Image classification on IoT edge devices: profiling and modeling

Salma Abdel Magid[1] · Francesco Petrini[1] · Behnam Dezfouli[1]

Abstract

With the advent of powerful, low-cost IoT systems, processing data closer to where the data originates, known as edge computing, has become an increasingly viable option. In addition to lowering the cost of networking infrastructures, edge computing reduces edge-cloud delay, which is essential for mission-critical applications. In this paper, we show the feasibility and study the performance of image classification using IoT devices. Specifically, we explore the relationships between various factors of image classification algorithms that may affect energy consumption, such as dataset size, image resolution, algorithm type, algorithm phase, and device hardware. In order to provide a means of predicting the energy consumption of an edge device performing image classification, we investigate the usage of three machine learning algorithms using the data generated from our experiments. The performance as well as the trade-offs for using linear regression, Gaussian process, and random forests are discussed and validated. Our results indicate that the random forest model outperforms the two former algorithms, with an R-squared value of 0.95 and 0.79 for two different validation datasets. The random forest also served as a feature extraction mechanism which enabled us to identify which predictor variables influenced our model the most.

Keywords Edge and fog computing · Machine learning · Energy efficiency · Accuracy

1 Introduction

Researchers at Gartner estimate that there will be 20 billion IoT devices connected to the Internet by 2020 [1]. The burgeoning of such devices has sparked many efforts into researching the optimal device design. Since most IoT devices are constrained in terms of processing power and energy resources, the traditional approach has been to transmit data generated by the device to a cloud platform for server-based processing. Although cloud computing has been successfully employed, it is sometimes not desirable due to concerns about latency, connectivity, energy, privacy, and security [2–4].

To overcome these concerns, edge and fog computing have emerged. These architectures aim to push processing capabilities closer to the IoT devices themselves, which is specifically possible given their significant increase in processing power. For example, the archetype of modern IoT devices, the Raspberry Pi 3, offers a quad-core processor with 1 GB of RAM for only $30. The reduction in latency offered by utilizing such devices in edge and fog computing is critical to the success of applications such as object detection and image classification. These applications are used in mission-critical systems such as autonomous vehicles, surgical devices, security cameras, obstacle detection for the visually-impaired, rescue drones, and authentication systems [5–7]. However, these tasks consume a considerable amount of energy. Thus, it is especially important to understand the relationship between these algorithms and their respective energy consumption to efficiently utilize the IoT device's power resources. This is particularly important due to two reasons: First, many of these IoT devices work in a duty-cycled fashion. They are triggered when an external event happens, perform a

✉ Behnam Dezfouli
bdezfouli@scu.edu

Salma Abdel Magid
sabdelmagid@scu.edu

Francesco Petrini
fpetrini@scu.edu

1 Internet of Things Research Lab, Department of Computer Science and Engineering, Santa Clara University, Santa Clara, CA, USA

🖄 Springer

processing task, and transition to sleep mode once the task completes. A sample scenario is a security camera that captures an image when motion is detected. Another example could be a flood monitoring system that captures images of a river when the water level is beyond a certain threshold to detect the type of debris being carried by the water. Enhancing energy efficiency is essential for these types of applications, especially when they are battery powered or rely on energy harvesting technologies [8]. The second important motivation towards energy profiling and enhancement is to reduce carbon emissions. According to a study published by the Centre for Energy Efficient Telecommunications, the cloud was estimated to consume up to 43 TWh in 2015, compared to only 9.2 TWh in 2012, an increase of 460% [9]. This is roughly equivalent to adding 4.9 million cars to the roads. Given the dramatic impact of inefficient energy management, it has become important to ensure that the most intensive of tasks, especially image classification, are using the appropriate resources and minimizing their energy consumption footprint.

Various machine learning (ML) algorithms, offering different accuracy and complexity, have been proposed to tackle the challenges of image classification. Despite their exceptional accuracy, they require high processing power and large storage. For example, some state-of-the-art neural network architectures, such as AlexNet [10], GoogleNet [11], and ResNet [12] require over a million parameters to represent them and more than a billion multiply and accumulate computations (MAC) [13]. Each MAC operation is generally associated with a number of memory accesses. In the worst case scenario, where there is no data re-use, each operation requires 3 reads and 1 write to memory. The simplest neural network from the aforementioned models requires around 2172 M memory reads and 724M memory writes. Since these operations consume a considerable amount of processing power, the energy consumption of these algorithms might not meet the requirements of various application scenarios. However, the overall energy consumption can be reduced if the number of operations performed by these algorithms is also reduced. This is possible through various approaches such as reducing image resolution, reducing dataset size, and choosing the algorithm that addresses the application requirements without introducing additional processing overhead. For example, ResNet-50 processing a $224 \times 224 \times 3$ image uses around 7 billion operations per inference [14]. Running this neural network on a 160×160 image would almost halve the number of operations and double the speed, immensely reducing the energy consumption. In terms of algorithm selection, some algorithms are better suited for servers (where there is a wider variety of accessible resources), whereas others can perform well

on IoT devices. If, for example, the energy consumed to classify a single image on the device was considerably less than the energy consumed to transmit the image to the cloud and receive the result, then, as one scales, it becomes advantageous to compute locally.

There have been research efforts to deliver preliminary observations as to how resource-constrained embedded devices perform while executing ML algorithms. Cui et al. [15] used a Raspberry Pi 2 as a gateway and a commercial Intel SR1560SF server. They found a strong relationship between energy and data size. In addition, they found that for some scenarios, the gateway, which employs a low-power processor, performs data processing tasks using a lower amount of energy compared to the server over a long period of time. However, their study focused on how ML algorithms perform for general tasks and generated a model to predict energy consumption solely based on data size. Unfortunately, they did not consider how the type and phase of the algorithm or how specific data characteristics, such as image resolution, impact performance. Carbajales et al. [16] investigated the power requirement of IoT monitoring and sensing on a Raspberry Pi 2 for a smart home application. Their goal was to present a user-friendly visualization of energy consumption across several single board computers (SBCs) including the Raspberry Pi 2B and the BeagleBone Black. Their data processing was limited to time-scaling, averaging, summing, and rounding with no consideration for more complex processing such as ML. In addition, they did not propose any method to predict or forecast the energy requirements of the system. Lane et al. [17] characterized neural network algorithms for various embedded devices including wearables and smartphones. They chose Nvidia Tegra, Qualcomm Snapdragon, and Intel Edison, and measured execution time and energy consumption for each. Out of the four deep learning architectures, two were used for object detection, namely AlexNet and Street View House Numbers (SVHN). While AlexNet has seen state-of-the-art accuracy and can distinguish more than 1000 object classes, SVHN has a more narrow use case: extracting numbers from noisy scenes. Although this research incorporated deep learning, it did not include analysis of how the data characteristics (such as image resolution) influenced the energy consumption. To summarize, despite the insights provided by the aforementioned research efforts into performance, in terms of duration and energy, none of them have investigated the relationship between image input data versus energy, duration, and accuracy. Furthermore, these studies did not provide a useful technique for predicting the energy consumption when multiple parameters are taken into account.

