International Journal of Data Science and Analytics (2021) 11:105–126
https://doi.org/10.1007/s41060-021-00246-4

REGULAR PAPER

Energy-aware very fast decision tree

Eva García-Martín[1] · Niklas Lavesson[1,2] · Håkan Grahn[1] · Emiliano Casalicchio[1,3] · Veselka Boeva[1]

Received: 19 December 2018 / Accepted: 12 October 2019 / Published online: 20 March 2021

Abstract
Recently machine learning researchers are designing algorithms that can run in embedded and mobile devices, which introduces additional constraints compared to traditional algorithm design approaches. One of these constraints is energy consumption, which directly translates to battery capacity for these devices. Streaming algorithms, such as the Very Fast Decision Tree (VFDT), are designed to run in such devices due to their high velocity and low memory requirements. However, they have not been designed with an energy efficiency focus. This paper addresses this challenge by presenting the *nmin adaptation* method, which reduces the energy consumption of the VFDT algorithm with only minor effects on accuracy. *nmin adaptation* allows the algorithm to grow faster in those branches where there is more confidence to create a split, and delays the split on the less confident branches. This removes unnecessary computations related to checking for splits but maintains similar levels of accuracy. We have conducted extensive experiments on 29 public datasets, showing that the VFDT with *nmin adaptation* consumes up to 31% less energy than the original VFDT, and up to 96% less energy than the CVFDT (VFDT adapted for concept drift scenarios), trading off up to 1.7 percent of accuracy.

Keywords Data stream mining · Green artificial intelligence · Energy efficiency · Hoeffding trees · Energy-aware machine learning

1 Introduction

State-of-the-art machine learning algorithms are now being designed to run in the edge, which creates new time, memory, and energy requirements. Streaming algorithms fulfill the time and memory requirements by building models in real-time, processing data with high velocity and low memory consumption. The Very Fast Decision Tree (VFDT) algorithm [10] was the first streaming algorithm with the aforementioned properties that still achieves competitive accuracy results. However, energy consumption has not been considered during the design of the VFDT and state-of-the-art streaming algorithms. Since machine learning algorithms are starting to be designed in the edge [18,34], we believe that it is not feasible anymore to build algorithms that are not energy-aware. To address this gap, this paper presents the *nmin adaptation* method, which reduces the energy consumption of the VFDT and other Hoeffding tree algorithms, with only minor effects on accuracy.

The *nmin adaptation* method adapts the value of the $nmin$ parameter on real time and based on the incoming data. The $nmin$ parameter sets the minimum number of observed instances (batch size) at each leaf to check for a possible split. By setting a unique and adaptive value of $nmin$ at each leaf, this method allows the tree to grow faster on those paths where there are clear splits, while delaying splits on more uncertain paths. By delaying the growth in those paths where there is not enough confidence, the algorithm saves significant amount on energy on unnecessary tasks, with only minor effects on accuracy.

This paper extends our previous work "Hoeffding Trees with *nmin adaptation*" [16] by adding extensive experiments that validate the proposed method with statistical tests on the results. The results clearly show how the VFDT with *nmin adaptation* (entitled VFDT-*nmin*) obtains significantly lower levels of energy consumption (7 percent on average, up to a 31 percent) affecting accuracy up to a 1.7%. In particular,

This work is part of the research project "Scalable resource- efficient systems for big data analytics" funded by the Knowledge Foundation (Grant: 20140032) in Sweden.

✉ Eva García-Martín
evagartin@gmail.com

[1] Blekinge Institute of Technology, Karlskrona, Sweden

[2] Jönköping University, Jönköping, Sweden

[3] Sapienza University of Rome, Rome, Italy

we have investigated the energy consumption and accuracy of the VFDT, VFDT-*nmin*, and CVFDT (Concept-Adapting Very Fast Decision Tree [22]) algorithms under 29 public datasets.

We first investigated the energy consumption and accuracy of the mentioned algorithms in a baseline scenario, where we empirically validated that handling numerical attributes is much more energy consuming than handling nominal attributes. We then examined the effect of concept drift on the mentioned algorithms, concluding that (i) VFDT-*nmin* achieves significantly lower energy consumption than VFDT and CVFDT on the majority of datasets, (ii) VFDT-*nmin* scales better than VFDT (and CVFDT) in terms of energy consumption when increasing the amount of drift, and (iii) CVFDT (designed to handle concept drift) obtains lower levels of accuracy in the majority of concept drift datasets, while VFDT and VFDT-*nmin* perform similarly. Finally, we showed how VFDT-*nmin* obtains significantly lower levels of energy consumption in 5/6 real-world datasets, while obtaining the highest accuracy in all real-world datasets.

The contributions of this paper are summarized as follows:

- We present the *nmin adaptation* method, to create energy-aware Hoeffding tree algorithms, which makes them suitable for running in the edge.
- We present a general approach to create energy models for different classes of machine learning algorithms. We apply this knowledge to create an energy model for the VFDT, independent of the programming language and hardware platform. One of the findings of the model is the high energy consumption of handling numerical attributes compared to nominal attributes.
- We empirically validate the previous claim in Sect. 6.2, where we show how handling numerical attributes consumes up to 12× more energy than nominal attributes. This is visible also in Fig. 6.
- We show how VFDT-*nmin* scales better than the VFDT in terms of energy consumption when increasing the amount of drift.
- We show how VFDT-*nmin* consumes significantly less energy than the VFDT on 76% of the datasets (7% on average, up to a 31%) and than the CVFDT on all the datasets, (86% on average, up to a 97%), while affecting accuracy by less than 1%, up to a 1.7%. These claims are validated with statistical tests on the data.

In this extension we expanded the datasets from 15 to 29, investigating more phenomena such as concept drift, and validating its use in real-world datasets. We also performed statistical tests on the results, validating that the VFDT-*nmin* consumes significantly less energy than the VFDT. We also added Sect. 4.3 that explains a general way to create energy models for different classes of algorithms. Finally, we theoretically bounded the batch size of instances that VFDT-*nmin* can adapt to, showing that it can affect accuracy by at most $\frac{\delta}{p}$, where δ is the confidence value and p the leaf probability.

The rest of the paper is organized as follows. The background and related work are presented in Sect. 2. The *nmin adaptation* method is presented in Sect. 3. The energy model that profiles the energy consumption of the VFDT is presented in Sect. 4.

Section 5 presents the experimental design. Section 6 presents the results and discussion. Section 7 details the limitations of this study. Section 8 concludes the paper with the significance and impact of our work.

2 Background and related work

In this section we explain the fundamentals of the VFDT. In addition, we introduce related studies in streaming data and resource-aware machine learning.

2.1 VFDT

Very Fast Decision Tree [10] is a decision tree algorithm that builds a tree incrementally. The data instances are analyzed sequentially and only once. The algorithm reads an instance, sorts it into the corresponding leaf and updates the statistics at that leaf. To update the statistics the algorithm maintains a table for each node, with the observed attribute and class values. Updating the statistics of numerical attributes is done by saving and updating the mean and standard deviation for every new instance. Each leaf also stores the instances observed so far. After $nmin$ instances are read at that leaf, the algorithm calculates the information gain ($\overline{G}$) from all observed attributes. The difference in information gain between the best and the second best attribute ($\Delta\overline{G}$) is compared with the Hoeffding Bound [20] (ϵ). If $\Delta\overline{G} > \epsilon$, then that leaf is substituted by a node, and there is a split on the best attribute. That attribute is removed from the list of attributes available to split in that branch. If $\Delta\overline{G} < \epsilon < \tau$, a tie occurs, splitting on any of the two top attributes, since they have very similar information gain values. The Hoeffding bound (ϵ),

$$\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}} \tag{1}$$

states that the chosen attribute at a specific node after seeing n number of examples, will be the same attribute as if the algorithm has seen an infinite number of examples, with probability $1 - \delta$.

We now discuss the computational complexity of the VFDT, shown in lines 1–21 from Algorithm 1. Suppose that

n is the number of instances and m is the number of attributes. The algorithm loops over n iterations. Every step between 6 and 9 requires execution time that is proportional to m. In the worst-case scenario the computational complexity of step 7 is $O(m)$ according to [10]. The function in step 7 traverses the tree until it finds the corresponding leaf. Since the attributes are not repeated for each branch, in the worst-case scenario the tree will have a depth of m attributes. Step 8 runs in constant time. The computational complexity of this part can be evaluated to $O(n \cdot m)$. The computational complexity of the remainder part of the algorithm (from step 11 downwards) depends on $n/nmin$. Moreover, the computational complexity of steps 11 to 13 is equal to $O(m)$, while steps 16 to 18 need constant time, i.e., the computational complexity of this part is $O(n/nmin \cdot m)$. The total computational complexity of the VFDT is $O(n \cdot m) + O(n/nmin \cdot m)$ and $n >> nmin$, i.e., it can be simplified to $O(n \cdot m)$.

2.2 Related work

Energy efficiency is an important research topic in computer engineering [11]. Reams et al. [32] provide a good overview of energy efficiency in computing for different platforms: servers, desktops, and mobile devices. The authors also propose an energy cost model based on the number of instructions, power consumption, the price per unit of energy, and the execution time. While energy efficiency has mostly been studied in computer engineering, during the past years green computing has emerged. Green IT, also known as green computing, started in 1992 with the launch of the Energy Star program by the US Environmental Protection Agency (EPA) [35]. Green computing is *the study and practice of designing, manufacturing, using, and disposing computers, servers, and associated systems efficiently and effective with minimal or no environmental impact* [35]. One specific area is energy-efficient computing [32], where there is a significant focus on reducing the energy consumption of data centers [37].

In relation to big data, data centers, and cloud computing, there have been several studies that design methods for energy-efficient cloud computing [7,33]. One approach was used by Google Deep Mind to reduce the energy used in cooling their data centers [12]. These studies focused on reducing the energy consumed by data centers using machine learning to, e.g., predict the load for optimization. However, we focus on reducing the energy consumption of machine learning algorithms.

Regarding machine learning and energy efficiency, there has been a recent increase in interest toward resource-aware machine learning. The focus has been on building energy-efficient algorithms that are able to run on platforms with scarce resources [6,13,14,25]. Closely related is the work done on building energy-efficient deep neural networks [34,42]. They developed a model where the energy cost of the principal components of a neural network is defined, and then used for pruning a neural network without reducing accuracy. Some work is also conducted in building energy and computational-efficient cluster solutions [30], accelerating the original algorithm by parallelizing some of the key features.

Data stream mining algorithms analyze data aiming at reducing the memory usage, by reading the data only once without storing it. Examples of efficient algorithms are the VFDT [10] and a KNN streaming version with self-adjusting memory [27]. There have been extensions to these algorithms for distributed systems, such as the Vertical Hoeffding Tree [26], where the authors parallelize the induction of Hoeffding trees, and the Streaming Parallel Decision Tree algorithm (SPDT). Applications of streaming algorithms have been conducted in many domains, such as fraud detection [5] and time series forecasting [31]. More focused on hardware approaches to improve Hoeffding trees is the work proposed by [28], where they parallelize the execution of random forest of Hoeffding trees, together with a specific hardware configuration to improve induction of Hoeffding trees. Other work has been done where the authors present the energy hotspots of the VFDT [15]. Our proposed work in this paper focuses on a direct approach to reduce the energy consumption of the VFDT by dynamically adapting the $nmin$ parameter based on incoming data, introducing the notion of *dynamic parameter adaptation* in data stream mining.

3 *nmin* adaptation

The *nmin adaptation* method, the main contribution of this paper, aims at reducing the energy consumption of the VFDT while maintaining similar levels of accuracy. There are many computations and memory accesses dependent on the parameter $nmin$, observed in the energy model presented in Sect. 4. However, the design of the original VFDT sets the value of $nmin$ to a fixed value from the beginning of the execution. This is problematic, because there are many functions that would be computed unnecessarily if the number of $nmin$ instances is not high enough to make a confident split (e.g., *calc_entropy*, *calc_hoeff_bound*, and *get_best_att*). Our goal is to set $nmin$ to a specific value on each leaf that ensures a split, so that the $\frac{N}{nmin}$ values in Eq. 20 are only computed when needed. *nmin adaptation* adapts the value of $nmin$ to a higher one, thus making $\frac{N}{nmin}$ smaller. This approach reduces computations, reduces memory accesses, and does not affect the final accuracy, since we are only computing those functions when needed.

