Forecasting the usage of appliances of shared use: an analysis of simplicity over complexity

Florian Desraches*, Oihane Kamara-Esteban†‡, Diego Casado-Mansilla†‡, and Cruz E. Borges†‡

*Icam site de Toulouse, 75 Av. de Grande Bretagne, CS 97615 31076, TOULOUSE CEDEX 3
†DeustoTech - Fundación Deusto, Avda. Universidades, 24, 48007, Bilbao
‡Facultad Ingeniería, Universidad de Deusto, Avda. Universidades, 24, 48007, Bilbao
florian.desraches@2018.icam.fr
{oihane.esteban, dcasado, cruz.borges}@deusto.es

Abstract—The current revolution of the Internet of Things around the world goes far beyond the goal of simply interconnecting devices and retrieving data from them. New challenges are appearing in relation to the data-flow architectures where Cloud-based initiatives are declining over the Edge computing paradigm. In this latter approach, the resources (i.e. devices involved) are optimized by processing the data as close as possible to the source of them. However, constrained devices still struggle embedding the computation that servers carry out in the Cloud. The case of load and devices usage forecasting is a particular example of this issue where efforts are being made to simplify the processing and device architecture towards reducing energy consumption in data flow and computation. The presented research focuses on the analysis of four typical forecasting models with different levels of complexity that predict the usage of several electrical coffee machines of shared use in office buildings. The results obtained draw on shedding light on the feasibility of embedding simple yet accurate probabilistic models on constrained devices with the aim of saving energy and costs on network infrastructure.

I. INTRODUCTION

Nowadays, we are increasingly surrounded by augmented devices in private settings, work environments, and the city. Cost reductions in hardware and miniaturization of technology appear to be the enabling factors for the advent of the Internet of Things (IoT). It seems then foreseeable that predictions envisaging more 21 billions of connected devices by 2020 are in the right direction [1]. However, beyond this enormous quantity of new devices, yet more equipment is necessary to complete the IoT ecosystem: devices to store the data gathered by deployed sensors, gateways to convey information to distant Cloud services or workstations that provide high-performance computing (HPC) over the data (e.g. apply machine learning for forecasting). If we take into consideration the limited resources of our planet [2], it is necessary to come up with new architectural proposals to reduce the number of devices and the energy consumed by each of them either in computation or in data transmission.

One approach to that trend is related to Fog or Edge computing where the idea is to bring closer the computation to the source of the data [3]. Another approach is partially performing the processing locally within the node (e.g. micro-controller or the sensing device) so what is just sent out to an enhanced device is data in a high-level format of reduced dimension [4], [5]. A final approach is to make the most of the forecasting computation in low-power micro-controller units (MCUs) getting rid of intermediate devices for performing the algorithms. Whereas the latter proposal is not always applicable due to the massive requirements in terms of memory and computational throughput of some machine learning algorithms [6], we believe that many existing works on forecasting tend to over-apply machine learning algorithms instead of just testing simplistic probabilistic yet accurate methods. These naïve methods are usually based on weighted averages and are powerful enough to address the forecasting close to the source of the data. Thus, these can be easily implemented on MCUs.

In this paper, we put the focus on forecasting the usage of capsule-based coffee makers which are widespread appliances of collective use in workplaces. These devices use a large amount of electricity to keep the heater and pump ready to work and, also, the standby modes make them consume more energy than some A-class ovens or A++ fridges [7]. The reason behind the focus on forecasting is due to its importance when deploying a predictive controller that takes into account the aspect of how people make use of appliances in order to understand how energy efficiency can be achieved [8]. Forecasting has been already investigated by using auto-regressive ARIMA models for predicting the usage of electrical coffee makers [9], where the authors deployed a Cloud-based infrastructure to delegate the forecasting to a distant server. They compared ARIMA vs three Artificial Neural Networks (ANNs) configurations reporting better results for the former. However, they reckoned that ANNs may exhibit lower power consumption during the forecasting operation (once trained and hyperparameters calculated) to be embedded into an MCU. This assumption is in line with two recent papers [4], [10] that sought to demonstrate performance results of ANNs running directly on wearable devices based on ARM architectures.