The contributions of this paper are two-fold. First, we identify and characterize how each individual factor of image classification can affect energy cost, duration, and

accuracy. This will equip the research community with the tools necessary to make an informed decision about the design of their edge/fog systems in a way that balances cost with performance. Second, we present a reliable method for predicting the energy consumption of a system without needing to construct and measure data from a prototype. More specifically, in this paper:

– We analyze and visualize the relationships between energy consumption, duration, and accuracy versus dataset size, image resolution, algorithm type, algorithm phase (i.e., training and testing), and device type, when executing ML algorithms on IoT devices. The machine learning algorithms we used in this study are support vector machines (SVM), k-nearest neighbors (k-NN), and logistic regression. These algorithms were selected based on their popularity for image classification, as well as their abundant implementations across several frameworks. We chose the Raspberry Pi 3 (RPi) and the BeagleBone Black Wireless (BB) because they are widely used by the IoT community. We found that despite the BB's access to lower-voltage DDR3L RAM, which has twice the clock speed and transfer rate potential of the RPi's RAM, it generally always took significantly longer for the BB to perform experiments, ultimately leading it to consume more energy. This discrepancy, in part, is credited to the RPi's CPU which, despite being unable to utilize all four of its cores for some experiments, still has a 20% faster clock speed than that of the BB. We present evidence that suggests increasing image resolution serves only to increase energy consumption while providing minimal benefit to accuracy. For example, using the RPi, we find that increasing the resolution of images by 40% for datasets of size 300 and 1500 results in an average increase in processing time of 191% and 217%, and an average increase in energy of 208% and 214%, respectively. Despite these significant increases in energy consumption, the accuracy for the same datasets is *decreased* by 3.64% and 4.64%, respectively, suggesting that, in general, for small datasets it is not beneficial to increase image resolution. Additionally, we conducted experiments utilizing the RPi's multi-core functionality and compared the results with the corresponding single-core data. In this way, we found that using multiple cores provided many benefits including a 70% and 43% reduction in processing time as well as a 63% and 60% decrease in energy consumption for k-NN and logistic regression, respectively.

– Since energy measurement is a lengthy process and requires the use of an accurate power measurement tool, we utilize our experimental data to present a novel energy prediction model. In our attempts to generate the most accurate model, we used three ML algorithms: multiple linear regression, Gaussian process, and random forest regression. After applying the ML algorithms to the validation datasets, random forest regression proved to be the most accurate method, with a R-squared value of 0.95 for the Caltech-256 dataset and 0.79 for the Flowers dataset. The proposed model facilitates decision making about the target hardware platform, ML algorithm, and adjustments of parameters, based on the application at hand.

Table 1 shows the key terms and notations used in this paper. The remainder of the paper is organized as follows. Section 2 discusses the methodology of our experiments. Section 3 presents experimentation results and provides a list of guidelines for the purpose of maximizing performance and system longevity. Section 4 describes our proposed method to generate a random forest model capable of predicting energy consumption. The paper concludes in Sect. 5 by summarizing our findings and highlighting future work directions.

2 Methodology

In this section, we present the main components of our measurement methodology, including hardware platforms, the power measurement tool, the ML algorithms, and the datasets.

2.1 Hardware platforms

In order to maximize the relevance and applicability of our model, we selected hardware devices that are widely adopted by the IoT community. Recent surveys suggest that the RPi is the most popular single board computer (SBC) [18, 19]. The RPi contains a 1.2 GHz quad-core ARM Cortex-A53 BCM2837 processor and 1 GB of DDR2 SDRAM [20]. The RPi also utilizes a 400 MHz Broadcom VideoCore IV GPU and has Wi-Fi, Bluetooth, and Ethernet capabilities. Similarly, the BB was selected because existing surveys place it between the second and third most popular SBC on the market [18, 19]. The BB contains a 1 GHz AM3358 ARM Cortex-A8 OSD3358-512 M-BAS processor and 512 MB of DDR3L SDRAM [21]. The BB also has both Wi-Fi and Bluetooth capabilities.

When comparing the hardware specifications of both devices, it is important to note two key differences. First, while the RPi has nearly twice the SDRAM of the BB, it uses DDR2 SDRAM, which has roughly half the clock speed and transfer rate at 400 to 1066 MHz and 3200 to 8533 MB/s, respectively. Additionally, DDR2 SDRAM

Table 1 Key terms and notations

Term	Description
k-NN	k-Nearest neighbors
SVM	Support vector machine
LOG	Logistic regression
Resolution	The square dimensions of an image measured in pixels.
Phase	The phase of the ML algorithm (i.e., training or testing)
#Images	The number of images present in a dataset (e.g., 300, 600, 900, 1200, 1500)
#Classes	The number of classes belonging to a specific dataset (e.g., 2, 7, 10)
RMSE	Root mean square error: a measure of average deviation between data points and the trend line
R^2	A measure of how closely data fits to a regression line

requires 1.8 V to operate, which is relatively high based on modern standards. In contrast, the BB, which utilizes DDR3 'Low Voltage', only requires 1.35 V. A second major difference between the two boards concerns their processor caches. For the RPi, the L1 cache level contains 32 kB of storage while the L2 cache level contains 512 kB of storage. The BB has 64K of L1 cache storage that is subdivided into 32K of i-cache and d-cache. Additionally, the BB also has 256K of L2 cache storage. Table 2 presents the hardware characteristics of these two boards.

According to the IoT Developer Survey conducted by Eclipse in 2018 [22], Linux (71.8%) remains the leading operating system across IoT devices, gateways, and cloud backends. As a result, we used Ubuntu Mate on the RPi and the Debian Jessie on BB. Both operating systems are 32-bit.

