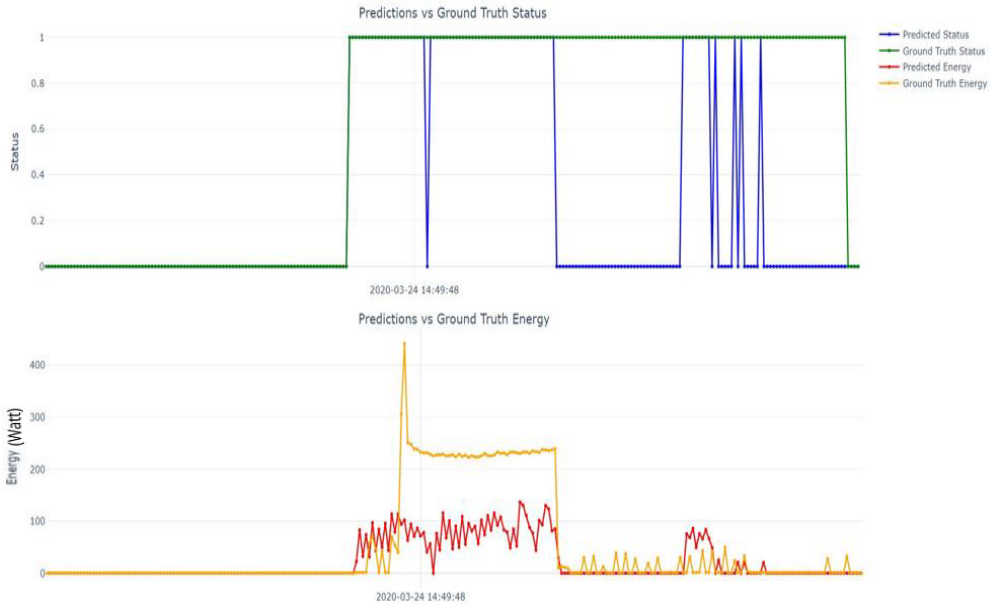

Figure 4.11: Predictions vs ground truth on Washing machine - First scenario (train on SynD and test on SynD)

The second scenario in Table 4.5 shows a significantly drop of performances, when the model is evaluated on the real dataset, especially in terms of precision (0.11), recall (0.15) and F1 score (0.12). This means that the model struggles to generalize on UKDALE, maybe due to a consistent class imbalance, in predicting minority classes. Unlike the fridge, a constantly on device, the washing machine presents the most of time a 'Off' operational status and washing programs usually last for few hours. This situation seems to be emphasized when we move from the synthetic dataset to the real one, in which the ground truth consumption patterns contain mostly zero values, as shown in Figure 4.12. When the model is trained on SynD and then tested on UK-DALE, it tends to predict operational patterns even when the device is actually turned off, leading to an incorrect classification of the washing machine in the 'On' state, generating a large amount of false positives. Consequently, a negative impact on the precision and recall metrics is registered. Since the precision measures the proportion of true positive predictions (correctly predicted "On" states) out of all predicted "On" status (true positives + false positives), the higher is the number of false positives, the lower is the value of this metric. Moreover, a low recall indicates the missing of a big portion of actual 'On' status by the model. Both metrics are negatively influenced by the model's inability to accurately distinguish between the "On" and "Off" operational states of

the device, particularly in the real-world dataset where "Off" states dominate.

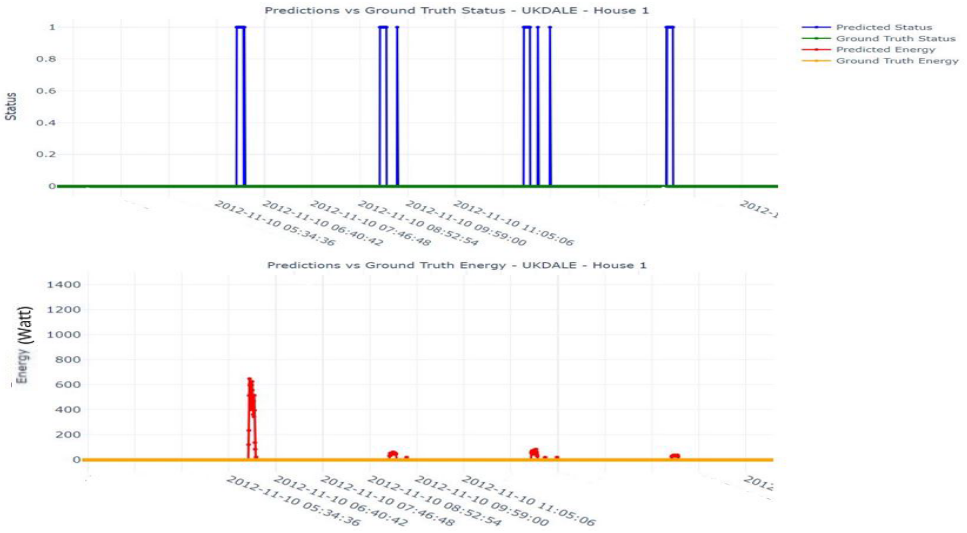

Figure 4.12: Predictions vs ground truth on Washing machine - Second scenario (train on SynD and test on UK-DALE - House 1)

Analyzing the regression task's performances, the MRE (0.15) normalizes the error by the true value, so if the actual values are small, even large absolute errors may still result in a relatively small MRE; the high value of MAE (31.01) is negatively impacted by the overestimation of the consumption when the device is 'Off', leading to large errors in absolute terms.

Finally, the third experiment on the washing machine is quite comparable with the results of the first case: the BERT model is quite good at reconstructing most of the individual consumption pattern, proving by small MRE and MAE metrics; however, despite the very high accuracy, the F1 score (0.46 in the first case and 0.33 in the second one) indicates that the model is not so good at correctly classifying 'On' operational status, even if it is trained and evaluated on a different portion of the same dataset. This limitation in the classification task could be influenced by the different way the status is assigned to the input values and the predicted ones: for input data the function consider the threshold and the minimum durations to consider the device on or off, while for the predicted consumptions the status is assigned by only considering the threshold, a variable parameter that can differ from the type of device. Moreover, the errors in the reconstruction of consumption patterns also lead the model to register many false positives. Overall, for washing machine the model is not able to generalize well maybe because the scale of active power values in the synthetic and in the real-world dataset are quite different: in SynD the values are mostly concentrated below 400 W, while in UKDALE they frequently reach also 2000W.

4.4 Experiments on dishwasher appliance

4.4.1 Dataset exploration and parameters settings

UKDALE

In Figure 4.13 it is possible to analyze the realistic usage pattern of the dishwasher, during an interval that covers the end of the day 21th January 2013 and the beginning of the immediately next day. Similarly to the washing machine, also the consumption of the dishwasher is influenced by different aspects, like the model type, the type of the wash cycle chosen, the energetic class it belongs to. The power values related to the peaks of consumption fall in the range [2000W,2500W]; these peaks are mainly registered during the most energy intensive activities of the dishwasher, like intensive cycles that require high water temperatures and long wash times, or heated dry activity.