 Whereas the application of already trained machine learning algorithms seems suitable for MCUs, this paper shows that for some forecasting needs, simplistic probabilistic methods yield better performance results than complex ANNs. This previous analysis is pivotal for reducing the number of devices or their resources’ capacity in certain IoT ecosystems since at implementation time they need less power to compute the prediction and besides, less memory usage.
II. RELATED WORK

Literature shows that Short Term Load Forecasting (STLF) has been widely studied over the last years, both at grid level [11], [12] and building level [13]–[15], with the models developed broadly relying on the use of ANNs and time series methods (e.g. ARIMA, ARMAX, MA, etc.). More recent papers also implement and evaluate novel load forecasting hybrid methods (i.e. ANN + ARIMA) [16], [17] that are able to capture the linear and non-linear behavior of the load curves. Forecasting the specific usage of appliances has also been a research interest in the past years with several studies employing smart meters data along with classifiers [18] and demonstrating consistent results for appliances with a lower frequency of use than that of a coffee machine (e.g. TV, dishwasher, washing machine). In [19], for example, a multilabel classifier was developed, based only on energy consumption. Aside from historical data, these STLF models are often fed with extra information, such as that given by discriminating the type of day to be forecast into a similar day approach [11], [13], [18]. This is often done by either assigning dummy variables or creating different models for each type of day. Weather data is also another type of information that is commonly used to increase the model accuracy [13], [20].

The current research aims to forecast the use of a specific kind of electrical appliance (e.g. coffee makers), which presents different load curves. This could be achieved by forecasting the energy consumption however, this approach can be biased by the appliance itself and its mode of consumption. The users behavior towards operating the modes of a coffee machine influences the overall energy consumption, as shown in [21]. The use of the standby mode, furthermore in a shared coffeemaker, can substantially change the energy consumption curve, while the actual usage of the appliance stays the same. The experiments conducted in [9] on the coffee consumption seem to show that ARIMA models are best suited for this purpose since their ARIMA model outperforms the three ANN models tested on the same problem. Considering that the authors reckoned that ANNs might be able to perform better, and the growing interest in the literature, the predictive model for the coffee consumption will be created using ANNs.

III. DATASET AND PROCEDURE

Forecasting models to improve energy efficiency in the workplace rely on understanding how people interact with the shared appliances. The dataset used for this research represents the use of 21 coffee machines [22]. Table I shows the data samples available and from which we build the forecasting model for predicting the use of these devices.

<table>
<thead>
<tr>
<th>Concept</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>deviceID</td>
<td>Name of the device the energy event comes from</td>
</tr>
<tr>
<td>energy_consumption</td>
<td>Energy consumption in Wh</td>
</tr>
<tr>
<td>consumption_time</td>
<td>Type of consumption of the event</td>
</tr>
<tr>
<td>date</td>
<td>Date written in 'YYYY-mm-dd' format</td>
</tr>
<tr>
<td>time</td>
<td>Time written in the 'HH:MM:SS' format</td>
</tr>
</tbody>
</table>

Note that these samples do not only provide information about the amount of energy consumed at a moment in time, but also identify the type of consumption. Indeed, the consumption type refers to each of the different operating modes of the coffee machine, whether it is heating the water, pouring coffee into a cup, or being in standby mode. The advantage of using this "labeled" time series is that it provides a further meaning to the evolution of the energy consumption according to the behavior of the device.

Since the 21 coffee machines were all physically located in various workplaces, each set of samples is completely independent and therefore, the data were not combined with information from any of the other devices. The dataset does not provide any additional information about the characteristics of the users of the coffee makers (e.g. number and type of users, type of building, etc.). The separate analysis of the data from each coffee maker showed that the number of samples per device varied sig-
nificantly. For example, whereas the data from the LifeUD device (Figure 1) seemed to be steady over time, the data from the Wikitoki device (Figure 2) was scarce and very spread, resulting in fewer data to explain the specific behavior of the coffee machine. For this reason, out of the initial set of 21 coffee makers, we selected the 5 that provided similarities with respect to the completeness of their time series, but also differences in their use: LifeUD, Cowork3c, SGDTech, Techabt, OfiMad.