In many industrial applications, IoT devices are often under strict energy constraints. Under these circumstances, the devices are set to use only absolutely essential services, protocols, and hardware in order to reduce the power consumption of the system [8]. There are many benefits to this system layout including an increase in energy efficiency, a reduction of operating costs for line-powered systems, and an increase in the operating life for battery-powered systems [8, 23]. In order for our energy consumption analyses and models to be realistic, we needed to eliminate the effect of all unwanted components on performance. Consequently, we disabled all the unnecessary modules that may interfere with energy consumption, such as Bluetooth, Wi-Fi, and Ethernet. In addition, we used a serial connection (UART) to communicate with the boards. This method consumes a negligible amount of power, as opposed to a traditional Ethernet or HDMI connection.

2.2 Power measurements

Accurate energy measurement requires enabling and disabling an energy measurement tool based on the operation being performed. For example, during our experiments, it was necessary to enable and disable energy measurement right before and after the training phase, respectively. Therefore, we required a tool that could be directly controlled by the ML program running on the RPi or BB. To this end, we use the EMPIOT tool [24], which enables the devices under test to precisely control the instances of energy measurement. EMPIOT is capable of supersampling approximately 500,000 readings per second to data points streamed at 1 kHz. The current and voltage resolution of this platform are $100\,\mu A$ and 4 mV, respectively,

Table 2 Specifications of the IoT boards used

Board	Raspberry Pi 3 (RPi)	BeagleBone Black (BB)
Processor	1.2 GHz 64-bit quad-core Broadcom BCM2837 ARMv8 [20]	1 GHz TI Sitara AM3359 ARM Cortex A8 [21]
Instruction set	ARMv8	ARMv7
L1 cache	32 kB	64K
L2 cache	512 kB	256 kB
RAM	1 GB LPDDR2	512 MB DDR3L
Storage	SD	4 GB eMMC, SD

when the 12-bit resolution mode is configured. The flexibility of this platform allowed us to integrate it with our testbed.

2.3 Machine learning algorithms

Our paper focuses on supervised image classification. Supervised learning uses *labelled data*. A labeled example consists of an input and output pair. The objective of the supervised algorithm is to produce a model that is able to map a given input to the correct output. Types of learning tasks that are considered as supervised learning include classification and regression. Popular supervised algorithms include Support Vector Machine (SVM) and linear classifiers [25].

In order to grasp the impact of the ML algorithm's effect on energy consumption, it is important to test each algorithm on a wide variety of datasets. As a result, we selected three algorithms: *SVM*, *logistic regression*, and *k-Nearest Neighbors* (k-NN). In addition to being very popular ML algorithms, each has specific strengths and weaknesses that we study in this paper.

SVM operates by mapping input data to a high-dimensional feature space so that data points can be classified, even when the points are not otherwise linearly separable [26]. The data is then transformed in such a way that a hyper-plane can separate them. The objective of SVM is to maximize the distance (margin) from the separating hyper-plane to the support vectors. Logistic regression is used to predict the probability of an event by fitting data to a logistic curve [27]. It is intended for predicting a binary dependent variable (e.g., $y = 1$ or $y = -1$). k-NN classifies a new sample point based on the majority label of its k-nearest neighbors.

SVM can effectively model non-linear decision boundaries while simultaneously being well insulated against pitfalls such as overfitting. However, because SVM may utilize multi-dimensional kernels, it is often memory intensive, thereby leading us to believe it would consume large amounts of energy for datasets with greater than two classes. Additionally, because SVM was originally designed to be a binary classifier, we wanted to measure the effectiveness of current SVM implementations when applied to datasets with multiple classes [26]. Scikit-Learn, the ML library used for our experiments, implements SVM using the 'one-vs-one' method to generate its classifiers. Using this approach, given k binary classifiers, all pairwise classifiers are evaluated resulting in $k(k-1)/2$ distinct binary classifiers. These classifiers, in turn, vote on the test values, which are eventually labeled as the class with the greatest number of votes [28].

Similar to SVM, logistic regression is also designed as a binary classifier, though it does not have the same access to non-linear kernels that SVM does. While logistic regression generally performs well for datasets consisting of two classes, its performance drops considerably as the number of classes increases. For Scikit-Learn's implementation of logistic regression, when the dataset contains a number of classes greater than two, it uses the 'one-vs-all' method. This involves training a separate binary classifier for each class. As the number of classes increases, so does the processing time per class.

The k-NN algorithm is among the simplest and most powerful ML algorithms used for classification and regression. When the task is classification, k-NN classifies an object by assigning it to the most common class among its k-nearest neighbors. While k-NN is generally recognized as a high-accuracy algorithm, the quality of predictions greatly depends on the method used for proximity measurements. Consequently, it was important to select an implementation that used an appropriate distance measurement method, especially when the data points occupy multiple dimensions.

2.4 SciKit-Learn framework

For the purposes of our experiments, we used Scikit-Learn, a Python library that includes ML algorithms such as SVM, logistic regression, and k-NN [29]. Although Scikit-learn offers options to utilize multi-core processing, only two of our three algorithms implemented in Scikit-Learn can make use of multiple cores, namely k-NN and logistic regression. In order to measure the benefits of multi-core utilization, we recorded data for the RPi and compared it with the data gathered throughout the single-core experimentation. The BB was excluded from this iteration of experimentation because it only has a single core.

Scikit-Learn also includes modules for hyper-parameter tuning and cross validation. One such module is GridSearchCV which performs an exhaustive search in the hyper-parameter space. When "fitting" the model on a dataset, all the possible combinations of parameter values are evaluated and the best combination is retained. This can be computationally expensive, especially if you are searching over a large hyper-parameter space and dealing with multiple hyper-parameters. A solution is to use RandomizedSearchCV, in which a fixed number of hyper-parameter settings are sampled.