In another publication, the authors [15] already confirmed the high energy impact of the functions involved in calculating the best attributes. This matches with our energy model

and motivates the reasons and objectives for *nmin adaptation*:

1. Reduce the number of computations and memory accesses by adapting the value of $nmin$ to a specific value on each leaf that ensures a split.
2. Maintain similar levels of accuracy by removing only unnecessary computations, thus developing the same tree structure.

nmin adaptation sets $nmin$ to the estimated number of instances required to guarantee a split with confidence $1-\delta$. The higher the value of $nmin$, the higher the chance to split. However, setting $nmin$ to a very high value can decrease accuracy if the growth of the tree is significantly delayed, and setting $nmin$ to a lower value increases the accuracy at the expense of energy, as it has to calculate the $\overline{G}$ of all attributes even when there are not enough instances to make a confident split. Thus, our solution allows for a faster growth in the branches with a higher confidence to make a split and delays the growth in the less confident ones. We have identified two scenarios that are responsible for not splitting. We set $nmin$ to a different value to address these scenarios, depending on the incoming data.

The two scenarios are the following:

Scenario 1 *($\Delta\overline{G} < \epsilon$ and $\Delta\overline{G} > \tau$) Fig. 1, left plot. The attributes are not too similar, since $\Delta\overline{G} > \tau$, but their difference is not big enough to make a split, since $\Delta\overline{G} < \epsilon$. The solution is to wait for more examples until ϵ (green triangle) decreases and is smaller than $\Delta\overline{G}$ (black star). Following this reasoning, $nmin = \left\lceil \frac{R^2 \cdot ln(1/\delta)}{2\cdot(\Delta G)^2} \right\rceil$, obtained by setting $\epsilon = \Delta\overline{G}$ in (1), to guarantee that $\Delta\overline{G} \geq \epsilon$ will be satisfied in the next iteration, creating a split.*

Scenario 2 *($\Delta\overline{G} < \epsilon$ and $\Delta\overline{G} < \tau$ but $\epsilon > \tau$) The top attributes are very similar in terms of information gain, but ϵ is still higher than τ, as can be seen in Fig. 1, right plot. The algorithm needs more instances so that ϵ (green triangle) decreases and is smaller than τ (red dot). Following this reasoning, $nmin = \left\lceil \frac{R^2 \cdot ln(1/\delta)}{2\cdot\tau^2} \right\rceil$, by setting $\epsilon = \tau$ in (1). In the next iteration $\epsilon \leq \tau$ will be satisfied, forcing a split.*

The upper bound of the batch size is given in scenario 2, since the lower the ϵ the higher the number of instances. The lowest value of ϵ is when $\epsilon = \tau$, because if $\epsilon < \tau$ then a split occurs, so there would be no need to adapt the value of $nmin$ in that case. The lower bound of the batch size is given in scenario 1, and for the case when $\Delta G = \epsilon$, which is approximately the initial value of $nmin$. Thus, the adaptive size of the batch can be bounded to the following interval: $\left\lceil \text{initial } nmin, \frac{R^2 \cdot ln(1/\delta)}{2\cdot\tau^2} \right\rceil$.

The upper and lower bound can be related to the notion of *intensional disagreement*, which is described in the VFDT original paper as *the probability that the path of an example through DT1 will differ from its path through DT2* [10]. There, the authors propose *Theorem 1*, which states that the intensional disagreement between the Hoeffding tree and the batch tree is lower than $\frac{\delta}{p}$, where p is the leaf probability.

The original VFDT and the VFDT-*nmin* only differ in the way the node split is created. The VFDT-*nmin* requires that more examples are seen at a node to make an informed decision. Based on the VFDT original paper, the authors confirm that when more examples are read at the node, i.e., increasing $nmin$, the value of δ decreases, increasing the confidence of the split. In this case, increasing the $nmin$ would create a tree that at most differs with the original (batch) tree in $\frac{\delta}{p}$, in a scenario with an infinite data stream. Since in our case the upper bound (around 3000 instances) is significantly lower than the size of the stream (around 1 million instances), accuracy should not be significantly affected.

The pseudocode of VFDT-*nmin* is presented in Algorithm 1. The specific part of *nmin adaptation* is shown in lines 22–26, where we specify how $nmin$ is going to be adapted based on the scenarios explained above. The idea is that, when those scenarios occur, we adapt the value of $nmin$, so that they do not occur in the next iteration, thus ensuring a split. Figure 2 shows a diagram of the main functions and functionalities of the VFDT-*nmin* algorithm, inspired in the work by [8].

In relation to the computational complexity of the *nmin adaptation*, we can observe that this method does not add any overhead. Thus, the computational complexity of VFDT-*nmin* is $O(n \cdot m)$.

In relation to concept drift, state-of-the-art Hoeffding tree algorithms that are able to handle concept drift [1] also use the $nmin$ parameter to decide when to check for possible splits. Thus, our method can be directly applied to those class of algorithms and should theoretically output similar results. We have planned that for future works.

Finally, we show an example of how *nmin adaptation* works for two of the datasets used in the final experiments. These datasets are described in Table 1. Figure 3 shows the $nmin$ variation for the cases when $nmin$ is initially set to 20, 200, 2000. So, after those instances, $nmin$ will adapt to a higher value depending on the data observed so far at that specific leaf. The airline dataset shows many adaptations of $nmin$ when $nmin$ is initially set to 20. This is expected, since we are showing the adaptations per leaf, so at the beginning all the leaves starting with $nmin = 20$ will adapt that value to a much larger one. The same reasoning occurs when $nmin = 200$ initially, since there will be less adaptations because the leaves need to wait for more instances, and there is a higher chance to split when more instances are read. The poker dataset exhibits a different behavior, where $nmin$

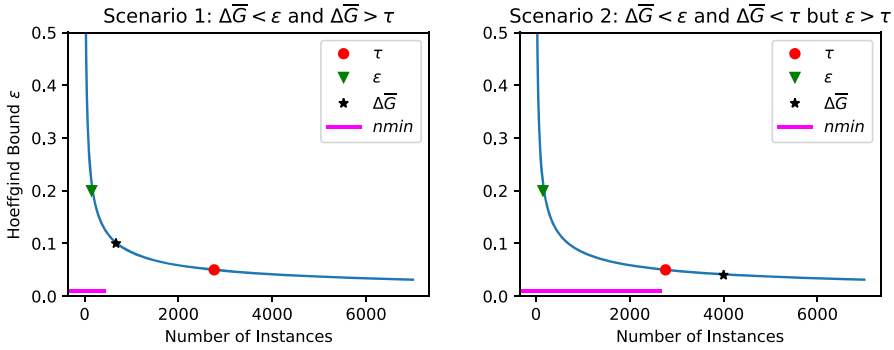

Fig. 1 Variation of ϵ (Hoeffding bound) with the number of instances. *nmin adaptation method* for scenarios 1 and 2

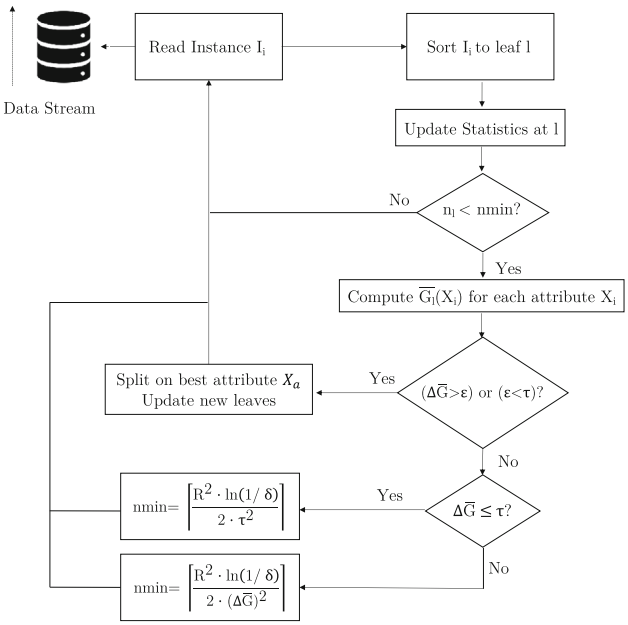

Fig. 2 Flowchart diagram of the VFDT-*nmin* algorithm

adapts to a higher value, 30, 491. This occurs in Scenario 2, but since the poker dataset has 10 classes, the range R of Hoeffding bound Eq. (1) is higher. Finally, looking at the cases where $nmin = 2000$ (green), we observe how there is almost no adaptation. VFDT-*nmin* either splits after 2000 instances, or it adapts $nmin = 2763$ or $nmin = 30,491$, because the attributes are very similar.

4 Energy consumption of the VFDT

Energy consumption is a necessary measurement for today's computations, since it has a direct impact on the electricity bill of data centers and battery life of embedded devices. However, measuring energy consumption is a challenging task. As has been shown by researchers in computer architecture, estimating the energy consumption of a program is not straightforward and is not as simple as measuring the execution time, since there are many other variables involved [29].

In this section, we first give a general background on energy consumption and its relationship to software energy consumption. We then propose a general approach to create energy models applicable to any class of algorithms. We then use this approach to create a theoretical energy model for the VFDT algorithm, based on the number of instances of the stream, and number of numerical and nominal attributes.

4.1 General energy consumption

Energy efficiency in computing usually refers to a hardware approach to reduce the power consumption of processors or

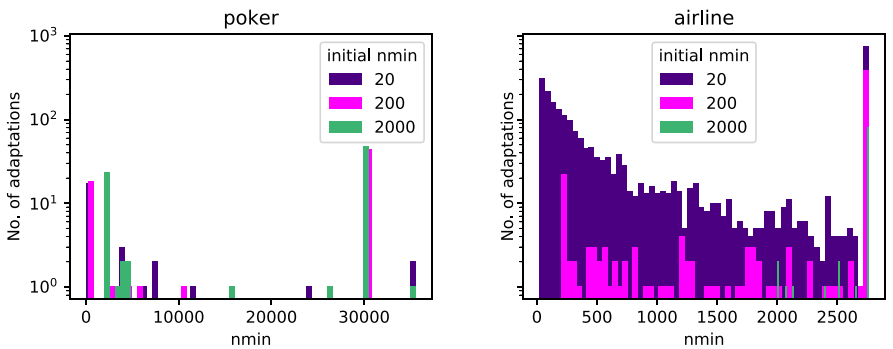

Fig. 3 Variation of *nmin* for *nmin* initially set to 20, 200, 2000 on poker and airline datasets (Table 1). With a lower *nmin*, *nmin adaptation* adapts *nmin* to a higher value more frequently. The peaks on $nmin = 2763$ and $nmin = 30,491$ are explained by Scenario 2, since τ is a fixed hyperparameter

Algorithm 1 VFDT-*nmin*: Very Fast Decision Tree with *nmin adaptation*

HT: Tree with a single leaf (the root)
X: set of attributes
$G(\cdot)$: split evaluation function
τ: hyperparameter set by the user
$nmin$: hyperparameter initially set by the user
while stream is not empty **do**
  Read instance I_i
  Sort I_i to corresponding leaf l using HT
  Update statistics at leaf l
  Increment n_l: instances seen at l
  if $nmin \leq n_l$ **then**
    Compute $\overline{G_l}(X_i)$ for each attribute X_i
    X_a, X_b = attributes with the highest $\overline{G_l}$
    $\Delta\overline{G} = \overline{G_l}(X_a) - \overline{G_l}(X_b)$
    Compute ϵ
    if $(\Delta\overline{G} > \epsilon)$ or $(\epsilon < \tau)$ **then**
      Replace l with a node that splits on X_a
      for each branch of the split **do**
        New leaf l_m with initialized statistics
      end for
    else
      Disable attr $\{X_p | (\overline{G_l}(X_p) - \overline{G_l}(X_a)) > \epsilon\}$
      if $\Delta\overline{G} \leq \tau$ **then**

        $nmin = \left\lceil \frac{R^2 \cdot ln(1/\delta)}{2 \cdot \tau^2} \right\rceil$
      else
        $nmin = \left\lceil \frac{R^2 \cdot ln(1/\delta)}{2 \cdot (\Delta G)^2} \right\rceil$
      end if
    end if
  end if
end while

ways to make processors handle more operations using the same amount of power [24].

Power is the rate at which energy is being consumed. The average power during a time interval T is defined as [40]:

$$P_{avg} = \frac{E}{T} \tag{2}$$

where E, energy, is measured in joules (J), P_{avg} is measured in watts (W), and time T is measured in seconds (s). We can distinguish between dynamic and static power. Static power, also known as leakage power, is the power consumed when there is no circuit activity. Dynamic power, on the other hand, is the power dissipated by the circuit, from charging and discharging the capacitor [11]:

$$P_{dynamic} = \alpha \cdot C \cdot V_{dd}^2 \cdot f \tag{3}$$

where α is the activity factor, representing the percentage of the circuit that is active. V_{dd} is the voltage, C the capacitance, and f the clock frequency measured in hertz (Hz). Energy is the effort to perform a task, and it is defined as the integral of power over a period of time [11]:

$$E = \int_0^T P(t)\mathrm{d}t \tag{4}$$

In this study we focus on the measurement of energy consumption, since it gives an overview of how much power is consumed in an interval of time.