Figure 4.13: Active power consumption of Dishwasher - UKDALE - 21th/22th January 2013

The different range of values of dishwasher's consumption data is also shown in Figure 4.14, which shows how the values' frequency is distributed.

Thus, the cutoff was set to 2500W, which represents the maximum value the dishwasher can reach in real scenarios, the minimum on-threshold was set to 10W; while, both the min on and min off durations are equal to 1800s.

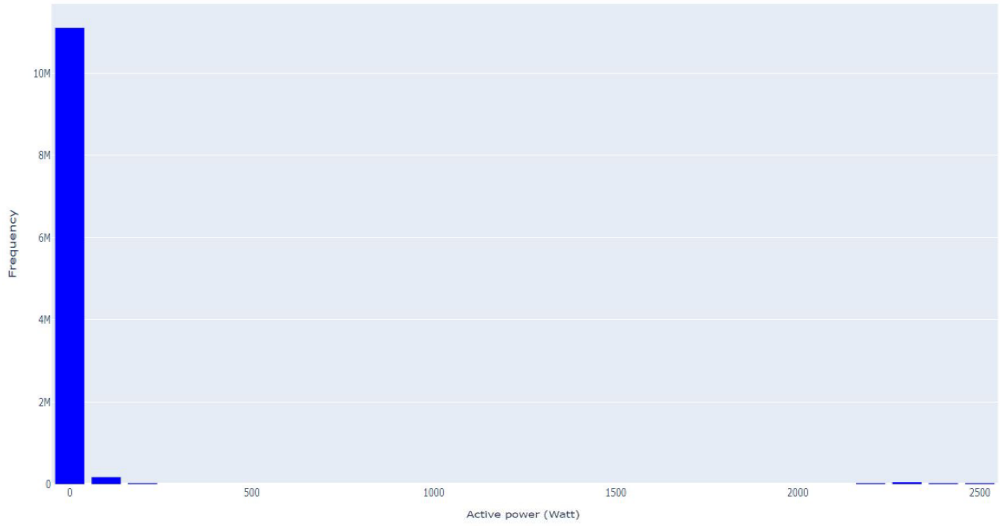

Figure 4.14: Dishwasher's distribution of occurrences as a function of frequency (Hz) - UKDALE

SynD

Concerning the synthetically generated dishwasher's usage pattern, its data distribution is analyzed in Figure 4.15 and Figure 4.16.

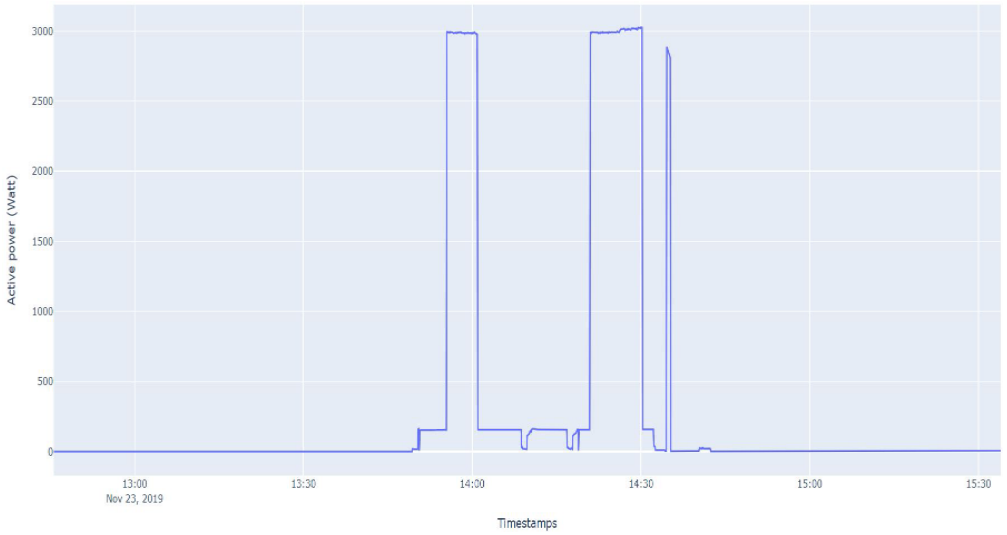

Figure 4.15: Active power consumption of Dishwasher - SynD - 23rd November 2019

The first one exhibits a sample of its consumption, specifically related to the

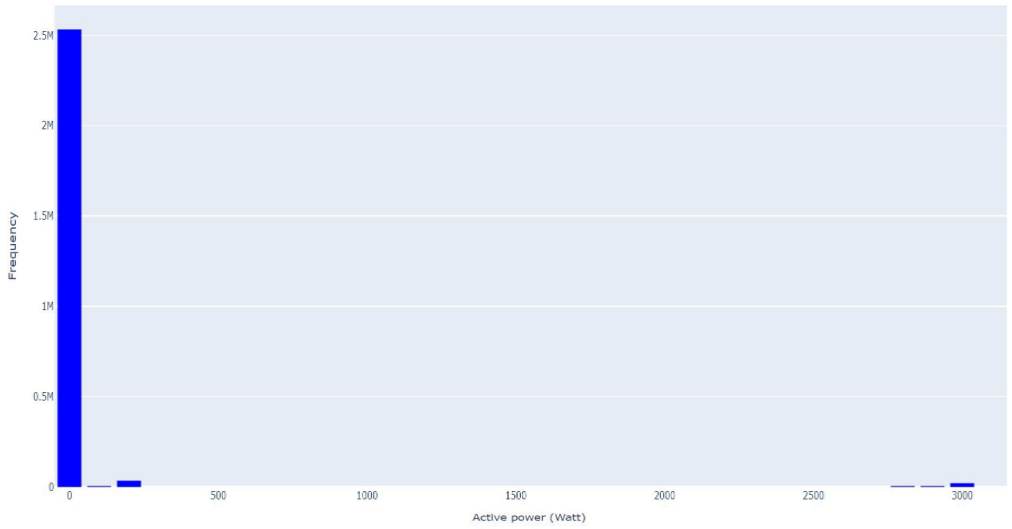

Figure 4.16: Dishwasher's distribution of occurrences as a function of frequency (Hz) - SynD

day 23rd November 2019, while the second one shows the frequency distribution of its values. In the SynD dataset the duration of the dishwasher's activity is usually shorter than the one in UKDALE dataset; moreover, the peaks of consumption in Figure 4.15 appear more regular with higher values around 3000W, which represent anomalies. So, to align the experiments' settings to the ones chosen for the realistic dataset, the same parameters were chosen since the consumption data follows a quite similar distribution in both contexts.

The corresponding parameters settings are summarized in Table 4.7.

Dataset	Cutoff	Threshold	Min on	Min off	λ
UKDALE	2500	10W	1800s	1800s	1
SynD	2500	10W	1800s	1800s	1

Table 4.6: Parameters setting for Dishwasher appliance in UKDALE and in SynD datasets

4.4.2 Analysis

Table 4.7 shows the BERT model performances for the dishwasher in all the three considered scenarios.