2.5 Training datasets

In order to utilize a diverse range of data, we chose a total of 5 datasets that originally varied in many factors such as image resolution, number of classes, and dataset size. We standardized all of these factors in order to fairly compare energy consumption results across multiple datasets. No
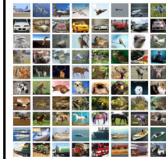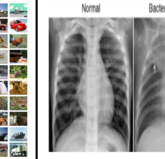
Dataset	MNIST Digits	MNIST Fashion	CIFAR-10	Chest X-ray	Faces in the Wild
No. of Classes	10	10	10	2	7
Class Names	0,1,2,3,4, 5,6,7,8,9	T-shirt, Trousers, Dresses, Coats, Sandals, Shirts, Sneakers, Bags, Pullovers, Ankle Boots	Airplane, Automobile, Bird, Cat Deer, Dog, Frog, Horse, Ship Truck	Normal, Pneumonia	Person1, Person2, Person3, Person4, Person5, Person6, Person7
Image Dimension	2-D	2-D	3-D	3-D	2-D
Color	No	No	Yes	No	Yes
Graphic					

Fig. 1 Summary of the datasets used in this paper. These datasets enable us to study the impact of various parameters on processing time and energy

classes overlapped between the datasets, ensuring that our results are pooled from a wide range of test sources. The datasets are summarized in the following section and in Fig. 1.

2.5.1 MNIST digits

The Modified National Institute of Standards and Technology (MNIST) Digits dataset consists of 70,000 black and white images of handwritten digits [30]. Each of the digits have been centered in a standardized 28×28 image. Each digit corresponds to a separate class resulting in a total of ten classes. This dataset was selected because it is a standard benchmarking dataset.

2.5.2 Fashion-MNIST

The Fashion-MNIST dataset was created by researchers at an e-commerce company called Zalando [31]. According to the creators, it is intended to serve as a direct drop-in replacement for the original MNIST dataset for benchmarking ML algorithms. Similar to the Digits dataset, the Fashion-MNIST dataset consists of 70,000 black and white 28×28 images separated into ten classes. We selected this dataset because it is a more challenging version of the Digits dataset. Though widely popular, the Digits dataset has become too elementary with many ML algorithms easily achieving 97% accuracy. Furthermore, most pairs of digits can be distinguished with a single pixel [31].

2.5.3 CIFAR-10

The CIFAR-10 (CIFAR) dataset was created by the Canadian Institute For Advanced Research and consists of 60,000 color images [32]. Each image is 32×32 pixels, and there are a total of ten classes. The classes in this dataset are very diverse ranging from dogs and cats to airplanes and trucks. This dataset was selected because it is considered more challenging relative to the other datasets. The 3-D color images ensure the matrices representing this dataset's images are dense, thus requiring more computation. Additionally, because this dataset has ten significantly different classes and the maximum dataset size is a mere 1500 images, it was intended to represent a scenario where accuracy is low.

2.5.4 Chest X-ray

The chest X-ray (CHEST) dataset is provided by Kermany et al. [33]. The dataset contains 5863 high-resolution greyscale X-ray images divided into two classes: normal and pneumonia. The images are not square and resolutions are non-uniform. This dataset was selected because it only has two classes, which is ideal for SVM and logistic regression.

2.5.5 Faces in the wild

The Labeled Faces in the Wild dataset was created by researchers at the University of Massachusetts Amherst and consists of 13,000 non-square, color images [34]. The

images were collected from the web using the Viola–Jones face detector. Each of the images is labeled with the name of the person in the picture. A total of 1680 individuals pictured within the dataset contain at least two distinct photos. This dataset was selected because much like the CIFAR dataset, this dataset contains many classes and color images. However, unlike CIFAR, the images within the Faces in the Wild dataset are two dimensional.

2.6 Dataset standardization

We performed dataset standardization in order to fairly determine the nature of the relationship between certain parameters and energy consumption when executing image classification algorithms. We began by first selecting 1500 images from each dataset. Then, we created four more subsets by reducing the size by 300 images at each iteration. This yielded subsets of 1200, 900, 600, and 300 images. Next, we scaled each of the images from those five subsets into three resolutions: 28×28, 22×22, and 17×17. For 3-D data, the dimensionality of the images were maintained.

For each iteration of the experiment, we tested a unique combination selected from 3 ML algorithms, 5 datasets, 2 phases, 5 sizes, and 3 resolutions. This resulted in 450 tests per single complete experiment iteration. Furthermore, in order to ensure a reliable measurement, the experiment was executed 5 times for a total of 2250 experiments. Figure 2 depicts a visualization of the total number of single-core experiments conducted.

3 Results and analysis

This section presents and analyzes the different relationships we observed throughout the experiments. Specifically, we explore how the algorithm used, as well as various image characteristics, affect energy consumption, processing duration, and accuracy.

3.1 Image resolution

In this section, we study the effect of image resolution on energy consumption and processing duration. Figure 3 displays a subset of the collected results for both the RPi and BB during the training phase of logistic regression when the dataset size is held constant. We observe a linear trend between image resolution and energy consumption for each algorithm during both phases.

A higher resolution implies the device must analyze more pixels in order to classify the image. An increase in the number of features (pixels) increases the memory consumption and prediction latency. For a matrix of M

instances with N features, the space complexity is in $O(N \times M)$ [29]. From a computing perspective, this also means that the number of basic operations (e.g., multiplications for vector-matrix products) increases as well. Overall, the prediction time increases linearly with the number of features. Depending on the global memory footprint and the underlying estimator used, the prediction time may increase non-linearly [29]. Increasing the memory and the prediction latency directly increases the energy consumption.

3.2 Dataset size

For this experiment, we held all other parameters constant and varied the number of images in the training and testing sets. Through an analysis of polynomials of varying degrees, we find a quadratic relationship exists between energy consumption and dataset size for each algorithm when the image resolution was held constant. Figure 4 is another subset of the collected results that shows this relationship on both hardware platforms during the testing phase of k-NN. Please note the scale difference between Fig. 4a and b, which highlights the significant difference between the RPi and BB.

This trend was expected because as the number of images the device has to process during the training phase and classify during the testing phase increases, the longer the device will be running and consuming energy.