Finally, we conclude with an explanation of how programs consume energy. The total execution time of a program is defined as [11]:

$$T_{exe} = IC \times CPI \times T_c \tag{5}$$

where IC is the number of executed instructions, CPI (clock cycles per instruction) is the average number of clock cycles needed to execute each instruction, and T_C is the clock cycle time of the processor. The total energy consumed by a program is:

$$E = IC \times CPI \times EPC \tag{6}$$

where EPC is the energy per clock cycle, and it is defined as

$$EPC \propto C \cdot V_{dd}^2 \tag{7}$$

The value CPI depends on the type of instruction, since different instructions require different number of clock cycles to complete. However, measuring only time does not give a realistic view on the energy consumption, because there are instructions that can consume more energy due to a long delay (e.g., memory accesses), or others that consume more energy because of a high requirement of computations (floating point operations). Both could obtain similar energy consumption levels; however, the first one would have a longer execution time than the last one.

4.2 Theoretical energy model

This section explains how to create theoretical energy models for different algorithms from a general perspective. We then apply this knowledge in Sect. 4.3 to create a specific model for the energy consumed by the VFDT.

The energy consumed by an algorithm can be estimated by identifying the type of events in the algorithm, e.g., floating point calculation, memory accesses, etc. We propose the following steps:

1. Identify the main functions of the algorithm
2. Define the type of events that are of interest, i.e., memory accesses, cache misses, floating point operations, and integer point operations (such as Eq. (8))
3. Map the different algorithm functions to the type of events, to end up with an equation based on the number of memory accesses and number of computations [such as Eqs. (12) and (15)]

4. If needed, characterize the amount of energy consumed per type of event for that specific processor, based on the work by [21].

This approach can be adapted to any algorithm and can thus provide insights into the energy behavior of an algorithm. The model is independent of programming language, etc., and focuses on basic operations in a particular algorithm, including access to resources such as data and memory. One of the main objectives with the model is to gain an understanding of which parts of the algorithms are most energy consuming.

In practice, when we measure the energy consumption on a real system, this can be done in two ways: externally, i.e., we measure the current, etc., consumed by the hardware, or internally, i.e., we measure how the software behaves. Most internal energy measurement estimation tools work similarly, i.e., they count a number of hardware events using performance counters in the processors. These numbers are then fed into an energy model (similar to our simplified model), and then an estimation of the energy consumption is done. The RAPL framework by Intel that we use in our paper works in this way.

4.3 VFDT energy model

The energy model of the VFDT is based on the different steps to create an energy model from Sect. 4.2. The functions are taken from the pseudocode of the VFDT [10]. Algorithm 1 shows the pseudocode for the VFDT algorithm with the *nmin adaptation* functionality added, but the main functions can also be observed there. The main functions are the following:

- *Sort instance to leaf* When an instance is read, the first step is to traverse the tree based on the attribute values of that instance, to reach the correspondent leaf.
- *Update attributes* Once the leaf is reached, the information at that leaf is updated with the attribute/class information of the instance. The update process is different if the attribute is numerical or nominal. For nominal attributes a simple table with the counts is needed. For updating the numerical attribute the mean and the standard deviation are updated.
- *Update instance count* After each instance is read, the counter at that leaf is updated.
- *Calculate entropy* Once *nmin* instances are observed at a leaf, the entropy (information gain in this case) is calculated for each attribute.
- *Get best attribute* The attributes with the highest information gain are chosen.
- *Calculate Hoeffding bound* We then compare the difference between the best and the second best attribute with the Hoeffding bound, calculated with Eq.(1).
- *Create new node* If there is a clear attribute to split on, we split on the best attribute creating a new node.

Based on the information provided above, we present the energy consumption of the VFDT in the following model:

$$E_{VFDT} = E_{comp} + E_{cache\_tot} + E_{cache\_miss\_tot}, \tag{8}$$

where E_{comp} is the energy consumed on computations, E_{cache_tot} is the energy consumed on cache accesses, and $E_{cache\_miss\_tot}$ is the energy consumed on cache misses. They are defined as follows:

$$E_{comp} = n_{FPU} \cdot E_{FPU} + n_{INT} \cdot E_{INT}, \tag{9}$$

where n_{FPU} is the number of floating point operations, E_{FPU} is the average energy per floating point operation, n_{INT} is the number of integer operations, and E_{INT} is the average energy per integer operation.

$$E_{cache\_tot} = n_{cache} \cdot E_{cache}, \tag{10}$$

where n_{cache} is the number of accesses to cache, and E_{cache} is the average energy per access to cache. Finally,

$$E_{cache\_miss\_tot} = n_{cache\_miss} \cdot (E_{cache\_miss} + E_{DRAM}), \tag{11}$$

where n_{cache_miss} is the number of cache misses, E_{DRAM} is the average energy per DRAM access, and E_{cache_miss} is the average energy per cache miss.

The next step is to map these n_{FPU}, n_{INT}, n_{cache}, and n_{cache_miss} to the VFDT algorithm's functions, explained at the beginning of this section.

$$\begin{aligned} n_{FPU} = {} & n_{comp}(updating\_numerical\_atts) \\ & + n_{comp}(calc\_entropy) \\ & + n_{comp}(calc\_hoeff\_bound) \\ & + n_{comp}(get\_best\_att) \end{aligned} \tag{12}$$

$$\begin{aligned} n_{INT} = {} & n_{comp}(updating\_nominal\_atts) \\ & + n_{comp}(updating\_instance\_count), \end{aligned} \tag{13}$$

where $n_{comp}(f_i)$ refers to the number of computations required by function f_i.

$$n_{cache} = n_{acc}(updating\_atts) \tag{14}$$

$$\begin{aligned} n_{cache\_miss} = {} & n_{acc}(sorting\_instance\_to\_leaf) \\ & + n_{acc}(updating\_atts) \\ & + n_{acc}(calc\_entropy) \\ & + n_{acc}(calc\_hoeff\_bound) \\ & + n_{acc}(new\_node), \end{aligned} \tag{15}$$

where $n_{acc}(f_i)$ represents the number of accesses to memory or cache in order to execute function f_i. The number of cache and memory accesses of updating the attributes ($updating\_atts$) will depend on the block size of the cache. If the block size is big enough, then we would have one cache miss to update the information of the first attribute, and then cache hits for the rest of the attributes. However, if there are many attributes, thus not fitting on the block size B, then there will be a cache miss for every attribute that exceeds the block size. We also consider the presence of a cache miss every time a node of the tree is traversed, and every time we calculate the entropy and Hoeffding bound values.

The last step is to express these number of accesses and computations based on the number of instances (N), the $nmin$ value, the number of numerical attributes (A_f), the number of nominal attributes (A_i), and the block cache size B. We then obtain the following:

$$\begin{aligned} n_{FPU} &= N \cdot A_f + \frac{N}{nmin} \cdot (A_f + A_i) \\ &\quad + \frac{N}{nmin} + \frac{N}{nmin} \cdot (A_f + A_i) \\ &= N \cdot A_f + 2 \cdot \frac{N}{nmin} \cdot (A_f + A_i) + \frac{N}{nmin} \end{aligned} \tag{16}$$

Updating numerical attributes is one access per instance per numerical attribute; calculating the entropy is one access per attribute (thus the sum of nominal and numerical attributes) every $nmin$ instances; calculating the Hoeffding bound is one access every $nmin$ instances; and calculating the best attribute is the same as calculating the entropy.

$$n_{INT} = N \cdot A_i + N \tag{17}$$

Updating nominal attributes is, as before, one access per instance per nominal attribute and one access per instance for updating the counter.

$$n_{cache} = N \cdot \left(A_f + A_i - \frac{A_f + A_i}{B} \right) \tag{18}$$

To update the attributes, we consider one cache hit per all attributes per instance, minus all the attributes that do not fit on the block size B and create cache misses.

$$\begin{aligned} n_{cache\_miss} &= N \cdot \left(A_f + A_i + \frac{A_f + A_i}{B} \right) \\ &\quad + \frac{N}{nmin} + \frac{N}{nmin} + \frac{N}{nmin} \\ &= N \cdot \left(A_f + A_i + \frac{A_f + A_i}{B} \right) + 3 \cdot \frac{N}{nmin} \end{aligned} \tag{19}$$

To calculate the number of accesses of sorting an instance to a leaf we assume that we need to access one level per attribute, which is the worst-case scenario. So the total number of accesses in this case is one per instance per attribute. To update the attributes, as was explained before, it is one miss per all attributes that exceed the block size B, per instance. To access the needed values to calculate the entropy, the Hoeffding bound, and to split, we consider one access every $nmin$ instances.

Based on Eqs. (8), (9), (10), (11), (16), (17), (18), and (19), our final energy model equation is the following:

$$\begin{aligned} E_{VFDT} = {} & E_{FPU} \cdot \left(N \cdot A_f + 2 \cdot \frac{N}{nmin} \cdot (A_f + A_i) \right. \\ & \left. + \frac{N}{nmin} \right) + E_{INT} \cdot (N \cdot A_i + N) \\ & + E_{cache} \cdot \left(N \cdot \left(A_f + A_i - \frac{A_f + A_i}{B} \right) \right) \\ & + (E_{cache\_miss} + E_{DRAM}) \cdot \left(N \cdot \left(A_f + A_i \right. \right. \\ & \left. \left. + \frac{A_f + A_i}{B} \right) + 3 \cdot \frac{N}{nmin} \right) \end{aligned} \tag{20}$$

This is a general and simplified model of how the VFDT algorithm consumes energy. The energy values (i.e., E_{cache}, E_{FPU}, E_{INT}, E_{DRAM}, and E_{cache_miss}) will vary depending on the processor and architecture, although there is a lot of research that ranks these operations based on their energy consumption [21]. For instance, a DRAM instruction consumes three orders of magnitude more energy than an ALU operation. We can see the importance of the number of attributes in the overall energy consumption of the algorithm. Since E_{FPU} is significantly higher than E_{INT}, numerical attributes have a higher impact on energy consumption than nominal attributes.

5 Experimental design

In comparison with our previous work [16], we have designed extensive experiments to better understand the behavior of VFDT, VFDT-*nmin*, and CVFDT (Concept-Adapting Very Fast Decision Tree [22]) in three setups:

- Baseline
- Concept Drift

– Real World

The baseline setup presents a sensitivity analysis where we evaluate the accuracy and energy consumption of the mentioned algorithms while varying the input parameters of the dataset. In particular, we vary the number of instances, nominal, and numerical attributes, to understand how that affects energy consumption and accuracy. We have already observed through our energy model that the number of numerical attributes affected significantly the energy consumption. This setup aims at validating empirically that observation, to be used as a baseline to the other experiments.

The concept drift setup investigates the effect of concept drift in accuracy and energy consumption. We have taken three synthetically generated datasets, LED, RBF, and waveform, and added two levels of change.

The real-world setup investigates the energy consumption and accuracy of the VFDT, VFDT-*nmin*, and CVFDT, in six real datasets.

The datasets used in our experiments are explained, per setup, in Table 1, and described in Sect. 5.1. We run the experiments on a machine with an 3.5 GHz Intel Core i7, with 16 GB of RAM, running OSX. To estimate the energy consumption we use Intel Power Gadget,[1] that accesses the performance counters of the processor, together with Intel's RAPL interface to obtain energy consumption estimations [9]. The implementation of VFDT-*nmin* together with the scripts to conduct the experiments is publicly available.[2]

5.1 Datasets

We have used synthetic datasets for the baseline and concept drift setup and six different real datasets for the last setup. The choice of datasets is inspired by the work of [3].

The datasets are described in Table 1. There are a total of 29 datasets, 23 artificial datasets generated with Massive Online Analysis (MOA) [2], and 6 real-world datasets. The artificial datasets are listed in Table 1.

*RT_inst_*A_i*_*A_f: Random tree dataset with $inst$ number of instances, A_i number of nominal attributes, and A_f number of numerical attributes. This dataset is inspired from the dataset proposed by the authors of the original VFDT [10]. It first builds the tree, by randomly selecting attributes to split, assigning random values to the leaves. The leaves will be the classes of the instances. Then new examples are generated, with random attribute values, and they are labeled based on the already created tree.

LED_x: LED dataset with x attributes with drift. The goal is to predict the digit on a LED display with seven segments, where each attribute has a 10% chance of being inverted [4].