IV. EXPERIMENTAL METHODS

The forecasting models used to predict the usage of the coffee makers (i.e. the number of coffees that will be made) are explained next. These models can be classified into probabilistic algorithms due to their use of a certain degree of randomness as part of their logic, and machine learning models which base their operations on training processes.

A. Naïve model (NA)

The Naïve model, often used in time series forecasting, follows a very simple approach: the predicted value will be set as the value of the last observed value.

\[ \hat{y}_{T+24} = y_T. \]

Given the simplicity of the NA, it is known that if any other model provides worse predictions than the NA model, the forecasting ability of said model is close to a random process.

B. Average daytype model (DT)

In order to have an assessment model that will behave accurately, a seasonal parameter must be added to the NA model. The seasonal parameter of this assessment model will be a feature called daytype. The daytype is based on the day of the week, Table II, according to the typical work schedule of office environments.

<table>
<thead>
<tr>
<th>Day</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Monday, Tuesday, Wednesday, Thursday or Friday</td>
</tr>
<tr>
<td>1</td>
<td>Saturday</td>
</tr>
<tr>
<td>2</td>
<td>Sunday</td>
</tr>
</tbody>
</table>

The final model is:

\[ \hat{y}^d_T = \frac{1}{n} \sum_{k=1}^{n} \hat{y}^d_{T-24k}, \]

where \( d \) is the daytype, \( n \) is the number of past days used for the average method.

C. Artificial Neural Network (ANN)

ANNs are based on the functionality of the human brain, and their billions of interconnected neurons [23]. The Multi-Layer Perceptron (MLP) is a supervised machine learning algorithm that can approximate a multivariate nonlinear function by training on a dataset. As depicted in Figure 3, the simplest MLPs are composed of three different layers:

- The inputs of the neural network are called features. For each of the features, there will be a neuron receiving its value, in the first layer of the MLP (leftmost layer).
- In the middle, there will be another layer of neurons, of which each neuron will be connected to every neuron of the first layer. The value of each neuron in this hidden layer will be equal to a weighted linear sum of all the neurons from the input layer, followed by a non-linear function (like the hyperbolic tan, sigmoid, etc.)
- The output layer, in our case a single neuron, receives the information from the hidden layer and returns the predicted value for the set of input features. Through an iterative algorithm, using gradient descend, for example, the minimum of a loss function (squared error for instance) for our MLP can be found, by tuning the weights and biases of the network for each iteration to achieve the best score with respect to the loss function.

D. ANN with hyperparameter tuning (GANN)

As seen previously, a machine learning model can learn its own parameters to approximate the desired function by training over a representative dataset. Hyperparameters are other parameters of the algorithm that must be set before the training of the model.

Some hyperparameters for our ANN are, for example, the size and number of hidden layers (i.e. number of neurons composing each hidden layer), the activation function of the neurons, and the solver for the weight optimization. These parameters need to be set manually since there is no ground rules theory on how to choose them and they depend heavily on the very problem to be solved. Grid search and manual
search are the most widely used strategies for hyper-parameter optimization [24] but they are computationally expensive and sometimes not efficient enough because they often converge to local minima. Therefore, as suggested by [25], an evolutionary algorithm was used to tune the hyperparameters of our MLP.

The parameters to be set by our evolutionary algorithm are the following:

- **Number of neurons in the hidden layer.** We decided against building a "deep" NN since it would be time-consuming, and certainly, considering the number of inputs, would not provide significantly better results than a non-deep NN. The shape of the MLP is the one depicted in Figure 3, with only 1 hidden layer.

- **Alpha parameter.** L2 penalty (regularization term) parameter. This parameter is used to avoid overfitting or underfitting in the model, by constraining the size of the weights of the ANN.

- **Tolerance parameter.** When the loss is not improving by at least the tolerance for two consecutive iterations, convergence is considered to be reached and training stops.