Model	Train data	Test data	Acc.	Prec.	Recall	F1	MRE	MAE
BERT	SynD	SynD	0.99	0.72	0.79	0.72	0.02	3.45
BERT	SynD	UKDALE	0.96	0.10	0.08	0.09	0.04	29.47
BERT	UKDALE	UKDALE	0.96	0.75	0.60	0.67	0.05	16.18

Table 4.7: Model performances for Dishwasher

The experiments results of the first scenario, reported in Table 4.7, show an accuracy very close to 1, indicating that the model performs extremely well on the synthetic training data and it is able to correctly recognize the dishwasher's pattern. Also the value of F1 score (0.72) indicates a good overall performance of the model in correctly classifying most of the positive events, with few errors. The classification errors made by the model are mainly attributed to the presence of false negative instances, as shown in Figure 4.17; despite a low difference between the predicted and the real consumption values, the model registers some false negatives maybe due to the fact that, on top of realistic signals, the status is assigned based on threshold and minimum and maximum durations , while for the predicted consumptions it is assigned only considering the threshold. Thus, the initial definition of the threshold acquires importance and could be decisive. Exploring the regression results, the very low value of MRE (0.02) means that the model does not make significant prediction errors with respect to the magnitude of the data (Figure 4.17). These results are quite similar to the ones reported for the experiment of the third scenario, however the performances drop significantly when the model trained on the synthetic dataset is evaluated on UK-DALE, especially in terms of F1 score (0.09). This numerical result indicates that the model does not generalize well, due to the fact that the model misses a large portion of actual positive instances and classifies the dishwasher's operational status with a low

precision. Concerning the regression task, the MRE value is quite high, so the model makes large errors in absolute terms, indicating that is not able to capture and distinguish the relevant aspects of the real-world dataset (Figure 4.18). The results of the third scenarios could be mainly explained by the incapacity of the model to learn and distinguish the complexities and the variations in the real consumption patterns. The model seems to overfit when its is trained and evaluated on a different portions of the same dataset; thus, when it is trained on SynD and then evaluated on noisy and real data of UK-DALE, the model does not generalize well.

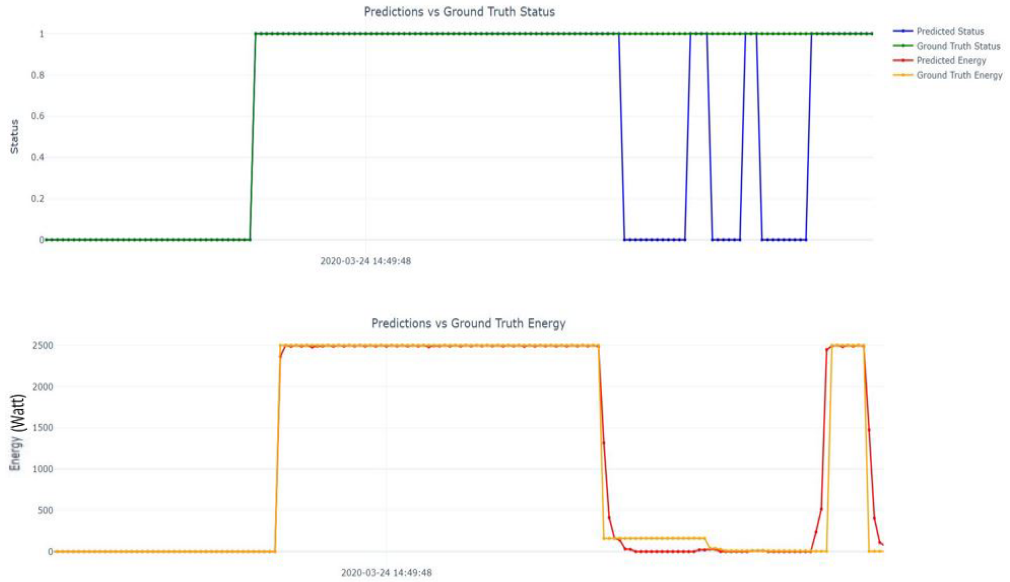

Figure 4.17: Predictions vs ground truth on Dishwasher - First scenario (train on SynD and test on SynD)

4.5 Experiments on microwave appliance

4.5.1 Dataset exploration and parameters settings

UKDALE

The active power consumption's intensity of a domestic microwave depends on the type of use, on its characteristics and on the energetic class it belongs to. For example, if the microwave is set to the lower power level, the amount of active power consumed is low, however this setting requires a longer device's usage time. Figure 4.19 shows a small part of the usage pattern of a microwave in the UK-DALE dataset, specifically for the day 20th January 2013 in House 1.

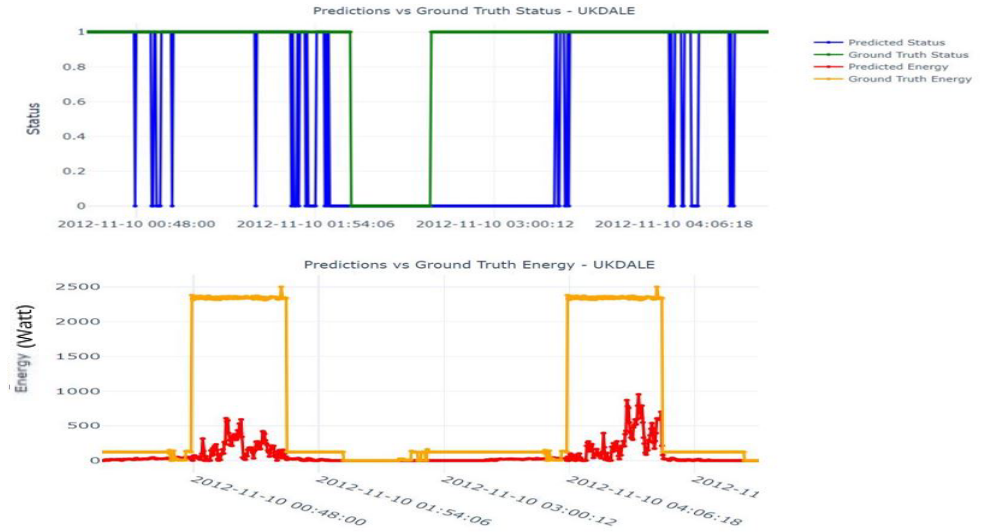

Figure 4.18: Predictions vs ground truth on Dishwasher - Second scenario (train on SynD and test on UK-DALE - House 1)