Table 1 Datasets used in the experiment to compare VFDT, VFDT-*nmin*, and CVFDT

Dataset	Train	Test	A_i	A_f	Class
Baseline					
RT_10k_10_10	6700	3300	0	10	2
RT_100k_10_10	67,000	33,000	0	10	2
RT_1M_10_10	670,000	330,000	0	10	2
RT_10M_10_10	6,700,000	3,300,000	0	10	2
RT_1M_10_0	670,000	330,000	10	0	2
RT_1M_20_0	670,000	330,000	20	0	2
RT_1M_30_0	670,000	330,000	30	0	2
RT_1M_40_0	670,000	330,000	40	0	2
RT_1M_50_0	670,000	330,000	50	0	2
RT_1M_0_10	670,000	330,000	0	10	2
RT_1M_0_20	670,000	330,000	0	20	2
RT_1M_0_30	670,000	330,000	0	30	2
RT_1M_0_40	670,000	330,000	0	40	2
RT_1M_0_50	670,000	330,000	0	50	2
Concept drift					
LED	670,000	330,000	24	0	10
LED_3	670,000	330,000	24	0	10
LED_7	670,000	330,000	24	0	10
RBF	670,000	330,000	0	10	2
RBF_m	670,000	330,000	0	10	2
RBF_f	670,000	330,000	0	10	2
waveform	670,000	330,000	0	21	3
waveform_5	670,000	330,000	0	21	3
waveform_10	670,000	330,000	0	21	3
Real world					
Airline	539,383	99,999	4	3	2
Electricity	30,359	14,953	1	6	2
Poker	555,564	273,637	5	5	10
CICIDS	461,802	230,901	78	5	6
Forest	387,342	193.670	40	10	7
kddcup	3,265,621	1,632,810	7	34	23

A_i and A_f represent the number of nominal and numerical attributes, respectively. The details of each dataset are presented in Sect. 5.1

RBF_v: The radial-based function (RBF) dataset has 10 numerical attributes. The generator creates n number of centroids, each with a random center, class label, and weight. Each new example randomly selects a center, considering that centers with higher weights are more likely to be chosen. The chosen centroid represents the class of the example. Drift is introduced by moving the centroids with speed v, either moderate (0.001), or fast (0.01). More details are given by [3].

waveform_x: Waveform dataset with x attributes with drift. The waveform dataset comes from the UCI repository. The function generates a wave as a combination of two or three

[1] https://software.intel.com/en-us/articles/intel-power-gadget-20.

[2] https://github.com/egarciamartin/hoeffding-nmin-adaptation.

Table 2 Difference in accuracy (ΔAcc) and energy consumption (ΔEnergy) between VFDT and VFDT-*nmin* and VFDT-*nmin* and CVFDT

Dataset	VFDT-nmin versus VFDT		VFDT-nmin versus CVFDT	
	ΔAcc (%)	ΔEnergy (%)	ΔAcc (%)	ΔEnergy (%)
Baseline				
RT_10K_10_10	**0.00**	**−12.35**	**1.91**	**−66.44**
RT_100K_10_10	**0.00**	**−2.44**	−9.67	**−82.23**
RT_1M_10_10	−0.33	**−4.77**	**3.91**	**−89.86**
RT_10M_10_10	−0.31	**−1.70**	**6.60**	**−84.02**
RT_1M_0_10	−0.25	**−0.26**	**1.41**	**−94.05**
RT_1M_0_20	−0.20	**−2.75**	**1.39**	**−93.33**
RT_1M_0_30	−0.03	**−2.04**	**1.04**	**−95.41**
RT_1M_0_40	**0.05**	**−1.47**	**1.19**	**−96.29**
RT_1M_0_50	−0.06	**−1.37**	**0.98**	**−97.04**
RT_1M_10_0	**0.73**	**−1.15**	**11.63**	**−84.15**
RT_1M_20_0	−1.65	**−1.19**	**9.96**	**−84.50**
RT_1M_30_0	**3.88**	2.04	**17.84**	**−83.98**
RT_1M_40_0	**5.13**	0.78	**16.34**	**−85.40**
RT_1M_50_0	**25.28**	0.33	**42.80**	**−84.29**
Concept drift				
LED	**1.78**	2.21	**2.25**	**−82.19**
LED_3	**1.78**	1.14	**2.25**	**−81.42**
LED_7	**1.78**	0.33	**2.25**	**−81.47**
RBF	−0.89	**−11.72**	**1.16**	**−93.85**
RBF_m	−0.29	**−19.96**	**0.63**	**−91.18**
RBF_f	**0.28**	**−19.95**	**1.45**	**−95.80**
waveform	−1.09	**−24.92**	**1.26**	**−87.18**
waveform_10	−0.85	**−21.87**	**1.49**	**−86.74**
waveform_5	−0.85	**−21.98**	**1.49**	**−86.81**
Real				
CICIDS17	**0.18**	**−3.70**	**2.09**	**−89.84**
Airline	**0.07**	**−11.19**	**11.97**	**−87.39**
Electricity	**3.32**	**−31.61**	**5.10**	**−77.07**
Forest	**0.20**	**−2.47**	**3.19**	**−83.36**
kddcup	**0.00**	**−5.82**	**39.62**	**−82.80**
Poker	**3.17**	8.81	**16.98**	**−82.37**
Average	1.41	−6.59	6.91	−86.57

A positive number in accuracy means that VFDT-*nmin* obtained a higher accuracy. A negative number in energy means that the VFDT-*nmin* reduced the energy consumption by that percentage. Higher accuracy and lower energy consumption of the VFDT-*nmin* are presented in bold

Table 3 Results from performing a Wilcoxon signed-rank test on the differences in accuracy and energy consumption between the VFDT and VFDT-*nmin* on all datasets

Measure	*p* Value	Null Hypothesis
Accuracy	5.89×10^{-6}	Rejected. Lower than 0.01
Energy	2.56×10^{-6}	Rejected. Lower than 0.01

base waves. The task is to differentiate between the three waves.

We have also tested six real datasets, some of them available from the MOA official Web site.[3] The poker dataset is a normalized dataset available from the UCI repository. Each instance represents a hand consisting of five playing cards, where each card has two attributes: suit and rank. The electricity dataset is originally described in [19] and is frequently used in the study of performance comparisons. Each instance represents the change of the electricity price based on different attributes such as day of the week, represented by the

[3] https://moa.cms.waikato.ac.nz/datasets/.

Table 4 Baseline setup. Energy consumption and accuracy results of varying the number of nominal attributes, numerical attributes, and instances

Dataset	Accuracy (%)			Total energy(J)			CPU energy (J)			DRAM energy(J)		
	vfdt-nmin	cvfdt	vfdt	vfdt-nmin	cvfdt	vfdt	vfdt-nmin	cvfdt	vfdt	vfdt-nmin	cvfdt	vfdt
# Instances												
RT_10K_10_10	**66.42**	64.52	**66.42**	**3.52**	10.49	4.02	3.39	10.10	3.86	0.13	0.39	0.15
RT_100K_10_10	**74.45**	**84.12**	74.45	**53.58**	301.52	54.92	51.77	287.76	52.94	1.82	13.76	1.98
RT_1M_10_10	94.31	90.40	**94.64**	**537.88**	5303.19	564.81	516.12	5062.22	541.86	21.77	240.98	22.94
RT_10M_10_10	98.09	91.49	**98.41**	**10063.04**	62963.14	10237.39	9567.58	60072.85	9731.41	495.46	2890.29	505.98
# Nominal Att												
RT_1M_10_0	**97.07**	85.44	96.35	**159.90**	1008.99	161.75	153.27	971.01	155.25	6.63	37.99	6.50
RT_1M_20_0	79.60	69.64	**81.25**	**268.46**	1731.49	271.68	258.43	1668.67	261.55	10.03	62.82	10.13
RT_1M_30_0	**74.78**	56.93	70.90	382.26	2386.39	**374.63**	367.55	2305.67	361.07	14.71	80.72	13.55
RT_1M_40_0	**95.38**	79.05	90.25	494.98	3389.69	**491.15**	476.80	3267.65	472.49	18.18	122.04	18.66
RT_1M_50_0	**98.24**	55.44	72.96	604.05	3843.90	**602.05**	582.30	3712.76	580.33	21.75	131.14	21.72
# Numerical Att												
RT_1M_0_10	99.39	97.98	**99.64**	**408.95**	6867.78	410.03	394.53	6464.23	395.51	14.43	403.55	14.52
RT_1M_0_20	99.36	97.97	**99.56**	**723.50**	10847.07	743.93	689.65	10225.92	708.61	33.85	621.16	35.32
RT_1M_0_30	99.26	98.21	**99.28**	**1060.73**	23129.46	1082.80	1016.84	21075.55	1037.06	43.89	2053.91	45.75
RT_1M_0_40	**99.44**	98.26	99.40	**1443.50**	38868.37	1465.11	1396.36	34382.45	1416.21	47.14	4485.93	48.90
RT_1M_0_50	99.22	98.24	**99.28**	**1777.81**	60079.04	1802.47	1719.28	52027.50	1743.44	58.54	8051.54	59.03

RT_$inst$_A_i_A_f, where $inst$ is the number of instances, A_i is the number of nominal attributes, and A_f is the number of numerical attributes. Algorithms: VFDT, VFDT-*nmin*, and CVFDT. Measurements: accuracy, total energy, CPU energy, DRAM energy. Total energy = CPU energy + DRAM energy. Higher accuracy and lower total energy consumption values per setup are presented in bold

Australian New South Wales Electricity Market. The airline dataset [23] predicts if a given flight will be delayed based on attributes such as airport of origin and airline. The forest dataset *contains the forest cover type for* 30×30 *meter cells obtained from US Forest Service (USFS) Region 2 Resource Information System (RIS) data*.[4] The KDDCUP dataset[5] [39] was created for a competition in 1999, where the goal was to detect network intrusions. A similar but newer data are the CICIDS [36], a dataset in cybersecurity where the task is again to detect intrusions.

5.2 Algorithms

We compare VFDT, VFDT-*nmin*, and CVFDT under the mentioned datasets. The initial value of $nmin$ is set to 200, which is the default value used by the authors of the VFDT. We evaluate all algorithms based on the following measures: accuracy (percent of correctly classified instances), energy consumed by the processor, and energy consumed by the DRAM. We evaluate the accuracy by having a training set and a test set that is different from the training set, as can be observed in Table 1. We have not performed yet prequential evaluation as with this method, however that is planned for future works.

5.3 Statistical significance

To test whether the differences between accuracy and energy consumption between the VFDT and the VFDT-*nmin* are statistically significant, we perform a nonparametric test, namely the Wilcoxon signed-rank test [41].

We choose a nonparametric test after having tested for normality between the differences in accuracy and energy consumption between the VFDT and VFDT-*nmin*, obtaining p values smaller than 0.01. We choose this test in particular since the observations are paired based on the dataset and thus can be considered as dependent.

We first test whether the VFDT-*nmin* obtained significantly higher accuracy than the VFDT, since the data indicate that the average of the accuracy of the VFDT-*nmin* is 1.41% higher than for the VFDT. Thus, we propose the following null and one-tailed alternative hypothesis [38]:

$\boldsymbol{H_0}$: $\mu_{A1} = \mu_{A2}$, where μ_{A1} represents the mean of accuracy values for the VFDT, and μ_{A2} represents the mean of accuracy values for VFDT-*nmin*. Thus, the null hypothesis states that the means of the accuracy values between the VFDT and the VFDT-*nmin* are equal.

$\boldsymbol{H_1}$: $\mu_{A1} < \mu_{A2}$, stating that the mean of the accuracy of VFDT-*nmin* is higher than the mean of the accuracy of the VFDT.

[4] https://moa.cms.waikato.ac.nz/datasets/.

[5] http://kdd.ics.uci.edu/databases/kddcup99/task.html.