3.3 Image dimensions

As expected, datasets with 3-D data (e.g., $28 \times 28 \times 3$) generally show higher energy consumption than datasets with 2-D data (e.g., 28×28). Both the CIFAR and CHEST datasets had 3-D data and their energy consumption was consistently higher than the remaining datasets. In addition, the CIFAR dataset consistently represents the highest energy level because not only does it contain 3-D data, but the matrices representing the images are not sparse. In order to quantify this increase, we took an average of the energy consumption for all non-CIFAR data and compared it with the energy consumption of the CIFAR data. The values in Table 3 are calculated in this way. However, in Table 4 we calculated the average energy consumption of the Fashion, Digits, and Faces datasets and compared it with the average energy consumption of CIFAR and CHEST separately. On average, we found that training a logistic regression model using CIFAR images for dataset sizes of 300, 900, and 1500 consumes 550%, 612%, and 636% more energy, respectively, on the RPi. We observe a similar trend on the BB, which, under the same circumstances, consumes 446%, 583%, and 633% more energy, respectively. This is because the CIFAR dataset images

Fig. 2 Visualization of all the experimental combinations conducted per board. Each experiment is conducted 5 times for a total of 2250 experiments
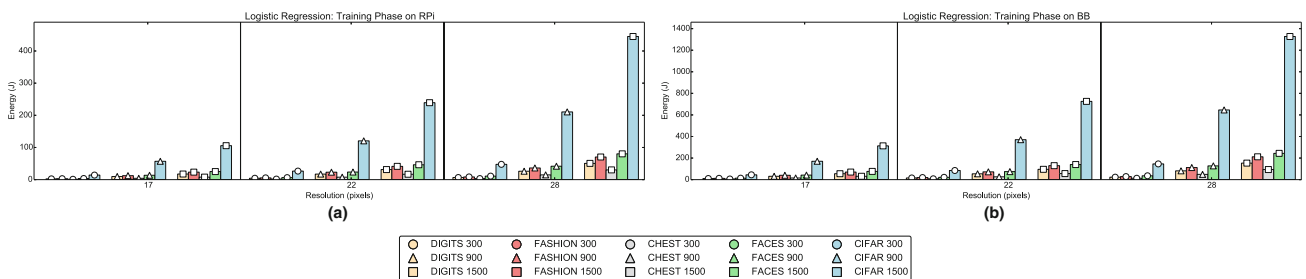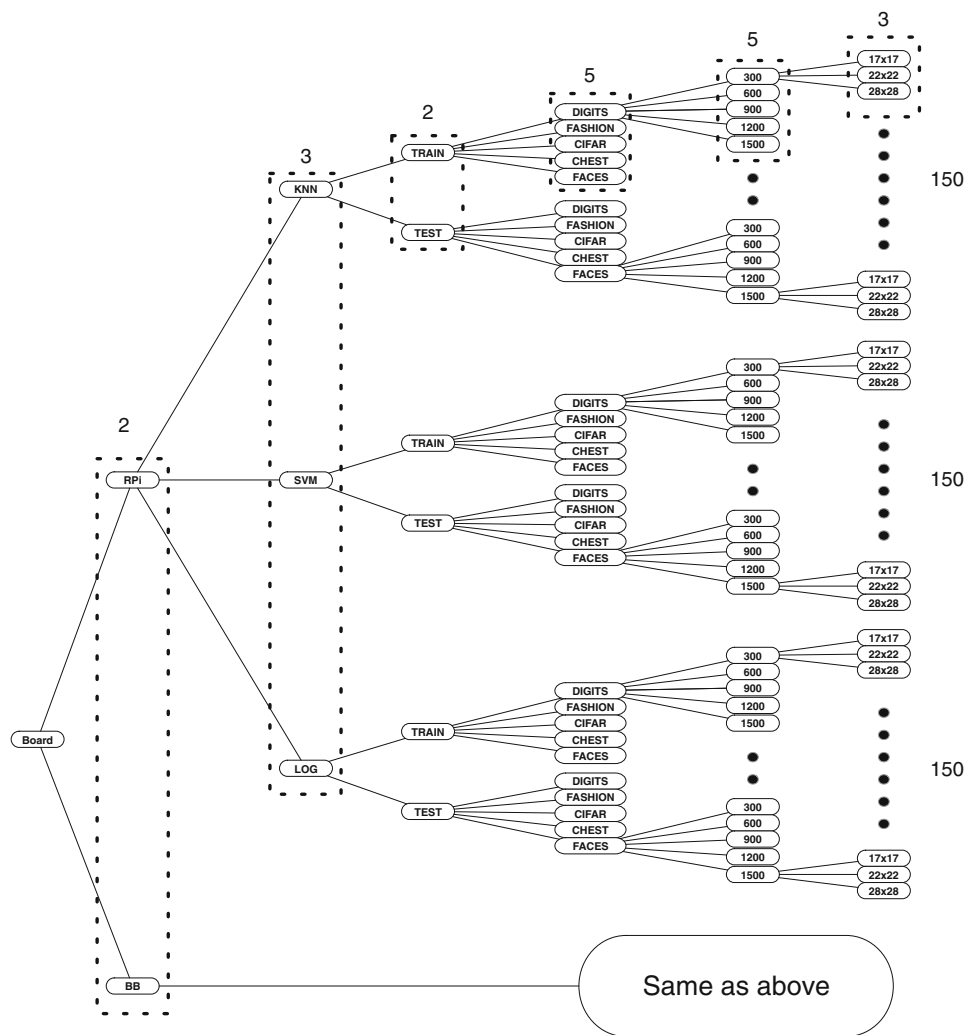

Fig. 3 Energy consumption versus resolution during the training phase of Logistic Regression on **a** RPi and **b** BB. Each resolution is separated into three groups representing dataset sizes 300, 900, and 1500. The training phase of logistic regression consumes up to approximately 450 J and 1400 J for the RPi and BB, respectively. The trend holds across dataset sizes and boards as demonstrated by the figure