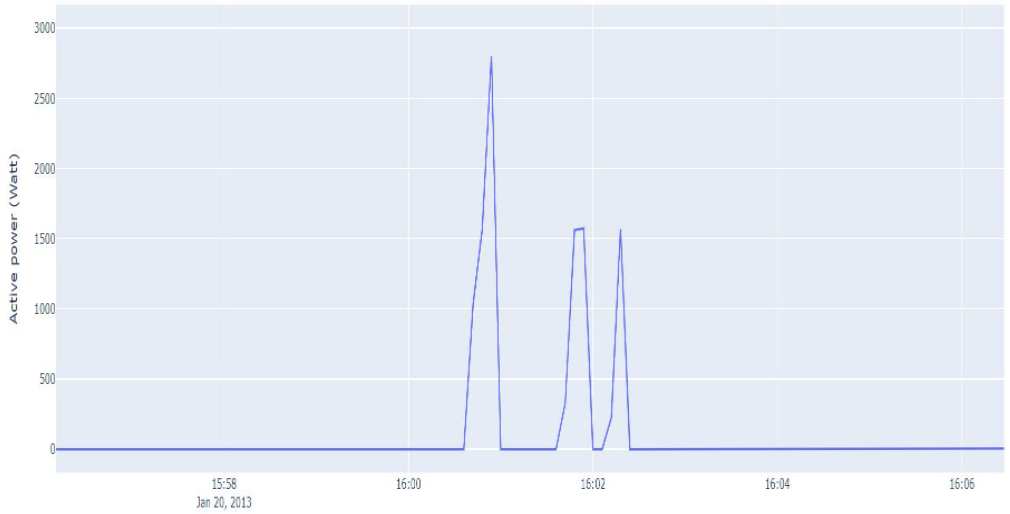

Figure 4.19: Active power consumption of Microwave - UKDALE - 20th January 2013

Unlike other appliances, like dishwasher, fridge or washing machine, the power consumptions of a microwave typically last for few seconds or minutes, with peaks of consumption in the range [1500W, 3000W], as shown in Figure 4.20.

In fact, the authors in [12] set the cutoff to 3000W and the on-threshold equals to 200W; while, the min on and the min off durations were set to 12s and 30s,

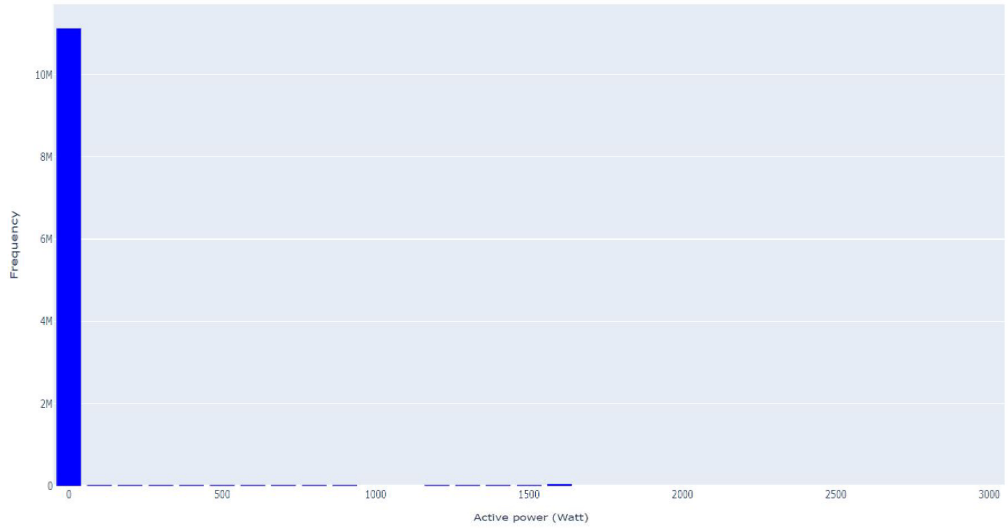

Figure 4.20: Microwave's distribution of occurrences as a function of frequency (Hz) - UKDALE

respectively.

SynD

Figures 4.21 and 4.22 exhibit the synthetic microwave's individual consumption pattern during the day 22th November 2019 and the frequency distribution of the microwave in SynD dataset, respectively. By comparing these plots with the ones concern the UK-DALE dataset, it is possible to notice that the synthetic active power values of microwave's consumption are distributed on a very different scale: the most of values are below 200W while the maximum active power value reaches 1300W.

Thus, the choice of the cutoff and threshold parameters for the experiments on the synthetic dataset must be adapted by decreasing the values to 1500W and 100W, respectively.

Table 4.8 summarizes the parameters set for the experiments on UK-DALE and SynD datasets.

Dataset	Cutoff	Threshold	Min on	Min off
UKDALE	3000	200W	12s	30s
SynD	1500	100W	12s	30s

Table 4.8: Parameters setting for Microwave appliance in UKDALE and in SynD datasets