Table 5 Energy consumption and accuracy results of concept drift datasets

Dataset	Accuracy (%)			Total energy (J)			CPU energy (J)			DRAM energy (J)		
	vfdt-nmin	cvfdt	vfdt	vfdt-nmin	cvfdt	vfdt	vfdt-nmin	cvfdt	vfdt	vfdt-nmin	cvfdt	vfdt
LED	**72.77**	70.52	70.99	240.20	1348.48	**235.02**	229.38	1287.60	224.43	10.82	60.89	10.59
LED_3	**72.77**	70.52	70.99	242.19	1303.67	**239.46**	232.25	1247.24	228.97	9.94	56.43	10.48
LED_7	**72.77**	70.52	70.99	241.75	1304.80	**240.97**	231.32	1246.60	230.49	10.44	58.20	10.48
RBF	89.01	87.85	**89.90**	**529.95**	8622.95	600.31	506.49	7978.81	572.38	23.46	644.14	27.92
RBF_f	**53.05**	51.60	52.76	**512.69**	12201.72	640.45	488.52	11371.22	612.01	24.17	830.49	28.45
RBF_m	50.67	50.04	**50.96**	**521.68**	5915.54	651.81	499.72	5570.95	624.66	21.97	344.58	27.15
waveform	78.04	76.79	**79.14**	**929.01**	7248.51	1237.29	891.45	6835.17	1186.86	37.55	413.34	50.43
waveform_10	78.28	76.79	**79.12**	**946.79**	7141.90	1211.88	909.37	6724.86	1161.48	37.42	417.04	50.40
waveform_5	78.28	76.79	**79.12**	**949.20**	7198.39	1216.65	912.00	6782.25	1164.04	37.20	416.14	52.61

For the LED and waveform datasets, the number after the _ represents the number of attributes with drift. For the RBF dataset, the f represents fast drift (speed of 0.01) and m moderate drift (speed of 0.001). Algorithms: VFDT, VFDT-*nmin*, and CVFDT. Measurements: accuracy, total energy, CPU energy, DRAM energy. Total energy = CPU energy + DRAM energy. Higher accuracy and lower total energy consumption values for each dataset are presented in bold

Table 6 Energy consumption and accuracy results of real-world datasets

Dataset	Accuracy (%)			Total energy (J)			CPU energy (J)			DRAM energy (J)		
	vfdt-nmin	cvfdt	vfdt	vfdt-nmin	cvfdt	vfdt	vfdt-nmin	cvfdt	vfdt	vfdt-nmin	cvfdt	vfdt
CICIDS17	**98.11**	96.02	97.94	**1041.62**	10252.09	1081.70	1008.60	9617.72	1046.31	33.02	634.37	35.39
Airline	**67.01**	55.04	66.94	**89.47**	709.53	100.74	85.69	661.07	96.31	3.78	48.46	4.43
Electricity	**76.23**	71.13	72.91	**7.24**	31.56	10.58	7.00	29.71	10.14	0.23	1.85	0.44
Forest	**59.99**	56.80	59.79	**341.77**	2054.04	350.42	330.97	1964.07	339.08	10.81	89.97	11.33
kddcup	**87.03**	47.41	**87.03**	**3698.26**	21497.73	3926.66	3572.08	20729.69	3796.82	126.18	768.04	129.84
Poker	**75.44**	58.46	72.27	142.45	807.87	**130.91**	136.56	767.53	124.85	5.89	40.34	6.06

Algorithms: VFDT, VFDT-*nmin*, and CVFDT. Measurements: accuracy, total energy, CPU energy, DRAM energy. Total energy = CPU energy + DRAM energy. Higher accuracy and lower total energy consumption values for each dataset are presented in bold

If the p value, obtained as a result of the test, is lower than the chosen alpha level (0.01 for this paper), we can conclude that the VFDT-*nmin* obtains significantly higher accuracy than the VFDT.

We perform the same statistical tests to check whether the difference in energy consumption between the VFDT and the VFDT-*nmin* is statistically significant. After testing for normality, and obtaining a p value of less than 0.01, we choose to perform the same nonparametric test as before, Wilcoxon signed-rank test. The null and alternative hypothesis is the following:

$\boldsymbol{H_0}$: $\mu_{E1} = \mu_{E2}$, where μ_{E1} represents the mean of energy consumption values for the VFDT, and μ_{E2} represents the mean of energy consumption values for VFDT-*nmin*. Thus, the null hypothesis states that the VFDT and the VFDT-*nmin* consume equal amounts of energy.

$\boldsymbol{H_1}$: $\mu_{E1} > \mu_{E2}$, stating that the mean of the energy consumed by the VFDT is higher than the mean of the energy consumed by the VFDT-*nmin*. Since we employ directional alternative hypothesis, the null hypothesis can only be rejected if the data indicate that the mean of the energy consumed by the VFDT is significantly higher than the mean of the energy consumed by the VFDT-*nmin*.

6 Results and discussion

The results of the experiments are divided in the three setups defined above. Tables 4, 5, and 6 present the accuracy and energy consumption results of the *baseline, concept drift,* and *real datasets* setups, respectively. We have evaluated the accuracy as the percentage of correctly classified instances and the energy consumption as the energy estimated by the Intel Power Gadget tool, summing the energy consumed by the processor and the DRAM to obtain the total energy consumption. We have run the experiments 5 times and averaged the results. Table 2 summarizes the energy and accuracy results from all datasets by showing the difference between VFDT-*nmin* and VFDT and between VFDT-*nmin* and CVFDT.

We initially discuss the statistical significance results performed on the accuracy and energy consumption data from the VFDT and the VFDT-*nmin*. We then discuss the difference in energy consumption and accuracy from the baseline datasets between VFDT, VFDT-*nmin*, and CVFDT. The aim is to have a general understanding of the algorithm in terms of energy consumption, and how the number of nominal and numerical attributes affect it. We further compare how concept drift affects both accuracy and energy consumption in different datasets with different levels of concept drift. Finally we compare how well the *nmin adaptation* method works in real-world datasets, in terms of accuracy and energy consumption.

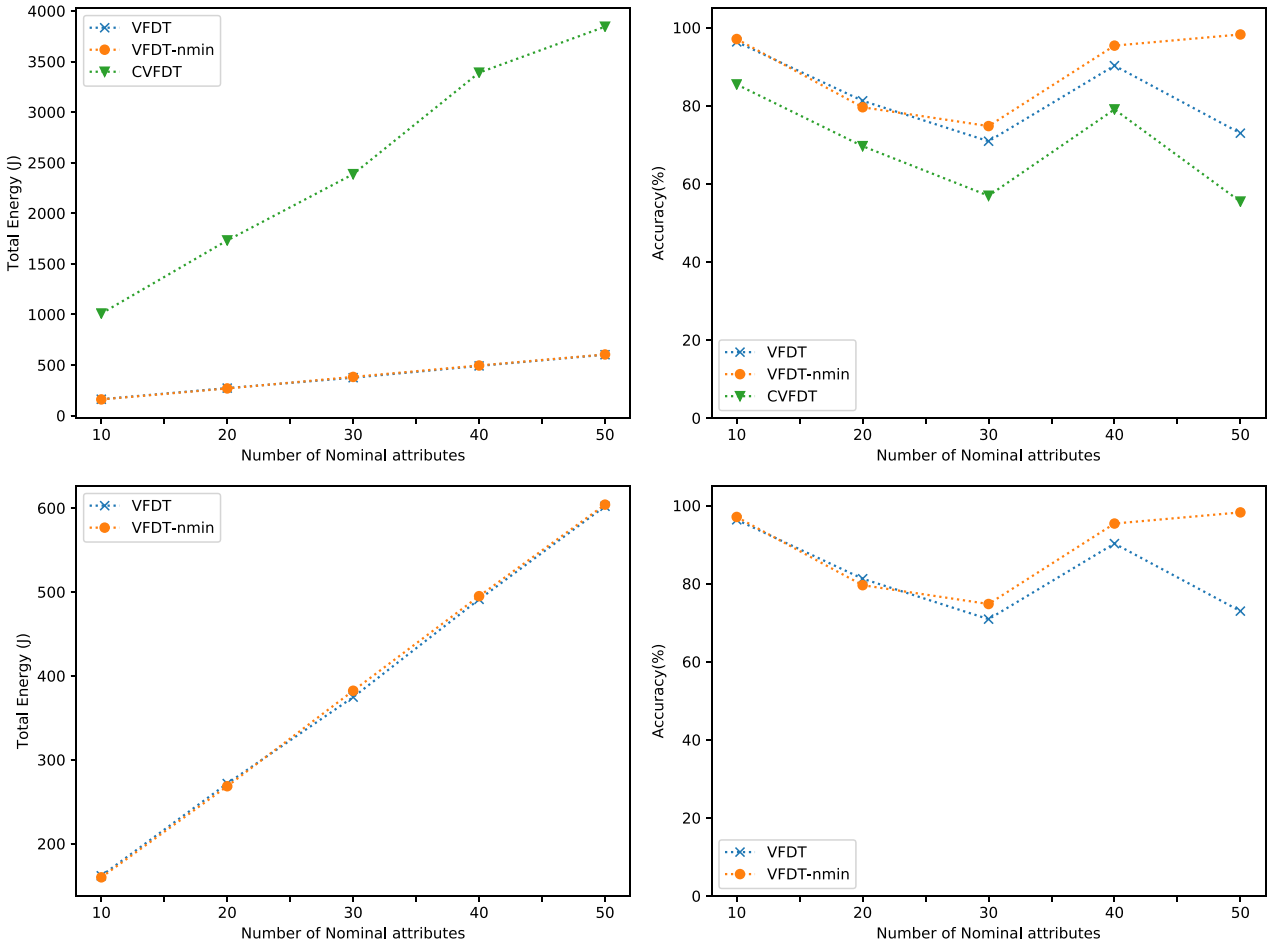

Fig. 4 Baseline setup. Comparison on energy consumption and accuracy of the number of nominal attributes for the VFDT-*nmin*, VFDT, and CVFDT. The lower figures are expanded version of the upper figures (without CVFDT)

6.1 Statistical characterization

This section presents the results of the statistical significance analysis between the VFDT and the VFDT-*nmin*. We omit the comparison against the CVFDT since the differences are clear, and CVFDT obtains significantly higher energy consumption on all datasets, and worse accuracy on all except one.

We have performed a Wilcoxon signed-rank test on the data, with a confidence level of 0.01. The p values of the tests are presented in Table 3.

The results show that there is a statistically significant difference between the accuracy of the VFDT and the VFDT-*nmin*, since the p value is lower than 0.01. Thus, the null hypothesis, which stated that the means of the accuracy of both algorithms are equal (Sect. 5.3), is rejected. The results also show that there is a statistical difference between the energy consumption values between the VFDT and the VFDT-*nmin*. The null hypothesis can be rejected, since the p value (2.56×10^{-6}) is lower than 0.01. Thus, the alternative hypothesis is supported, which stated that the energy consumed by the VFDT-*nmin* is significantly lower than the energy consumed by the VFDT.

6.2 Baseline setup

The baseline setup (Table 4) is done by varying the number of instances, nominal, and numerical attributes from the random tree dataset. Figure 4 compares the amount of accuracy and energy consumed when the number of nominal attributes is varied between 10, 20, 30, 40, and 50. We observe that the VFDT and VFDT-*nmin* behave very similarly, i.e., the higher the number of attributes, the higher the energy consumption. However, the CVFDT has a significantly higher increase in energy consumption when increasing the number of attributes. Based on this analysis, we can conclude that the CVFDT does not scale with the number of attributes. Figure 4 also shows the accuracy per number of nominal attributes, where we noticed a strange phenomenon. VFDT-*nmin* obtains significantly higher levels of accuracy than the

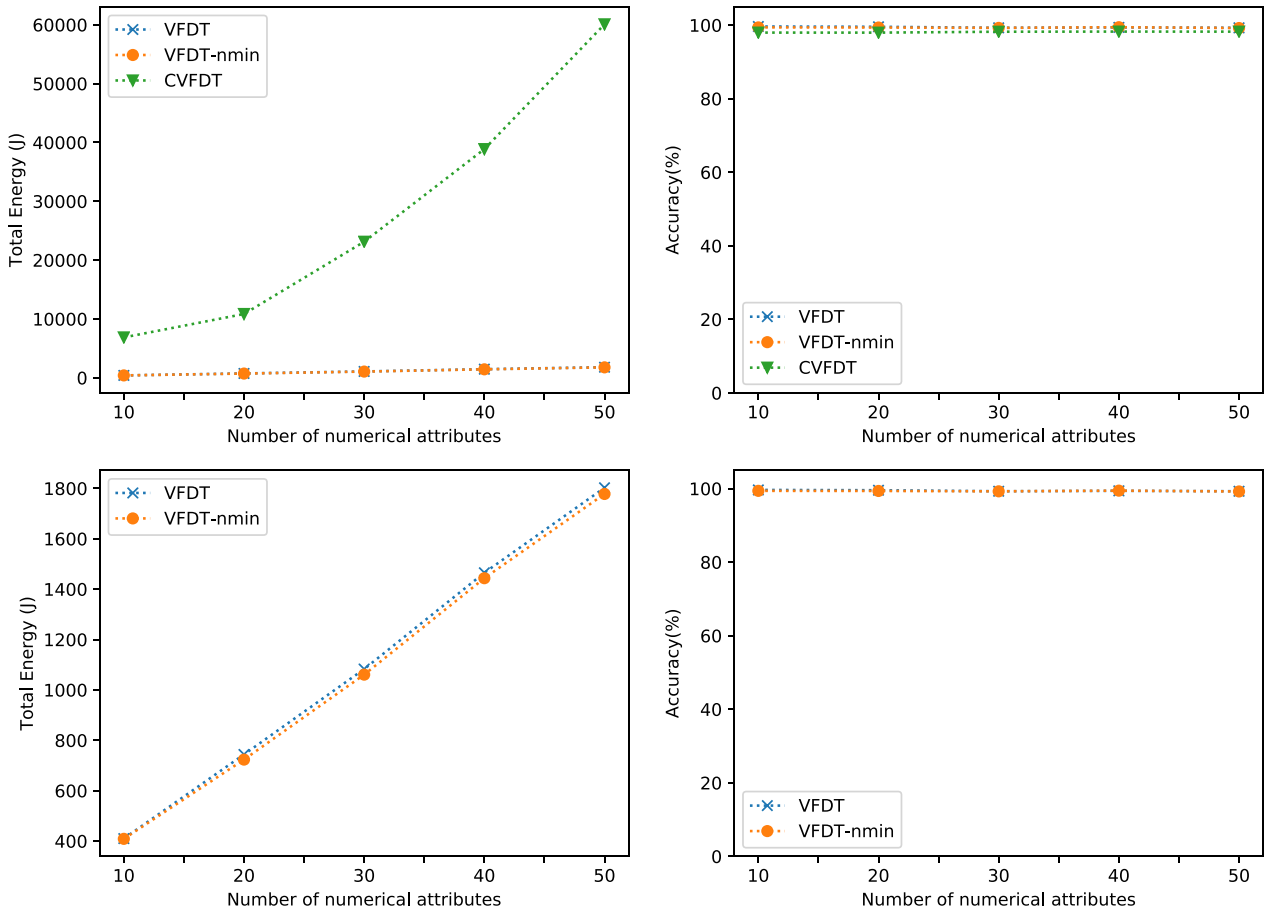

Fig. 5 Baseline setup. Comparison on energy consumption and accuracy of the number of numerical attributes for the VFDT-*nmin*, VFDT, and CVFDT. The lower figures are expanded version of the upper figures (without CVFDT)

VFDT. This is visible especially for the dataset with 50 nominal attributes. We believe that this may be due to some error with the code that was provided in the original implementation. We have tested the same dataset on the new version from MOA [2], and it has not shown such behavior.