There are, indeed, more hyperparameters that can be tuned, but we decided to focus on these three because we think them to be the most influencing ones for the results of the model. The solver used in the algorithm is set to "adam" which is a stochastic gradient-based optimizer proposed by [26]. The activation function of the neurons used in our models is the hyperbolic tangent.

The evolutionary algorithm for the tuning of the hyperparameters of our ANN follows the Pseudocode of the Evolutionary Algorithm presented in [25]. From this paper, we also use the following set of genetic operators.

- **Arithmetic crossover.** Being $P_1$ and $P_2$ the individuals to be crossed and a random variable uniformly distributed between [0,1] the two off-springs $C_1$ and $C_2$ are defined as:

  $$
  C_1 = \lambda P_1 + (1 - \lambda) P_2 \\
  C_2 = \lambda P_2 + (1 - \lambda) P_1
  $$

- **Mutation Operator (Non-Uniform).** If $P$ is the individual to be mutated and $min_i$, $max_i$ the inferior, and superior limits of its i attribute, $P_i$, the two possible offsprings are defined as:

  $$
  C_1 = P_i + \Delta(t, max_i - P_i) \\
  C_2 = P_i + \Delta(t, P_i - min_i)
  $$

where $\Delta(t, y) = yr(1 - \frac{y}{T})$ with $r$ being a random variable uniformly distributed in [0,1] and $T$ the maximum number of generations. Then only one of the two offsprings is selected as the mutated element.

V. EXPERIMENTAL SETUP

In order to evaluate and compare the models, we need a methodology to measure their accuracy. The literature provides several ways of measuring a prediction accuracy in statistics. All of them have their own characteristics, but the more suitable for our particular problem is the Root Mean Square Error (RMSE) [27]. Though the Mean Absolute Percentage error (MAPE) [27] would be an accurate measure because it assesses a score proportional with the actual value, it does not fit the problem since the datasets used have a majority of zeros.

Furthermore, the evaluation of a machine learning model accuracy based on data it has been trained on is unfair and biased. The model would have insight on the data it is evaluated on (i.e. training data), therefore training and testing data are always separated. The testing dataset is also called unseen data or out-of-sample data, which has not been used in the model training. The data being used here are time series and must, therefore, be split carefully.

For the same reason, k-fold cross-validation cannot be used to evaluate the model so another method should be provided. Following the advice given in [27] an expanding window validation procedure has been used in this work. Let us consider a time series $\Omega$, Figure 4, where $t_n$ represents the timestamp of the measurement in this dataset. $\Omega$ is then evenly split in $S + 1$ smaller sets defined as $\{A_0, A_1, \ldots, A_S\}$.

![](image1)

Fig. 4. $\Omega$ time series.

With this method, $A_0$ is used as the first training set, then the model is tested on $A_1$ using the RMSE, as seen on fig. 5. Then, the first two sets are concatenated to create the new training set $A_0A_1$, which will be tested on the third one and so forth. Figure 5 graphically depicts the procedure.

![](image2)

Fig. 5. Expanding windows validation $\Omega$.

The size of the training set is therefore not constant, it increases with each iteration of testing. However, the size of the test set is constant with every iteration of the algorithm, which means that the error scores will be consistent.

VI. PERFORMANCE RESULTS

Table III summarises the results of each model (i.e. NA, DT, ANN and GANN). Note that these numbers are the mean value of the 10 errors calculated on the tests sets. The data seem to show that, except between the NA model and the others, there is no real difference in the forecasting accuracy.
The differences between the DT and the best machine learning model for each dataset is between 1% and 3.2%.