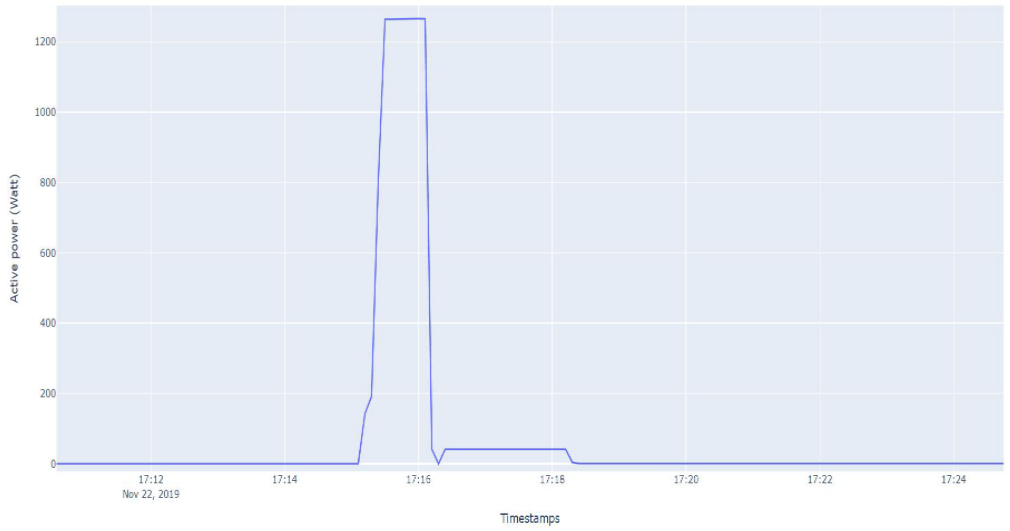

Figure 4.21: Active power consumption of Microwave - SynD - 22th November 2019

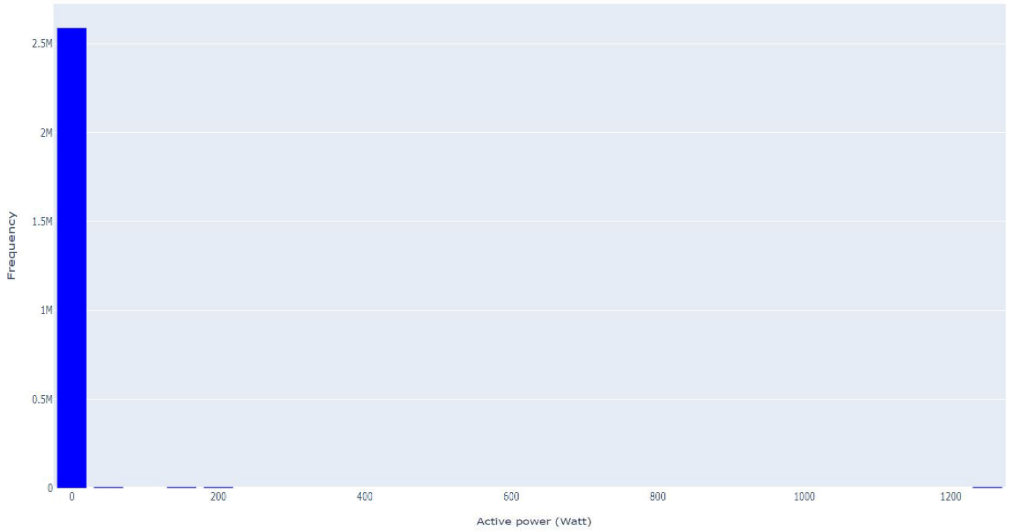

Figure 4.22: Microwave's distribution of occurrences as a function of frequency (Hz) - SynD

4.5.2 Analysis

Table 4.9 shows the BERT model performances for the microwave in all the three considered scenarios.

In the first scenario, shown in Figure 4.23, the model seems to perform perfectly on the training and the testing data, belonging to the same synthetic dataset, since

Model	Train data	Test data	Acc.	Prec.	Recall	F1	MRE	MAE
BERT	SynD	SynD	1.00	0.02	0.02	0.06	0.00	7.50
BERT	SynD	UKDALE	0.99	0.04	0.03	0.03	0.07	9.88
BERT	UKDALE	UKDALE	1.00	0.01	0.01	0.01	0.01	6.57

Table 4.9: Model performances for Microwave

the high accuracy suggests that it correctly classified all instances. Despite this, the very low value of F1 score (0.06) indicates a significant issue with class imbalance or poor handling of minority classes, meaning the model may not effectively recognize less common patterns. Unlike the patterns of more frequently used devices, such as the fridge, the microwave is one of the most rarely used devices during the day, whose usage pattern typically last few minutes. In our specific case, the model succeeds in correctly classifying both positive and negative instances, as it is shown in the first plot in Figure 4.23; however, due to the class imbalance, the F1 score registers a very low value since the dominant class is associated with the 'Off' status. Despite the low reconstruction errors, the model struggles to effectively reconstruct the individual pattern for the microwave, as illustrated in the second plot of Figure 4.23. The low relative error may be attributed to the model's ability to accurately reconstruct the majority of consumption patterns, which consist largely of zero-values that correspond to inactive periods of the device. However, when the model fails to predict power signals, it results in significantly high errors.

In the second scenario the BERT4NILM model is evaluated on the real-world dataset UKDALE, after being trained on the synthetic one. The obtained results, registered in Table 4.9, are displayed in Figure 4.24. Like the previous case, the extremely high accuracy (0.99) indicates that the model correctly classified all instances; however, the low F1 score suggests that, due to class imbalance, the number of correctly classified positive instances ('On' status) is significantly less than the negative class. This result is obtained also due to the high number of false positives ('On' operational status while the device is off in the ground truth), given the poor generalization capability of the model. In fact, due to significant differences in shape and characteristics of the synthetic and the real consumption patterns, the model struggles in correctly reconstructing the microwave's active power signal of UKDALE: the predicted signal seems to be more regular and periodical with respect to the real one, which is characterized by high and very low peaks of consumptions, alternatively. The relatively high MAE shows that model's predictions are often far from the actual values, emphasizing the model's challenges in adapting to the real-world context, as it is shown in Figure 4.24.

Finally, when the model is trained and evaluated on the real-world dataset, a similar result to the first scenario is obtained: the high value of accuracy indicates

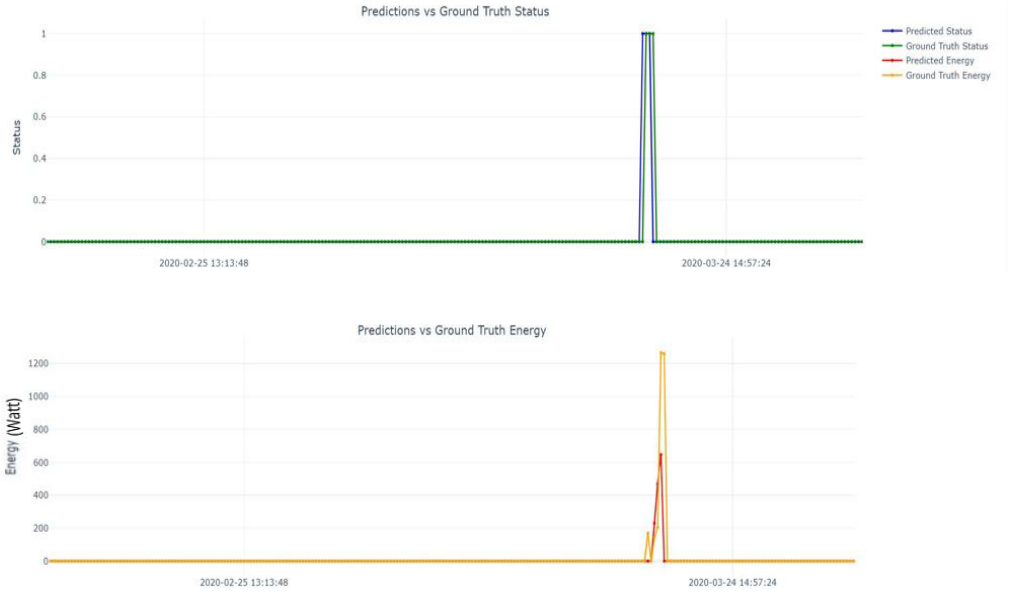

Figure 4.23: Predictions vs ground truth on Microwave - First scenario (train on SynD and test on SynD)

Figure 4.24: Predictions vs ground truth on Microwave - Second scenario (train on SynD and test on UK-DALE - House 1)

that the model correctly classifies all instances, however, the low F1 score could be mainly attributed to the large predominance of negative classes in the original dataset. The MAE is lower than in the second scenario, indicating that while

predictions are still accurate on the training and testing data, the model may be overfitting to the specific characteristics of the UKDALE dataset.

4.6 Experiments on kettle appliances

4.6.1 Dataset exploration and parameters settings

UKDALE

As it is shown in Figure 4.25, the active power consumption of a domestic kettle mainly depends on the amount of the energy it uses to heat water, which leads to a pattern quite stable, without high peaks of consumption. Like the microwave, also the activity of this appliance typically lasts for few seconds or minutes. Figure 4.25 concerns a subset of a real kettle's consumption pattern during the day 20th January 2013 in House 1.