Figure 5 shows the same comparison as Fig. 4 but for numerical attributes. We can see a similar pattern in terms of energy consumption. Namely, the higher the number of numerical attributes the higher the energy consumption. While VFDT-*nmin* and VFDT show a linear increase, CVFDT shows an exponential increase. The latter confirms our previous claim that it is nonscalable. However, in terms of accuracy, all algorithms behave as expected. The accuracy is almost identical, being independent of the number of attributes. Figure 6 compares the energy consumption of the nominal and numerical attributes by averaging all algorithms (VFDT, VFDT-*nmin*, and CVFDT). The figures empirically validate what we theoretically claimed with the energy model in Sect. 4.3. Namely, that handling numerical attributes results in a higher energy cost compared to handling nominal attributes. We can observe that handling 50 numerical attributes costs 12.6X more than handling 50 nominal attributes (averaged for VFDT-*nmin*, VFDT, and CVFDT). This matches with the results from the VFDT energy model (Sect. 4.2), where the numerical attributes require handling floating point operations, which are more energy consuming than integer operations.

We have studied the energy consumption and accuracy of varying the number of instances between 10k, 100k, 1M, and 10M. Figure 7 shows that there is not a linear increase between the number of instances and the energy consumption. When the number of instances reaches 10M, the energy consumption increases by 18× (compared to 1M), for the VFDT-*nmin* and VFDT, and by 12× for the CVFDT. The accuracy increases with the number of instances, which is the expected behavior, since with more instances the tree is able to reflect better the dataset.

From Table 4 we can observe that the VFDT-*nmin* obtained the lowest energy consumption in 11 out of 14 datasets. This is achieved while obtaining either the highest

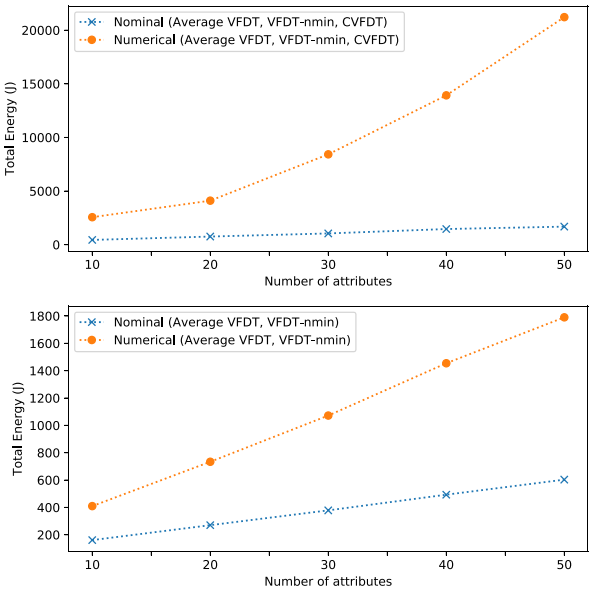

Fig. 6 Baseline setup. Comparison of the nominal and numerical attributes on energy consumption and accuracy for the VFDT-*nmin*, VFDT, and CVFDT. The lower figures are expanded version of the upper figures (without CVFDT)

accuracy or up to a 1.65% lower compared to the algorithm with the highest accuracy. The CVFDT obtains the highest energy consumption values for all datasets, and it only obtains the highest accuracy in one of the datasets. We conclude that, for the baseline datasets, the VFDT-*nmin* obtains significantly lower energy at a cost of less than 1% of accuracy on average.

6.3 Concept drift setup

Figures 8 and 9 show the energy consumption and accuracy results of running the VFDT, VFDT-*nmin*, and CVFDT algorithms on datasets with different levels of concept drift (see also Table 5). Figure 9 focuses on VFDT and VFDT-*nmin* for a clearer view, since CVFDT has high levels of energy consumption and does not allow to view the difference in energy between VFDT and VFDT-*nmin*. As we already observed with the baseline setup, CVFDT consumes significantly more energy than VFDT and VFDT-*nmin*. However, in terms of accuracy, it obtains a lower accuracy in all of the datasets with concept drift. This behavior is not expected, since the CVFDT is supposed to handle concept drift, but even in datasets with a high level of drift, such as RBF fast, it obtains lower accuracy than both VFDT and VFDT-*nmin*. Taking a look at Fig. 9, we observe how VFDT-*nmin* obtains significantly lower energy consumption than VFDT for the RBF and waveform datasets. On the other hand, VFDT-*nmin* obtains slightly higher energy consumption for the LED dataset, but it seems to scale better when increasing the number of attributes with drift. VFDT-*nmin* decreases energy consumption when instead of having 3 attributes with drift we have 7 (higher drift), while VFDT increases in energy consumption. This phenomenon is also visible for the RBF dataset, where VFDT-*nmin* scales in terms of energy consumption when increasing drift, and VFDT does not.

Moreover, all three algorithms decrease accuracy for the RBF dataset when increasing the amount of drift, although there is a slight increase when moving from moderate to fast drift. The LED and waveform dataset give almost the same accuracy when varying the amount of drift. The reason for this is that the algorithm is able to learn the data even with drift, and there is no possibility for a higher accuracy in these datasets (with this type of algorithms). VFDT-*nmin* obtains higher accuracy for the LED dataset independently on the amount of drift, but at a cost of higher energy consumption. All in all, VFDT-*nmin* obtains significantly lower energy consumption on 6/9 datasets. Moreover, VFDT-*nmin* scales better than VFDT in terms of energy consumption when increasing the amount of drift or the number of attributes with drift. VFDT and VFDT-*nmin* obtain very similar levels of accuracy (less than 1% difference in average), while VFDT-*nmin* even obtains a higher accuracy in 3/9 datasets. As for other the other datasets, CVFDT obtains significantly higher energy consumption on all drift datasets while not giving improvements in accuracy.

6.4 Real-world setup

This last section describes the energy consumption and accuracy results of running VFDT, VFDT-*nmin*, and CVFDT in six real-world datasets (explained in Sect. 5.1). As we can see from Table 6, VFDT-*nmin* obtains the highest accuracy on all datasets and the lowest energy consumption on 5/6 datasets. The differences in accuracy between VFDT and VFDT-*nmin* are in average of 1.16%. For the electricity dataset, VFDT-*nmin* obtains the maximum accuracy gain, of 3.3% compared to VFDT. Similar to the previous setups, CVFDT obtains significantly lower levels of accuracy on all datasets at a significantly higher energy cost.

This is clear from Figs. 10, 11, and 12. Figure 12 restricts the results to the VFDT and VFDT-*nmin* for a clear view. Figure 10 shows how there is not a clear direct connection between a higher energy consumption and a higher accuracy. Taking a look at the forest dataset, for instance, we see how this dataset consumes more energy than the electricity, airline, and poker datasets, but it obtains a lower accuracy than all of them. KDDcup is the dataset with the highest energy consumption because it is also the dataset with the highest number of instances. In relation to how the number of attributes impact the energy consumption, we observe how CICIDS obtains significantly higher energy consumption than the poker dataset, while poker has more number of

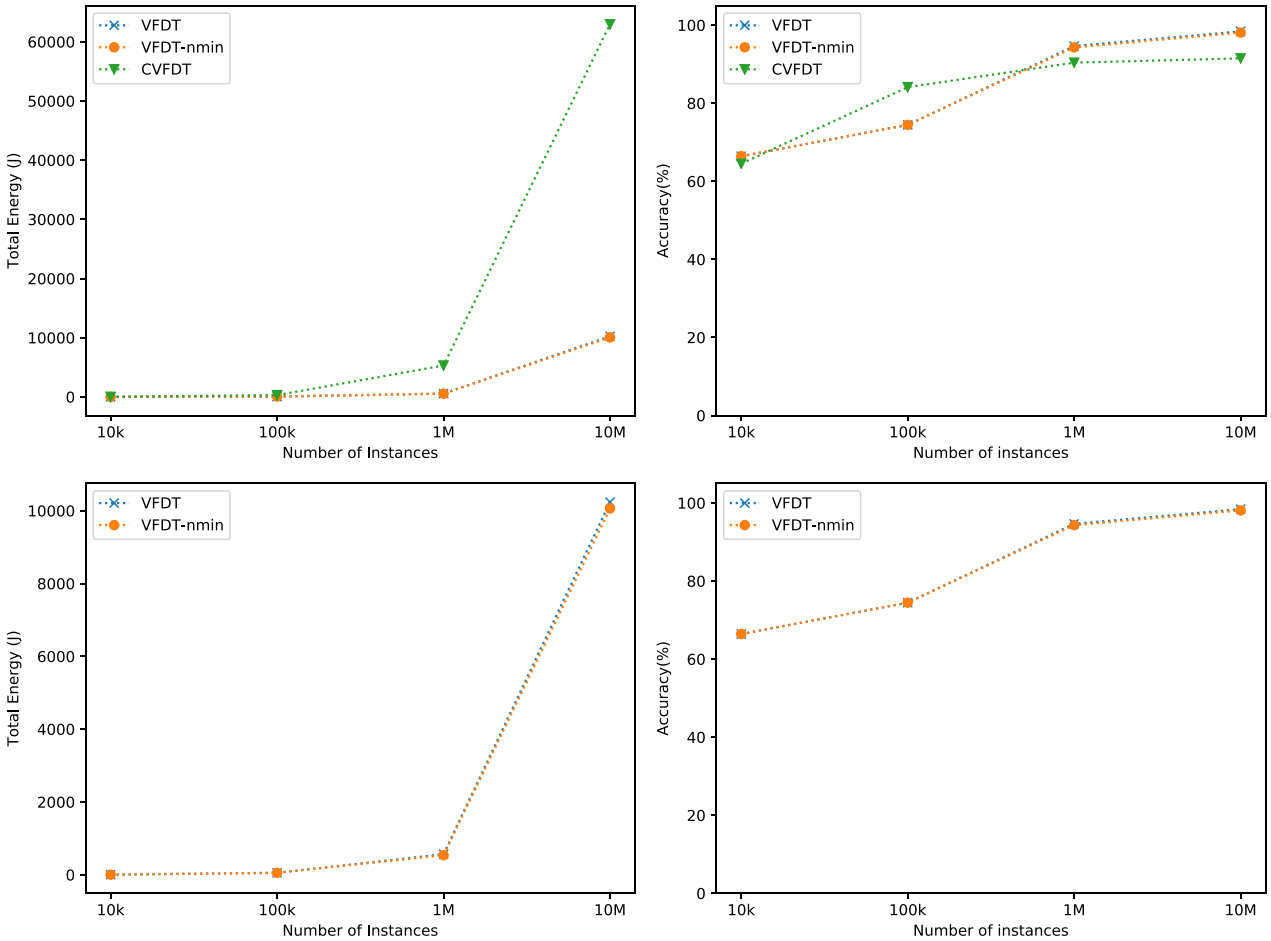

Fig. 7 Baseline setup. Comparison on energy consumption and accuracy of the number of instances for the VFDT-*nmin*, VFDT, and CVFDT. The lower figures are expanded version of the upper figures (without CVFDT)

instances than CICIDS. The reason is because CICIDS has 73 more attributes than poker.