<table>
<thead>
<tr>
<th></th>
<th>NA</th>
<th>DT</th>
<th>ANN</th>
<th>GANN</th>
</tr>
</thead>
<tbody>
<tr>
<td>LifeUD</td>
<td>2.33</td>
<td>1.84</td>
<td>1.79</td>
<td>1.78</td>
</tr>
<tr>
<td>Cowork3c</td>
<td>1.29</td>
<td>1.00</td>
<td>1.01</td>
<td>1.11</td>
</tr>
<tr>
<td>SGD Tech</td>
<td>1.98</td>
<td>1.55</td>
<td>1.54</td>
<td>1.56</td>
</tr>
<tr>
<td>Techabt</td>
<td>2.20</td>
<td>1.81</td>
<td>1.84</td>
<td>1.83</td>
</tr>
<tr>
<td>OfiMad</td>
<td>3.05</td>
<td>2.24</td>
<td>2.24</td>
<td>2.38</td>
</tr>
</tbody>
</table>

Figure 6 however, shows the division of the quantiles from each set of 10 errors (by model and by device). The division varies depending on the devices. The range of the errors seems to be roughly proportional to every model from dataset to dataset. The only relevant consideration is the poor performance of GANN algorithm on the Cowork3c dataset most probably due to an “overfit”.

VII. Analysis of the Results

From all these validations, we can see that the DT, ANN, and GANN models outperform the NA. Among these three models, however, it is unclear which gives the best results. Therefore, a set of statistical tests were conducted to evaluate the level of significance of the results.

A. Quantitative analysis

For the results presented in table III, the Friedman Test [28] gives a p-value of 0.005, meaning that there are at least two of the forecasting methods (i.e. NA, DT, ANN, and GANN) that have statistical differences among them. In order to identify which are the methods with differences, an NA vs ALL Nemeyeni post-hoc analysis was also performed [29]. Table IV shows that all algorithms are indeed producing better results than the NA model.

<table>
<thead>
<tr>
<th></th>
<th>NA</th>
<th>DT</th>
<th>ANN</th>
<th>GANN</th>
</tr>
</thead>
<tbody>
<tr>
<td>p-value</td>
<td>NA</td>
<td>0.010</td>
<td>0.041</td>
<td>0.041</td>
</tr>
</tbody>
</table>

B. Qualitative analysis

From a quantitative point of view, the test of the statistical hypothesis shows that the NA behaves worse than the other 3 models: DT, ANN, GANN. Further conclusions can be drawn from the qualitative analysis. The results show that these machine learning models behave as good as the DT model. However, there are also important differences:

- **Hard implementation.** In order to implement these machine learning models, the architecture of the network has to be set, features have to be selected, hyper-parameters have to be chosen, etc.
- **Long computational time.** For these models to be able to give this kind of results, complex algorithms need to be built and run. The training time of an ANN can be time-consuming. Furthermore, for the GANN, an evolutionary algorithm has been created, implemented, and run. This kind of GA requires massive computational time because they need to train the models as much as the number of elements in the population for each generation.

- **Complex behavior.** Even though the concepts of ANN are well known and studied in the literature, their behavior and analysis are more difficult than simpler linear models.

On the other hand, the DT model relies only on a moving average composed of 3 values and the daytype features. This model does not need any training nor complex or time-consuming algorithms to be set. The DT is faster to implement, understand, and make predictions with, making it the best of the four forecasting models evaluated in this research.

VIII. Conclusions

The current research focuses on the analysis of the behavior of four forecasting algorithms (NA, DT, ANN, and GANN) in terms of accuracy and processing time when predicting the use of a shared coffeemaker in the workplace. In this case, the objective is to predict the number of coffees that will be made each day. Results demonstrate that a simple probabilistic model (DT) performs better against the machine learning models (ANN and GANN). The simplicity of the operation of the DT algorithm and its low processing needs make it an ideal benchmark candidate for use in low-power MCUs, therefore getting rid of intermediate devices for performing the algorithms within the Edge Computing paradigm. This approach is expected to reduce the time and energy needed for the processing of a unique and centralized predictive controller.

Future works in this line will focus on a) quantify the real impact in terms of energy efficiency when the algorithm is executed on a centralized controller and when it is done on distributed MCUs, and b) extrapolate this methodology to other datasets explaining the use of other shared appliances (e.g. printers, lights, project-beamers) in office buildings.

Acknowledgment

This scientific paper has been supported by the H2020 project GreenSoul. The project received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 696129.

References


Fig. 6. Summary of the RMSE for the NA, DT, ANN, GANN models.