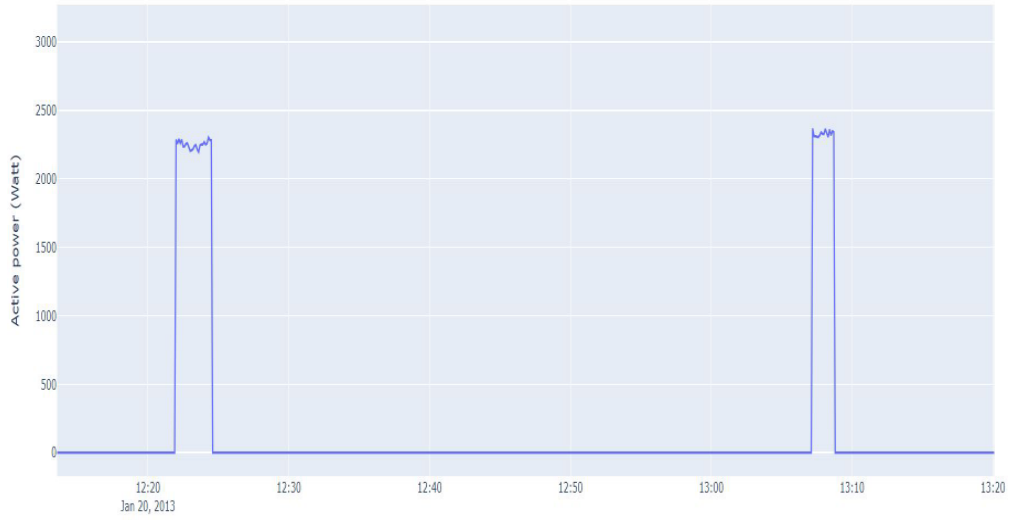

Figure 4.25: Active power consumption of Kettle - UKDALE - 20th January 2013

Figure 4.26 exhibits the frequency distribution of its consumption values, which are mainly concentrated in the range [2000W, 3000W]; in fact, most of domestic kettles record a general power consumption comprises between 1500W and 3000W.

Thus, the cutoff values was set to 3100W, while the minimum threshold to consider the kettle turned on was set to 2000W; moreover, the min on and min off durations were set equal to 12s and 0s, respectively.

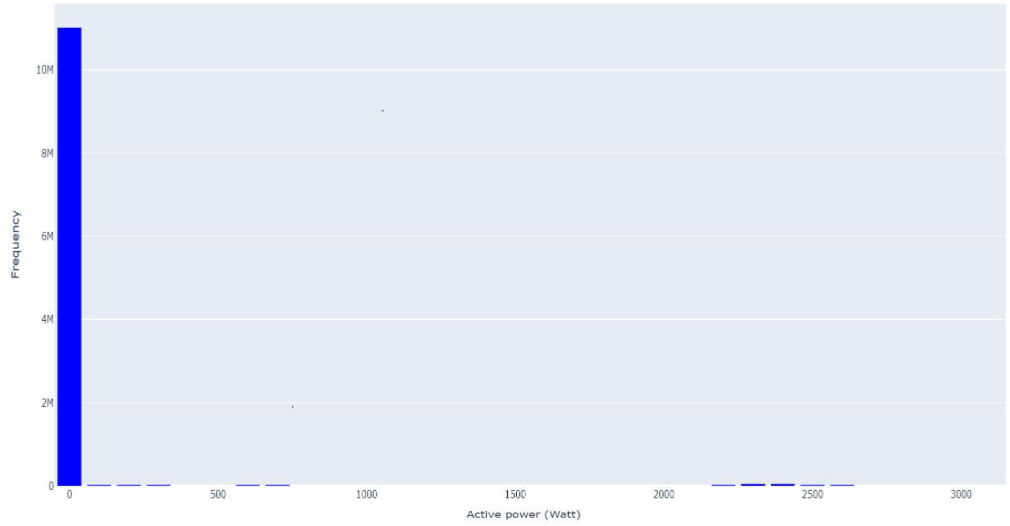

Figure 4.26: Kettle's distribution of occurrences as a function of frequency (Hz) - UKDALE

SynD

By comparing the synthetic kettle's usage pattern in Figure 4.27, about the consumption of the day 22th November 2019, with the one in UK-DALE dataset, it is possible to see that the range of synthetic active power values are distributed on a smaller scale, around the value 1800W. This can also be observed in Figure 4.28 that represents the kettle's frequency distribution in SynD dataset.

Thus, since the real and the synthetic datasets present a different distribution of consumption values for the kettle, the threshold for the experiments on SynD dataset must be changed; so, its value has been lowered to 1800W. While, the other parameters remain unchanged.

Moreover, from Figure 4.28 emerges that the synthetically generated kettle's usage pattern, synthetically generated, presents very few values of the device during its activity and all of them are above 1800W.

Table 4.10 collects the parameters settings for the experiments on the kettle device, both for the UK-DALE and the SynD datasets.

Dataset	**Cutoff**	**Threshold**	**Min on**	**Min off**
UKDALE	3100	2000W	12s	0s
SynD	3100	1800W	12s	0s

Table 4.10: Parameters setting for Kettle appliance in UKDALE and in SynD datasets