contain various colors throughout the entire image as shown in Fig. 1, whereas the CHEST images are greyscale and the variance in color is concentrated in the center of the images (the CHEST images generally show a white chest in the center and a black background), thus resulting in sparser matrices. Scipy, the Python module which Scikit-Learn is built on top of, provides sparse matrix data structures which are optimized for storing sparse data. The main benefit of sparse formats is that the space and time complexity decrease significantly. Specifically, the space complexity decreases because the format does not store zeros. Storing a non-zero value requires on average one 32-bit integer position, a 64-bit floating point value, and an additional 32 bits per row or column in the matrix [29].
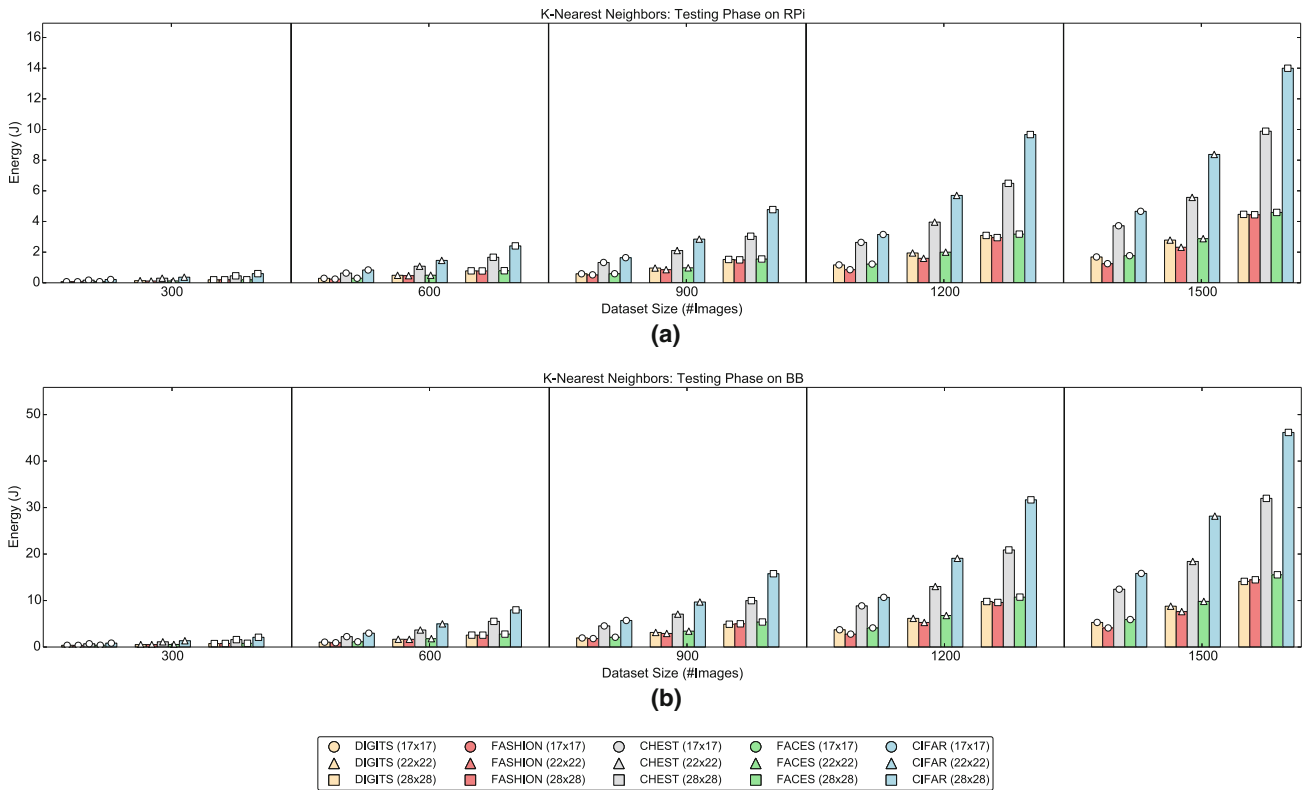
Fig. 4 Energy consumption versus dataset size during testing phase of k-Nearest Neighbors on **a** RPi and **b** BB. Each dataset size is separated into three groups representing image resolutions 17×17, 22×22, and 28×28 pixels. Our analysis confirms the quadratic increase of energy consumption versus dataset size

Table 3 The percent increase of energy consumed by CIFAR versus the other datasets when increasing the resolution from 17×17 to 28×28

Dataset	Device	Dataset size	% Increase
CIFAR	RPi	300	550
CIFAR	RPi	900	612
CIFAR	RPi	1500	636
CIFAR	BB	300	446
CIFAR	BB	900	583
CIFAR	BB	1500	633

Therefore, prediction latency can be dramatically sped up by using a sparse input format since only the non-zero valued features impact the dot product and thus the model predictions. For example, suppose there are 500 non-zero values in a 10^3 dimensional space. Using Scipy's sparse formats reduces the number of multiply and add operations from 10^3 to 500.

3.4 Algorithm

Though the image characteristics in isolation affect energy consumption, our results show that the ML algorithm used

Table 4 The percent increase of energy consumed by CIFAR/CHEST (left column) against the average energy consumed by the other datasets

Dataset	Device	Resolution	% Increase
CIFAR	RPi	17×17	104
CHEST	RPi	17×17	142
CIFAR	RPi	22×22	119
CHEST	RPi	22×22	127
CIFAR	RPi	28×28	122
CHEST	RPi	28×28	111
CIFAR	BB	17×17	100
CHEST	BB	17×17	128
CIFAR	BB	22×22	115
CHEST	BB	22×22	114
CIFAR	BB	28×28	122
CHEST	BB	28×28	106

is consistently the greatest predictor of energy consumption and processing duration. This is because these algorithms are designed for specific tasks. For example, the CHEST dataset, which contains 3-D images, generally consumes

the second highest amount of energy when using SVM and k-NN. However, when logistic regression, which is designed for binary classification, is run on the CHEST dataset, we see a dramatic decrease in energy, regardless of its high dimension because the CHEST dataset only has two classes.

In general, we found that logistic regression's training phase consumes significantly more energy than the training phases of the other two algorithms. Figure 3a and b show that the training phase of logistic regression consumes up to approximately 450 J and 1400 J for the RPi and BB, respectively. In comparison, the training phases of k-NN and SVM consume 2 J and 100 J on the RPi and 9 J and 450 J on the BB, respectively. This large discrepancy in energy cost is observed because training a logistic regression model for more than 2 classes involves creating a separate classifier for each class. On the other hand, the testing phase for logistic regression consumes significantly less energy than SVM and k-NN because predicting a single image is simply a matter of taking the maximum output across each classifier generated during the training phase. This trade-off is an important consideration when determining which algorithm to use for a resource-constrained edge device.

For k-NN, we also observe quadratic trends, as shown in Fig. 4 for both the RPi and the BB. During its training phase, k-NN simply stores all the images in the training set. Suppose there are n training examples each of dimension d. Then the complexity to compute the distance to one example is $O(d)$. To find a single nearest neighbor, the complexity is $O(n \times d)$. Thus, to find the k nearest neighbors, the complexity is $O(k \times n \times d)$ [35]. As the dataset size increases, the overall complexity increases, which in turn increases the energy consumed. Therefore, the overall complexity is dependent on the dataset size and the value of k. Choosing k optimally is not a trivial task. In theory, with an infinite number of samples, as the value of k increases, the error rate approaches the optimal Bayes error rate. The caveat being that all k neighbors have to be close to the example. However, this is impossible since the number of samples is finite. A large k leads to oversmoothed decision boundaries, and a small k leads to noisy decision boundaries. For our experiments, we used cross-validation to tune k to a value of 5.