6.5 Summary

The VFDT-*nmin* outperforms VFDT in terms of energy consumption in the majority of real-world and concept drift datasets, irrespectively of the amount of instances and numerical attributes. VFDT-*nmin* consumes more energy than the VFDT in the poker and the LED dataset, both datasets containing 10 different classes. However, the VFDT-*nmin* consumes less energy than the VFDT for the kddcup dataset, which has 23 different classes. Thus, VFDT-*nmin* energy performance compared to the VFDT is not correlated with the number of classes. VFDT-*nmin* consumes less energy than the VFDT on the baseline setup for all instances and numerical attributes datasets. The results of the paper are summarized below:

- VFDT-*nmin* obtains significantly lower levels of energy consumption in comparison with VFDT for 22 out of the 29 studied datasets (76% of the datasets).
- VFDT and VFDT-*nmin* obtain on average less than 1% difference in accuracy. VFDT-*nmin* obtains higher accuracy than VFDT on 55% of the datasets.
- CVFDT obtains significantly higher energy consumption and lower accuracy in almost all datasets, even on concept drift datasets.
- Regarding the baseline setup, we can see how handling numerical attributes costs up to 12X more energy than handling nominal attributes. This matches with our theoretical claims from the energy model (Sect. 4.3). The reason for this difference in energy is because the average energy per floating point operation (E_{FPU}) is significantly higher than the average energy per integer instruction (E_{INT}), as explained by [21].
- We also observed that while VFDT and VFDT-*nmin* scale linearly in terms of energy consumption when increasing

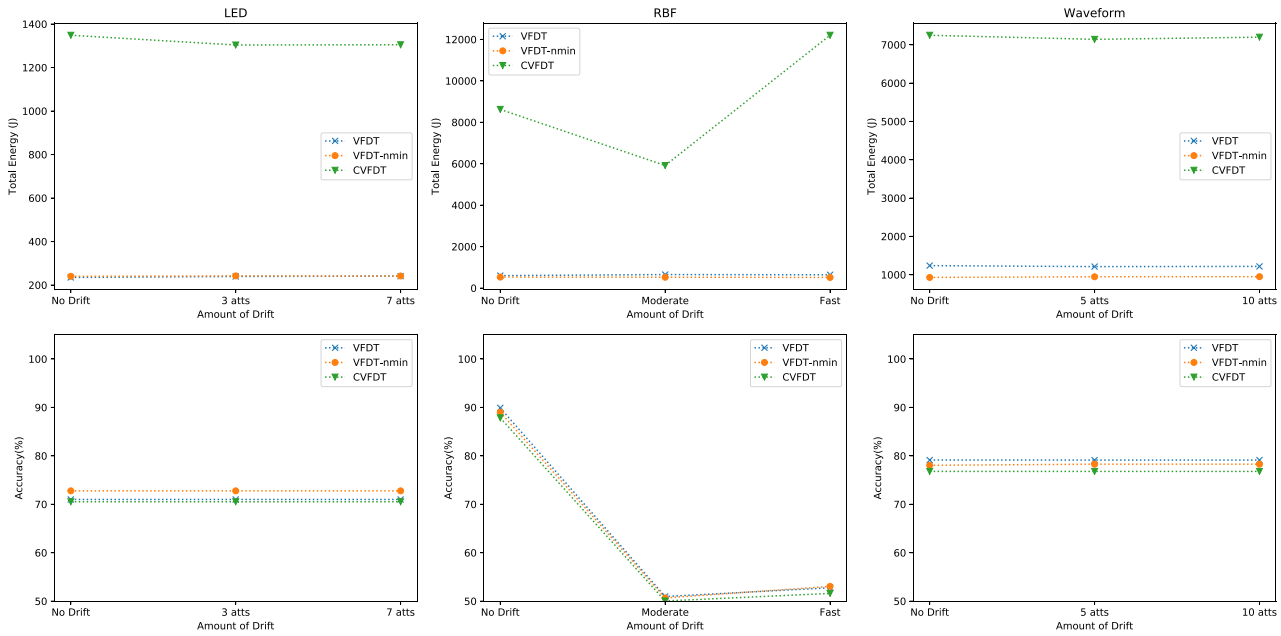

Fig. 8 Concept drift setup. Comparison on energy consumption and accuracy for the LED, RBF, and waveform datasets for the VFDT-*nmin*, VFDT, and CVFDT

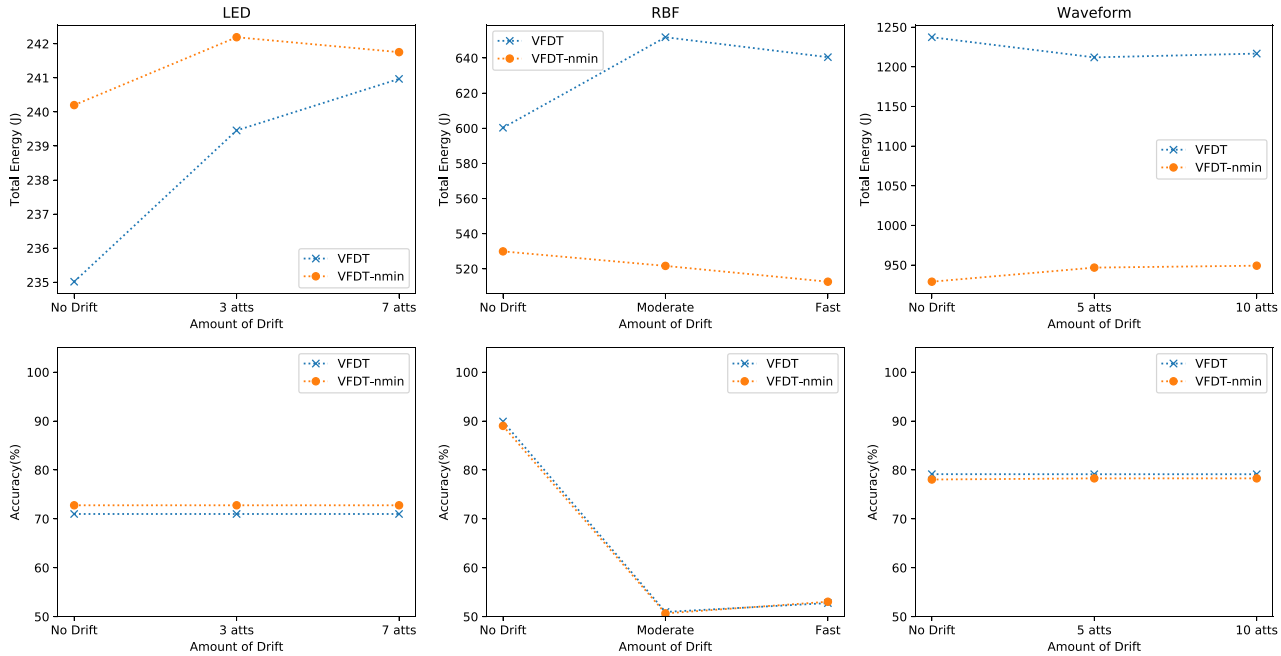

Fig. 9 Concept drift setup. Comparison on energy consumption and accuracy for the LED, RBF, and waveform datasets for the VFDT-*nmin*, and VFDT

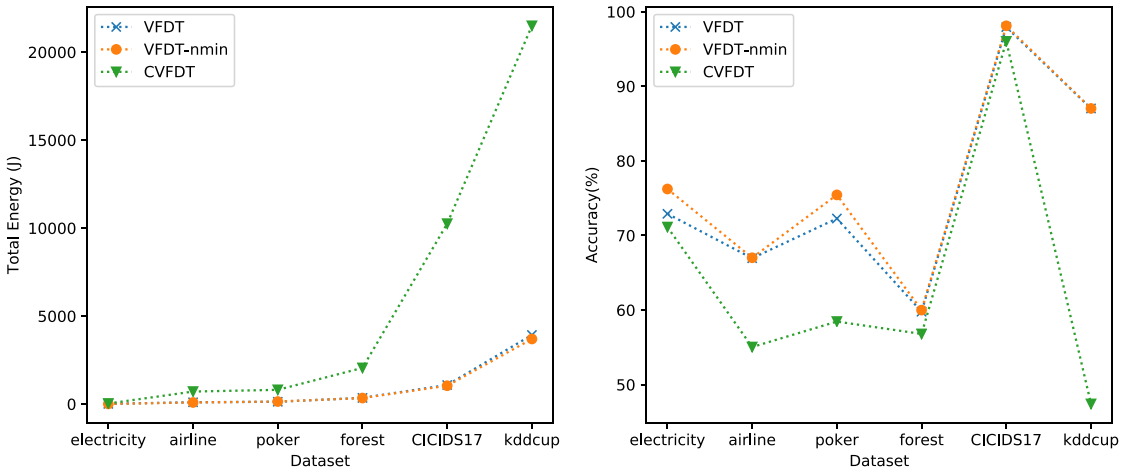

Fig. 10 Real-world setup. Results of running the VFDT, VFDT-*nmin*, and CVFDT on real-world datasets. The datasets are sorted by the energy consumption (ascending) of the VFDT-*nmin*

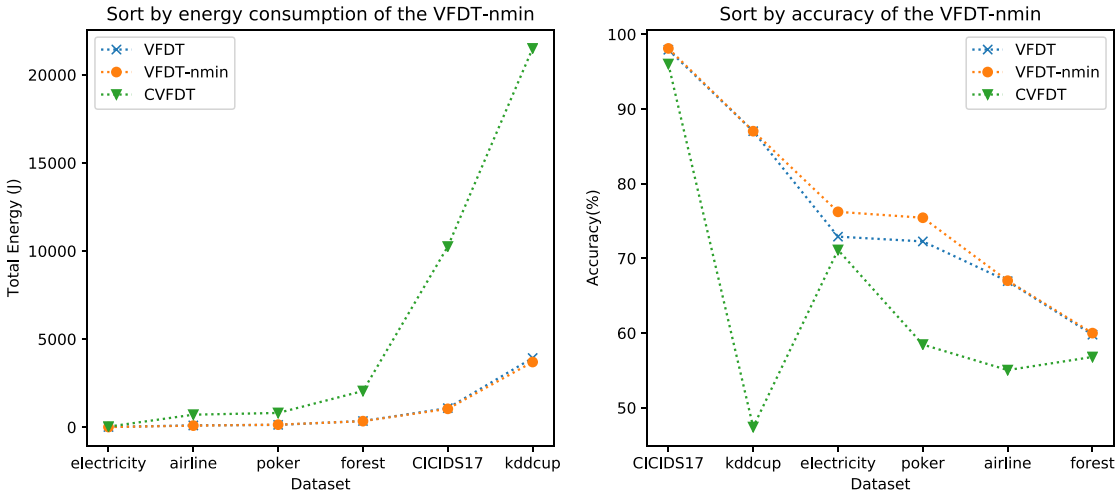

Fig. 11 Real-world setup. Results of running the VFDT, VFDT-*nmin*, and CVFDT on real-world datasets. The datasets are sorted by the energy consumption (ascending) of the VFDT-*nmin* on the left plot and by the accuracy (descending) of the VFDT-*nmin* on the right plot

the number of attributes, CVFDT scales poorly with the increase of both number of attributes and instances.

– Regarding concept drift, we observed that VFDT-*nmin* scales better than VFDT in terms of energy consumption when increasing the amount of drift or the number of attributes with drift.
– VFDT-*nmin* obtained a higher accuracy compared to VFDT for all real-world datasets, obtaining a significantly lower energy energy consumption in 83% of the datasets. We also observed how there was not a direct correlation between a higher energy consumption needed to obtain a higher accuracy. This was explained simply by the size and the dimension (number of attributes) of the dataset.

7 Limitations

There are a few limitations worth mentioning in this study. Regarding the *nmin adaptation* method, it adapts to the optimal nmin parameter making the assumption that the future observed data will follow the same distribution as the already observed data. Regarding the implementation and code used, we are aware that there might be a bug in the original implementation of the VFDT that is the one used in this study. This was mentioned in the baseline setup, since we obtained unexpected results for the datasets with high number of nominal attributes. We are going to use MOA, which has the newer implementation of the code, in future studies.