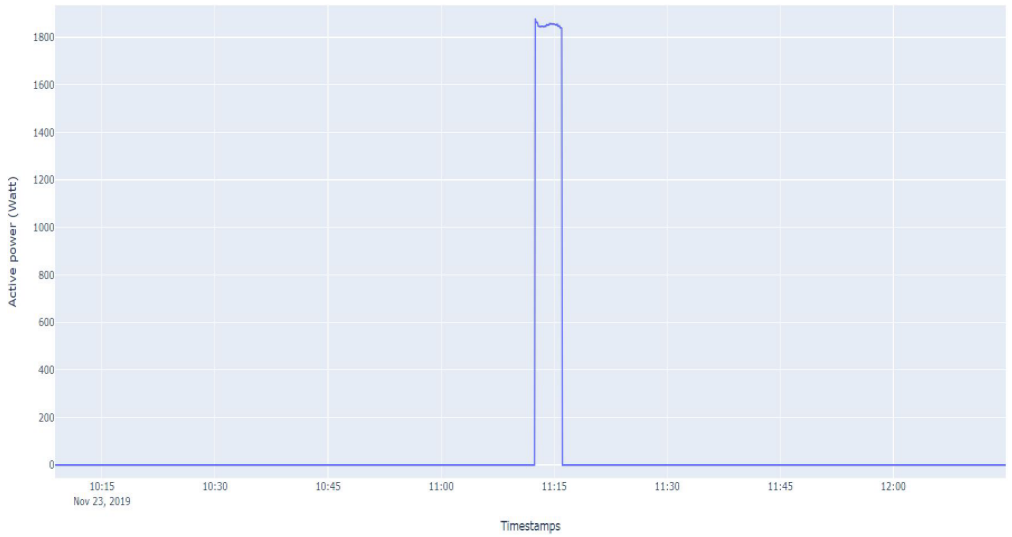

Figure 4.27: Active power consumption of Kettle - SynD - 22th November 2019

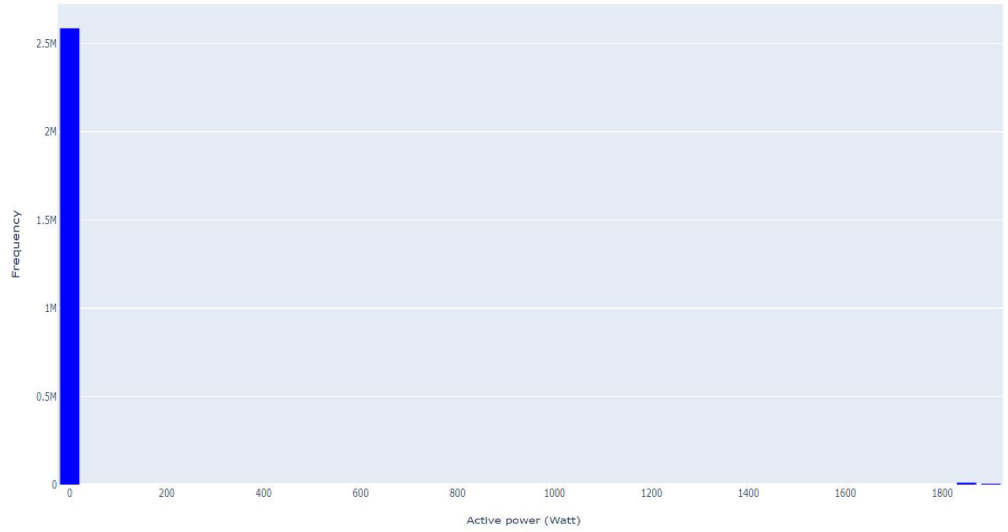

Figure 4.28: Kettle's distribution of occurrences as a function of frequency (Hz) - SynD

4.6.2 Analysis

Table 4.11 shows the BERT model performances for the microwave in all the three considered scenarios.

In the first scenario (Figure 4.29), the high value of accuracy means that the model correctly classifies all the instances in the synthetic portion of the test data. However, limiting the analysis to the accuracy could be misleading especially

Model	Train data	Test data	Acc.	Prec.	Recall	F1	MRE	MAE
BERT	SynD	SynD	1.00	0.24	0.24	0.24	0.0002	0.88
BERT	SynD	UKDALE	0.99	0.20	0.05	0.08	0.006	14.66
BERT	UKDALE	UKDALE	1.00	0.95	0.87	0.91	0.002	6.82

Table 4.11: Model performances for Kettle

with an imbalanced dataset, like in this specific case in which the number of 'Off' operational status predominates on the positive instances. Precision and recall are both equal to 0.24, exactly the same low value of the F1 score, which is their harmonic mean. These results suggest that, even if the model reaches a high overall accuracy, it struggles to classify the minority class, maybe due to the class imbalance, indicating a potential overfitting on the majority class. Analyzing the regression task, a very low value of MRE indicates that, on average, the BERT model makes almost no relative errors, as shown in Figure 4.29; this result could also be attributed to the fact the it consistently predicts the dominant zero-values, which collapses the value of this metric. Despite the MAE is relatively low, the low values of recall and precision suggest that the few misclassified instances generate significantly high errors, when the model fails in the reconstruction task. Despite the model accurately predicts the overall shape of the kettle's power consumption signal, as the low reconstruction error shows, a possible reason for which the F1 score collapses could be mainly related to the presence of false positives: since all the ground truth non-zero values are above the classification threshold of 1800W, even small prediction errors around this threshold can lead to misclassification, where values slightly above 1800W are incorrectly classified as "on." So, the model often predicts "on" events, even when the actual power consumption is zero, resulting in low precision.

Also the second experiment of Table 4.11 registers an high accuracy; however, when the model is evaluated on the real-world dataset, its generalization capability seems to fail, as shown by the drastic decline of the F1 score. The first possible cause could be find in the domain shift, which can not allow the model to accurately adapt to the different distribution of the real-world kettle's consumption pattern, which tend to assume higher active power signals. Thus, this indicates a poor generalization ability of the model on a different dataset. The low MRE (0.006) suggests that, on average, the relative errors remain quite low, maybe due to the model's tendency to predict the dominant class (zero-values), leading to few errors in relative terms. This is shown in Figure 4.30 in which the model wrongly predicts the consumption pattern of the microwave with null values, leading to a collapse of MRE since the actual value is quite high. Moreover, this also justifies the low value of F1 score, given a quite high number of false negatives. While, unlike the

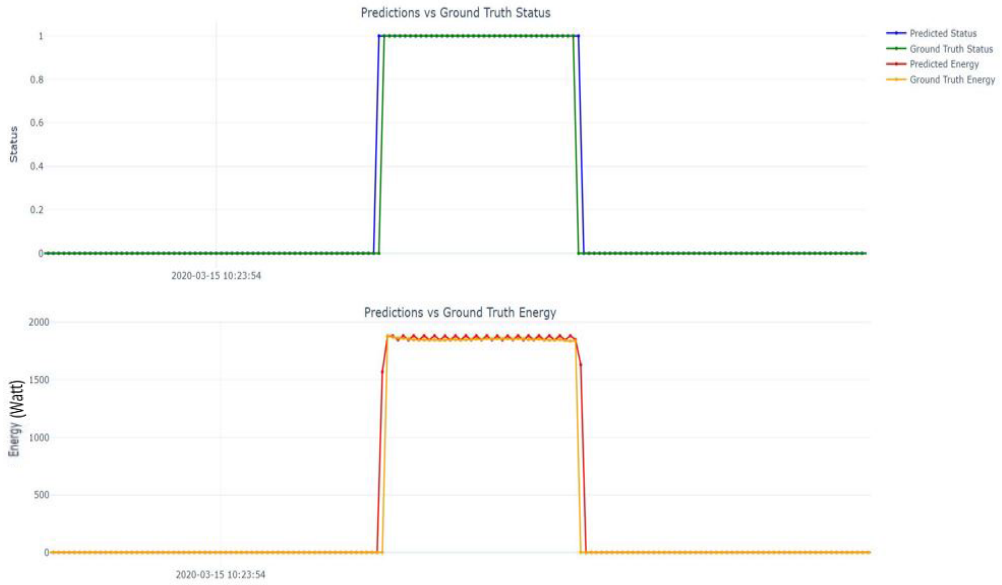

Figure 4.29: Predictions vs ground truth on Kettle - First scenario (train on SynD and test on SynD)

first scenario, the MAE is quite high (14.66) meaning that the magnitude of the absolute errors is large, despite the low MRE; this could means that when the model fails in predicting instances, it makes quite high errors, failing to accurately predict the correct class (Figure 4.30).