For SVM, the energy consumption depends on the number of support vectors. A higher number of support vectors indicates a higher model complexity. Our results show that, the processing duration asymptotically grows linearly with the number of support vectors. The number of support vectors increases when we increase resolution or dataset size, as demonstrated in Fig. 5.

In addition, the non-linear kernel used (radial basis function in Scikit-learn) also influences the latency as it is
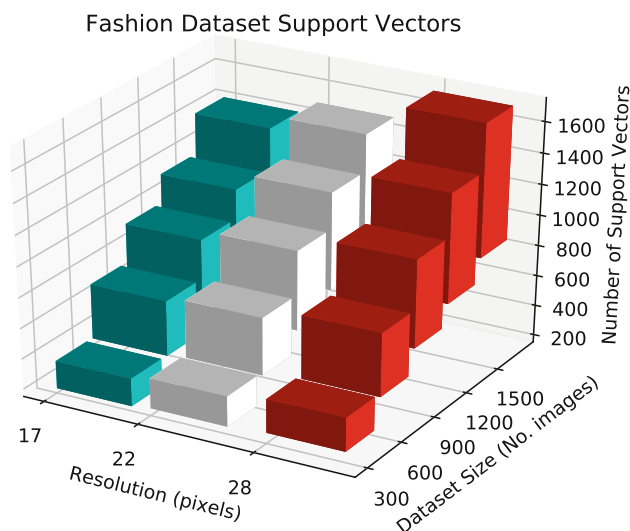

Fig. 5 The number of support vectors for the FASHION Dataset. Increasing the number of images in a dataset has a greater effect on support vector complexity when compared to increasing image resolution

Table 5 The percent increase of energy consumed between dataset and resolution pairs versus lowest to highest dataset size in Fig. 6

Dataset	Device	Resolution	% Increase
CIFAR	RPi	17×17	405
CIFAR	RPi	22×22	486
CIFAR	RPi	28×28	504
CIFAR	BB	17×17	290
CIFAR	BB	22×22	377
CIFAR	BB	28×28	425

used to compute the projection of the input vector once per support vector. Furthermore, since the core of a SVM is a quadratic programming problem which separates support vectors from the rest of the training data, Scikit-learn's implementation of the quadratic solver for SVM scales between $O(n_f \times n_s^2)$ and $O(n_f \times n_s^3)$, where n_f is the number of features and n_s is the number of samples. If the input data is very sparse, n_f should be replaced by the average number of non-zero features in a sample vector. Figure 6 shows that the CIFAR and Faces datasets (which have dense matrices) consistently consume more energy relative to the other datasets during the training phase of the algorithm across all resolutions. Figure 6a highlights this trend with the CIFAR dataset consuming 636% more energy than the average consumption of the other four datasets for the resolution of 28. Table 5 summarizes the effect of image resolution on energy consumption when using CIFAR dataset.
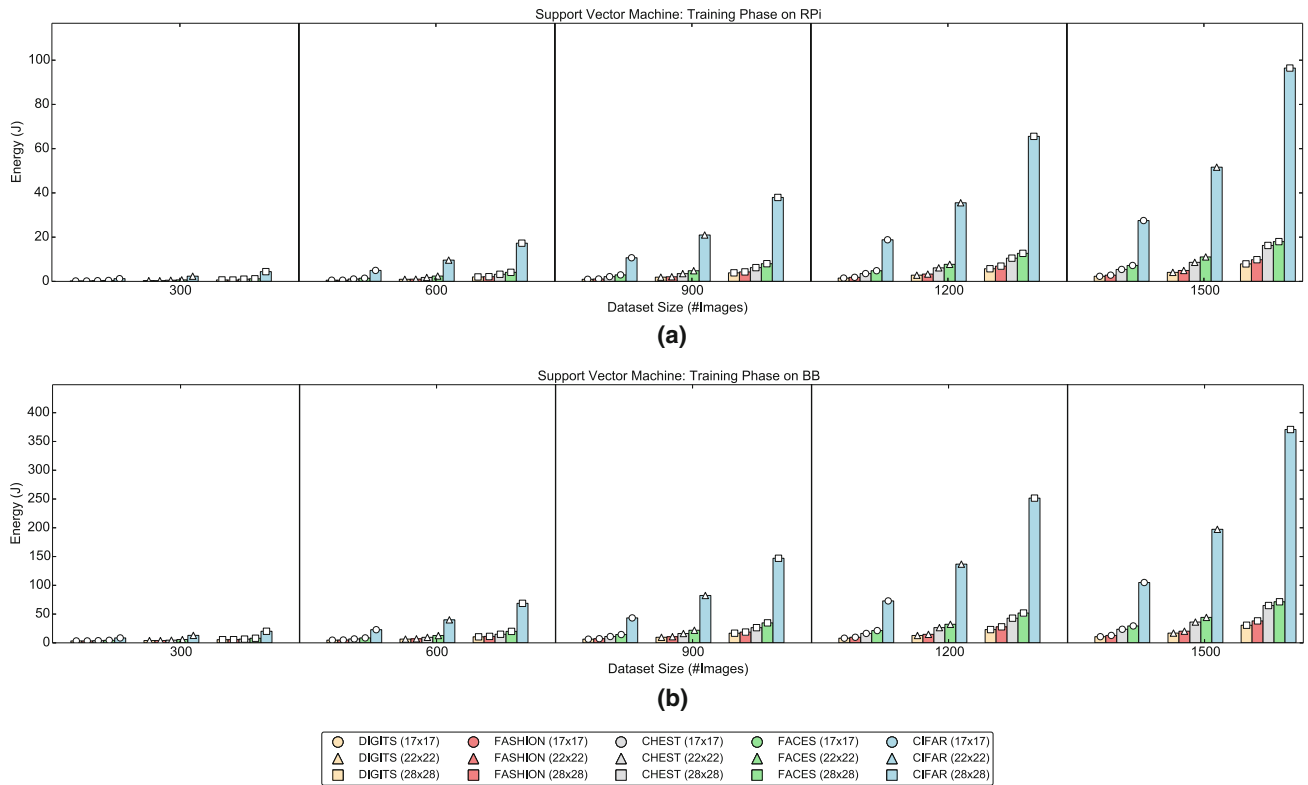
Support Vector Machine: Training Phase on RPi