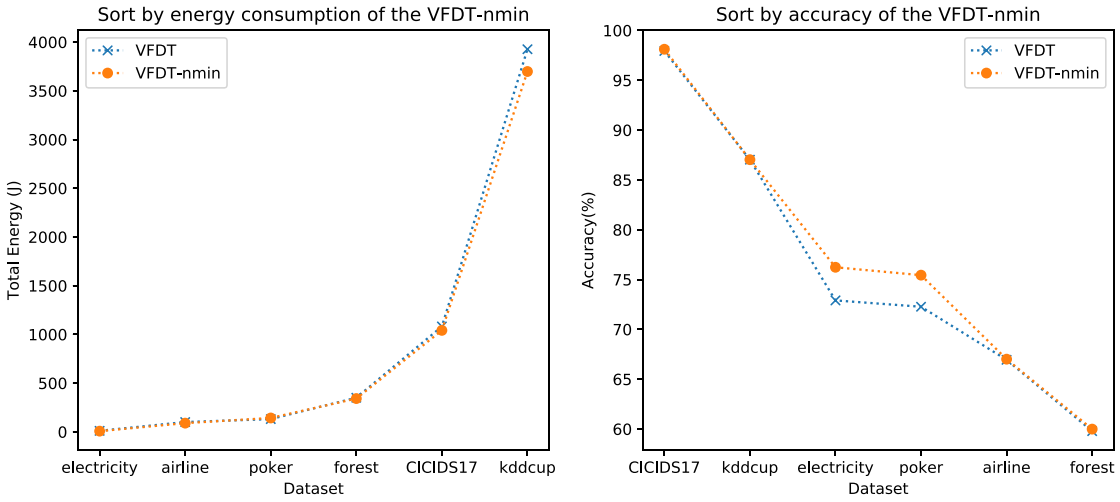

Fig. 12 Real-world setup. Results of running the VFDT, VFDT-*nmin*, on real-world datasets. The datasets are sorted by the energy consumption of the VFDT-*nmin* on the left plot and by the accuracy of the VFDT-*nmin* on the right plot

Regarding the energy consumption estimation, the energy consumption values are not given per program, but for the complete system (DRAM or CPU). Thus, we base our conclusions on the comparison between the algorithms and setups, since all of them are run on the same scenarios. Since estimating energy consumption in software is challenging [17], we have created the energy model presented in Sects. 4.2 and 4.3, which is independent of the programming language and hardware platform. We understand that this is a simplified model of all the events present in the algorithm. However, it is a straightforward approach to understand, from a theoretical perspective, which parts of the algorithm are the most energy inefficient. More details on different approaches to estimate energy are giving in the already mentioned survey [17].

8 Conclusions

This paper introduced *nmin adaptation*, a method that extends standard Hoeffding trees to reduce their energy consumption. *nmin adaptation* allows for a faster growth on the branches with higher confidence to split and delays the growth on the less confident branches. This reduces unnecessary computations, reducing energy consumption with only minor effects on accuracy.

This paper extends our previous work: "Hoeffding Trees with *nmin adaptation*" by adding extensive experiments that validate the proposed method with statistical tests. We have evaluated the accuracy and energy consumption of the VFDT, VFDT-*nmin* (VFDT with *nmin adaptation*), and CVFDT algorithms, under 29 public datasets. The results show that VDFT-*nmin* consumes up to 31% less energy, affecting accuracy at most by a 1.65%, in comparison with the standard VFDT. In comparison with CVFDT, VFDT-*nmin* consumes 85% less energy, obtaining 6% higher accuracy values, on average.

In particular, we have first conducted a sensitivity analysis to determine the energy consumption and accuracy of the mentioned algorithms when varying the number of instances, nominal, and numerical attributes. The results of this analysis show that handling numerical attributes can consume up to 12X more energy than handling nominal attributes. We then compared VFDT, VFDT-*nmin*, and CVFDT on concept drift datasets. The results of this setup conclude that VFDT-*nmin* consumes significantly less energy consumption than the other two algorithms (13% on average compared to the VFDT, 87% compared to the CVFDT), while obtaining similar levels of accuracy. We finally compared the mentioned algorithms in six real-world datasets. VFDT-*nmin* obtained higher accuracy than VDFT and CVFDT, while obtaining 7% less energy consumption than the VFDT and 84% less energy consumption than the CVFDT.

Low energy consumption is one of the key requirements for algorithms to be able to run in the edge (e.g., mobile and embedded devices). While data stream mining algorithms are designed to run in the edge, due to their high velocity and low memory usage, they still have not considered energy consumption. To address that gap, we present a method that allows for an energy-efficient approach to design Hoeffding trees, without affecting its predictive performance.

References

1. Bifet, A., Gavaldà, R.: Adaptive learning from evolving data streams. In: International Symposium on Intelligent Data Analysis, Springer, pp. 249–260 (2009)
2. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: massive online analysis. J. Mach. Learn. Res. **11**, 1601–1604 (2010)
3. Bifet, A., Zhang, J., Fan, W., He, C., Zhang, J., Qian, J., Holmes, G., Pfahringer, B.: Extremely fast decision tree mining for evolving data streams. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, pp. 1733–1742 (2017)
4. Breiman, L.: Classification and Regression Trees. Routledge, Boca Raton (2017)
5. Carcillo, F., Le Borgne, Y.A., Caelen, O., Bontempi, G.: Streaming active learning strategies for real-life credit card fraud detection: assessment and visualization. Int. J. Data Sci. Anal. **5**(4), 285–300 (2018). https://doi.org/10.1007/s41060-018-0116-z
6. Center CRC 876 at Technical University of Dortmund CR: Resource-aware machine learning. (2017). Retrieved from http://sfb876.tu-dortmund.de/SPP/index.html, online; accesed 1 August 2018
7. Chen, Y., Alspaugh, S., Borthakur, D., Katz, R.: Energy efficiency for large-scale mapreduce workloads with significant interactive analysis. In: Proceedings 7th European Conference on Computer Systems, pp. 43–56 (2012)
8. da Costa, V.G.T., de Leon Ferreira de Carvalho, A.C.P., Junior, S.B.: Strict very fast decision tree: a memory conservative algorithm for data stream mining. Pattern Recognit. Lett. **116**, 22–28 (2018). https://doi.org/10.1016/j.patrec.2018.09.004
9. David, H., Gorbatov, E., Hanebutte, UR., Khanna, R., Le, C.: Rapl: memory power estimation and capping. In: Proceedings of the 16th ACM/IEEE International Symposium on Low Power Electronics and Design, pp. 189–194 (2010)
10. Domingos, P., Hulten, G.: Mining high-speed data streams. In: Proceedings of 6th SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 71–80 (2000)
11. Dubois, M., Annavaram, M., Stenström, P.: Parallel Computer Organization and Design. Cambridge University Press, Cambridge (2012)
12. Evans, R., Gao, J.: DeepMind Reduces Google Data Centre Cooling Bill by 40%. (2016). Retrieved from https://deepmind.com/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-40/, online; accessed 1 August (2018)
13. Gaber, MM., Zaslavsky, A., Krishnaswamy, S.: Towards an adaptive approach for mining data streams in resource constrained environments. In: Kambayashi Y, Mohania M, Wöß W (eds) "Data Warehousing and Knowl. Discovery: 6th International Conference, Zaragoza, Spain, Sept. 1–3.", Springer, pp. 189–198 (2004)
14. Gaber, M.M., Krishnaswamy, S., Zaslavsky, A.: On-board Mining of Data Streams in Sensor Networks. Advanced Methods for Knowledge Discovery from Complex Data, pp. 307–335. Springer, Berlin (2005)
15. Garcia-Martin, E., Lavesson, N., Grahn, H.: Identification of energy hotspots: a case study of the Very Fast Decision Tree. In: Green, Pervasive, and Cloud Computing: 12th International Conference, GPC 2017, Cetara, Italy, May 11–14, 2017, Springer International Publishing, vol. 10232, pp. 267–281 (2017)
16. García-Martín, E., Lavesson, N., Grahn, H., Casalicchio, E., Boeva, V.: Hoeffding trees with nmin adaptation. In: 2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA), IEEE, pp. 70–79 (2018)
17. García-Martín, E., Lavesson, N., Grahn, H., Casalicchio, E., Boeva, V.: How to measure energy consumption in machine learning algorithms. In: Alzate, C., Monreale, A., Assem, H., Bifet, A., Buda, T.S., Caglayan, B., Drury, B., García-Martín, E., Gavaldà, R., Koprinska, I., Kramer, S., Lavesson, N., Madden, M., Molloy, I., Nicolae, M.I., Sinn, M. (eds.) ECML PKDD 2018 Workshops, pp. 243–255. Springer International Publishing, Cham (2019)
18. Gauen, K., Rangan, R., Mohan, A., Lu, YH., Liu, W., Berg, AC.: Low-power image recognition challenge. In: Design Automation Conference (ASP-DAC), 2017 22nd Asia and South Pacific, IEEE, pp. 99–104 (2017)
19. Harries, M.: Splice-2 comparative evaluation: electricity pricing. Technical Report (1999)
20. Hoeffding, W.: Probability inequalities for sums of bounded random variables. J. Am. Stat. Assoc. **58**(301), 13–30 (1963)
21. Horowitz, M.: 1.1 computing's energy problem (and what we can do about it). In: Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International, IEEE, pp. 10–14 (2014)
22. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, pp. 97–106 (2001)
23. Ikonomovska, E.: Datasets. (2013). Retrieved from http://kt.ijs.si/elena_ikonomovska/data.html, online; Accessed 1 August (2018)
24. Koomey, JG., Berard, S., Sanchez, M., Wong, H.: Assessing trends in the electrical efficiency of computation over time. In: IEEE Annals of the History of Computing 17 (2009)
25. Korb, I., Kotthaus, H., Marwedel, P.: mmapcopy: Efficient memory footprint reduction using application-knowledge. In: SAC 2016 31st ACM Symposium on Applied Computing, pp. 1832–1837 (2016)
26. Kourtellis, N., Morales, GDF., Bifet, A., Murdopo, A.: Vht: Vertical hoeffding tree. In: Big Data (Big Data), 2016 IEEE International Conference on, IEEE, pp. 915–922 (2016)
27. Losing, V., Hammer, B., Wersing, H.: KNN classifier with self adjusting memory for heterogeneous concept drift. In: IEEE 16th International Conference on Data Mining (ICDM), pp. 291–300 (2016)
28. Marrón, D., Ayguadé, E., Herrero, JR., Read, J., Bifet, A.: Low-latency multi-threaded ensemble learning for dynamic big data streams. In: Big Data (Big Data), 2017 IEEE International Conference on, IEEE, pp. 223–232 (2017)
29. Mazouz, A., 0001 DCW, Kuck, DJ., Jalby, W.: (2017) An incremental methodology for energy measurement and modeling. In: ICPE, pp. 15–26
30. Meng, T., Yuan, B.: Parallel edge-based visual assessment of cluster tendency on gpu. Int. J. Data Sci. Anal. **6**(4), 287–295 (2018). https://doi.org/10.1007/s41060-018-0100-7
31. Moniz, N., Branco, P., Torgo, L.: Resampling strategies for imbalanced time series forecasting. Int. J. Data Sci. Anal. **3**(3), 161–181 (2017). https://doi.org/10.1007/s41060-017-0044-3
32. Reams, C.: Modelling energy efficiency for computation. PhD thesis, University of Cambridge, Computer Laboratory (2012)
33. Rhoden, B., Klues, K., Zhu, D., Brewer, E.: Improving per-node efficiency in the datacenter with new os abstractions. In: Proceedings of the 2nd ACM Symposium on Cloud Computing, p. 25 (2011)
34. Rodrigues, C.F., Riley, G., Luján, M.: (2017) Fine-grained energy profiling for deep convolutional neural networks on the jetson tx1.

Workload characterization (IISWC). In: IEEE International Symposium on, IEEE, pp. 114–115 (2017)
35. Ruth, S.: Green it more than a three percent solution? IEEE Internet Comput. **13**(4), 74–78 (2009)
36. Sharafaldin, I., Lashkari, AH., Ghorbani, AA.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: ICISSP, pp. 108–116 (2018)
37. Shehabi, A., Smith, S., Sartor, D., Brown, R., Herrlin, M., Koomey, J., Masanet, E., Horner, N., Azevedo, I., Lintner, W.: United states data center energy usage report. Technical Report, Lawrence Berkeley National Laboratory, Berkeley, California (2016)
38. Sheskin, D.J.: Handbook of Parametric and Nonparametric Statistical Procedures. CRC Press, Boca Raton (2003)
39. Stolfo, J., Fan, W., Lee, W., Prodromidis, A., Chan, PK.: Cost-based modeling and evaluation for data mining with application to fraud and intrusion detection. In: Results from the JAM Project by Salvatore pp. 1–15 (2000)
40. Weste, N., Harris, D., Banerjee, A.: Cmos vlsi design. Circuits Syst. Perspect. **11**, 739 (2005)
41. Wilcoxon, F.: Individual comparisons by ranking methods. Biom. Bull. **1**(6), 80–83 (1945)
42. Yang, T.J., Chen, Y.H., Sze, V.: Designing energy-efficient convolutional neural networks using energy-aware pruning. In: CVPR 2017, Conference on Computer Vision and Pattern Recognition, Hawaii Convention Center, July, pp 21–26. Honolulu, Hawaii, USA (2017)