In the third scenario, unlike the previous cases, the high accuracy is supported by a higher recall, meaning that the model performs better when it is trained and tested on the same real-world dataset (UKDALE). Indeed, the model succeeds to correctly identify the vast majority of true positives. Moreover, both MRE and MAE register significantly low values, indicating that the model's predictions are far closer to the true values than the other two scenarios. So, the model seems to make accurate predictions when trained and tested on the same real dataset. In the third experiment on UKDALE dataset, the BERT4NILM model seems to significantly improve its generalization capabilities maybe because it has been exposed to the same patterns, noise and features during training which seem to frequently recur in the testing phase, too. However, the BERT4NILM model presents a limitation when it works on rarely used domestic devices, like microwave or kettle, maybe because it is characterized by a too complex architecture, so that it is not capable of reaching good performances on devices with a quite simple consumption pattern.

Figure 4.30: Predictions vs ground truth on Kettle - Second scenario (train on SynD and test on UK-DALE - House 1)

Chapter 5

Conclusions and future works

In conclusion, from the experiments conducted on the BERT4NILM model, considering both the synthetic and the real-world datasets, it turned out that it performs quite well on the synthetic dataset, especially for the frequently used domestic devices, like fridges. It succeeds in accurately reconstructing the individual consumption pattern with low errors, achieving high accuracy and F1 score which means that the model correctly recognizes and classifies most of the 'On' operational status. However, when BERT4NILM is evaluated on a real-world dataset, its overall performances drastically drop, indicating a poor model's generalization capability; it probably overfits on SynD, learning too well the features and the synthetic consumption patterns. This was observed especially for the most rarely used appliances, like microwaves and kettles. One of the possible causes could be the different distribution and features that characterize synthetic and real consumption patterns in residential houses; synthetic datasets, like SynD, are artificially generated and often simplify the complexities of the real-world power signals, leading the model to fail in correctly recognizing devices. A realistic household appliance's consumption pattern usually presents a less stable signal, eventually characterized by noises and missing records. Moreover, another aspect that limits the model to reach a high level of generalization is the choice of the minimum threshold to consider the device active: this value influences the assignment of the operational status, so its value should be accurately chosen according to the type of the device and its energetic class, since different devices of the same category could have different thresholds. From the experiments, it turned out that the distribution of energy consumption values of most of the considered household appliances in the synthetic dataset is quite different from the realistic ones, since they fall into a completely different interval of values. This aspect mainly affects

the model's classification performances: a minimal reconstruction error around the considered threshold can lead to determine false positives or false negatives, causing the collapse of the precision and recall metrics. Therefore, this thesis has mainly focused on investigating and identifying a consistent threshold based on synthetic data distribution and the general consumption behavior of the considered device, in order to align and make the experiments results comparable. Next steps could potentially be pretraining on syntehtic data and subsequently fine-tuning on a subset of real-world data, along with tuning model's hyperparameters, in order to determine the minimum amount of real-world data required for an effective generalization of the model. Finally, the BERT4NILM model seems to be too complex for rarely used devices with a very simple consumption pattern, thus, for this type of devices, a possible future work might be improving and further adapting simpler architecture like LSTMs and CNNs for the specific NILM tasks. Moreover, it also might be useful to integrate transient-state data, recorded at a high frequency, with low frequency data, in order to include more discriminative features that could help the model to correctly recognize and distinguish different household devices.

Bibliography

[1] Iqbal H. Sarker. «Machine Learning: Algorithms, Real-World Applications and Research Directions». In: (2021) (cit. on pp. 5, 7).

[2] «Classification in Machine Learning». In: (2024). URL: https://www.datacamp.com/blog/classification-machine-learning (cit. on pp. 5, 6).

[3] Munendra Singh, Sanjeev Kumar, Sunil Semwal, and RS Prasad. «Residential load signature analysis for their segregation using wavelet—SVM». In: *Power Electronics and Renewable Energy Systems: Proceedings of ICPERES 2014*. Springer. 2015, pp. 863–871 (cit. on p. 7).

[4] R. S. Prasad Chuan Choong Yang Chit Siang Soh. «A Non-Intrusive Appliance Load Monitoring for Efficient Energy Consumption Based on Naive Bayes Classifier». In: (2017) (cit. on p. 8).

[5] «K-Nearest Neighbors Algorithm in Machine Learning». In: (2024). URL: https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/ (cit. on p. 8).

[6] Abdul Halim Fitra Hidiyanto. «KNN Methods with Varied K, Distance and Training Data to Disaggregate NILM with Similar Load Characteristic». In: (2020) (cit. on p. 8).

[7] Dian Jiao Xin Wu Yuchen Gao. «Multi-Label Classification Based on Random Forest Algorithm for Non-Intrusive Load Monitoring System». In: (2019) (cit. on p. 8).

[8] Zhuo Chen, Junxingxu Chen, Xianyong Xu, Shuangjian Peng, Jian Xiao, and Hong Qiao. «Non-intrusive load monitoring based on feature extraction of change-point and xgboost classifier». In: *2020 IEEE 4th Conference on Energy Internet and Energy System Integration (EI2)*. IEEE. 2020, pp. 2652–2656 (cit. on p. 8).

[9] Geoffrey Hinton Yann LeCun Yoshua Bengio. «Deep learning». In: (2015) (cit. on p. 9).

[10] 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 AAAI conference on artificial intelligence*. Vol. 32. 1. 2018 (cit. on p. 10).

[11] Hasan Rafiq, Hengxu Zhang, Huimin Li, and Manesh Kumar Ochani. «Regularized LSTM based deep learning model: first step towards real-time non-intrusive load monitoring». In: *2018 IEEE International Conference on Smart Energy Grid Engineering (SEGE)*. IEEE. 2018, pp. 234–239 (cit. on pp. 11, 12).

[12] A Vaswani. «Attention is all you need». In: *Advances in Neural Information Processing Systems* (2017) (cit. on p. 13).

[13] Jacob Devlin. «Bert: Pre-training of deep bidirectional transformers for language understanding». In: *arXiv preprint arXiv:1810.04805* (2018) (cit. on p. 16).

[14] «BERT Explained: State of the art language model for NLP». In: (2018). URL: `https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270` (cit. on p. 17).

[15] Zhenrui Yue, Camilo Requena Witzig, Daniel Jorde, and Hans-Arno Jacobsen. «Bert4nilm: A bidirectional transformer model for non-intrusive load monitoring». In: *Proceedings of the 5th International Workshop on Non-Intrusive Load Monitoring*. 2020, pp. 89–93 (cit. on pp. 18, 19).

[16] William Knottenbelt Jack Kelly. «THE UK-DALE DATASET, DOMESTIC APPLIANCE-LEVEL ELECTRICITY DEMAND AND WHOLE-HOUSE DEMAND FROM FIVE UK HOMES». In: (2015) (cit. on pp. 20–22).

[17] Christoph Klemenjak, Christoph Kovatsch, Manuel Herold, and Wilfried Elmenreich. «A synthetic energy dataset for non-intrusive load monitoring in households». In: *Scientific data* 7.1 (2020), p. 108 (cit. on pp. 22, 23).