Support Vector Machine: Training Phase on BB

Fig. 6 Energy consumption versus dataset size during testing phase of k-Nearest Neighbors on **a** RPi and **b** BB. Each dataset size is separated into three groups representing image resolutions 17×17, 22×22, and 28×28 pixels. The CIFAR and Faces (dense matrix representations) datasets consistently consume more energy during the training phase of the algorithm across all resolutions

Table 6 The percent change in time and accuracy when varying between 17×17 and 28×28 image resolutions on the RPi platform

Algorithm	Dataset	Δt_{300} (%)	Δa_{300} (%)	Δt_{1500} (%)	Δa_{1500} (%)
k-NN	DIGITS	137	0	243	− 4.80
k-NN	CHEST	173	− 7.4	1.72	1.48
k-NN	FASHION	200	0	244	1.85
k-NN	FACES	160	0	164	0
k-NN	CIFAR	179	− 12.5	199	− 23.89
SVM	DIGITS	226	− 4.16	224	2.11
SVM	CHEST	200	0	203	− 0.69
SVM	FASHION	226	− 10	218	− 4.1
SVM	FACES	154	0	136	5.31
SVM	CIFAR	206	− 25	233	− 14
LOG	DIGITS	170	0	188	− 0.71
LOG	CHEST	209	7.69	289	0.69
LOG	FASHION	184	− 17.64	202	− 4.13
LOG	FACES	207	0	221	0
LOG	CIFAR	236	14.29	324	− 28.85

The dataset size is constant

3.5 Time and accuracy

In addition to measuring energy consumption, we also measured processing time and classification accuracy.

Specifically, these studies enable us to offer guidelines for establishing trade-offs between energy consumption and accuracy. Tables 6 and 7 show the accuracy increase for

⚫ Springer

Table 7 The percentage increase of processing duration and accuracy when varying image resolution between 17×17 and 28×28 on the BB

Algorithm	Dataset	Δt_{300} (%)	Δa_{300} (%)	Δt_{1500} (%)	Δa_{1500} (%)
k-NN	DIGITS	18.26	0	120	-4.80
k-NN	CHEST	40.29	-7.4	136	1.48
k-NN	FASHION	19.53	0	157	1.85
k-NN	FACES	19.44	0	119	0
k-NN	CIFAR	48.33	-12.5	166	-23.89
SVM	DIGITS	63.66	-4.16	186	2.11
SVM	CHEST	75.34	0	178	-0.69
SVM	FASHION	66.70	-10	194	-4.09
SVM	FACES	77.97	0	143	5.31
SVM	CIFAR	132	-25	244	-14
LOG	DIGITS	123	0	178	-0.71
LOG	CHEST	111	7.69	198	0.69
LOG	FASHION	136	-17.65	201	-4.13
LOG	FACES	161	0	218	0
LOG	CIFAR	225	14.28	329	-28.85

The dataset size is constant

each algorithm and dataset pair at sizes of 300 and 1500 images.

In general, when holding all other factors constant, accuracy does not significantly change when resolution was changed. For example, for the RPi, increasing the resolution from 17×17 to 28×28 (by 40%) while keeping the dataset size constant at 300 images, always resulted in at least double the time and little to no additional increase in accuracy. Table 6 shows that the maximum increase in accuracy across all the experiments is approximately 14% when running logistic regression on a subset of the CIFAR dataset consisting of 300 images. However, this increases time by 236%. We observe that 7 out of the 15 experiments have the same accuracy even when increasing the resolution. Additionally, 6 out of the 15 experiments show a decrease in accuracy. Thus, 13 out of the 15, or roughly 90% of the experiments show that there is no additional benefit to using higher-resolution images. These accuracy trends, which are identical for the BB, are a critical consideration for many applications. As a result, one should generally opt for the reduced resolution.

The increases in energy consumption associated with opting for higher resolution can instead be allocated to increasing the dataset size. This could lead to an increase in accuracy. Figures 4 and 6 demonstrate that choosing the higher resolution is roughly equivalent to choosing the lower resolution at a higher dataset size. For example, in Fig. 6, selecting a dataset size of 300 images at a resolution of 28×28, consumes approximately the same amount of energy as a dataset of 600 images at a resolution of 17×17. Again in Fig. 6, we observe that selecting a dataset size of 600 images at a resolution of 28 consumes

approximately the same amount of energy as that of a dataset of 1200 images at a resolution of 17×17.

3.6 Multi-core versus single-core

To quantitatively determine how the usage of multiple cores affects the energy consumption and processing time, we executed k-NN and logistic regression (the algorithms capable of multi-core processing) using all four cores on the RPi. Figures 7 and 8 demonstrate the differences between multi-core and single-core processing time and energy consumption for the testing phase of k-NN and the training phase of logistic regression. For both time and energy, there is a significant gap between using multi-core and single-core. On average, when utilizing multi-core functionality, the processing time for k-NN and logistic regression was reduced by 70% and 42%, respectively. Using multiple cores also translated in a 63% and 60% decrease in energy consumption for the same two algorithms, respectively.

3.7 Design considerations

In this section, we present our main observations regarding the effect of hardware on performance as well as a set of design guidelines to create real-time IoT systems for the purpose of image classification.

3.7.1 Hardware

While the trends we identified in the previous sections are consistent across the two hardware platforms, it is
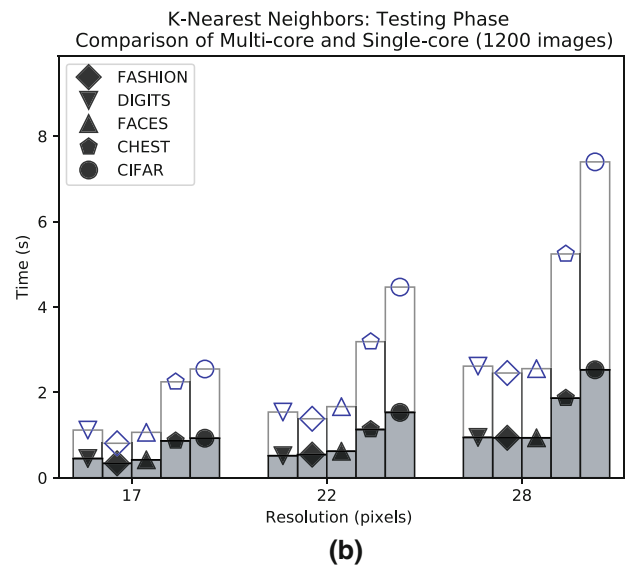
(a)

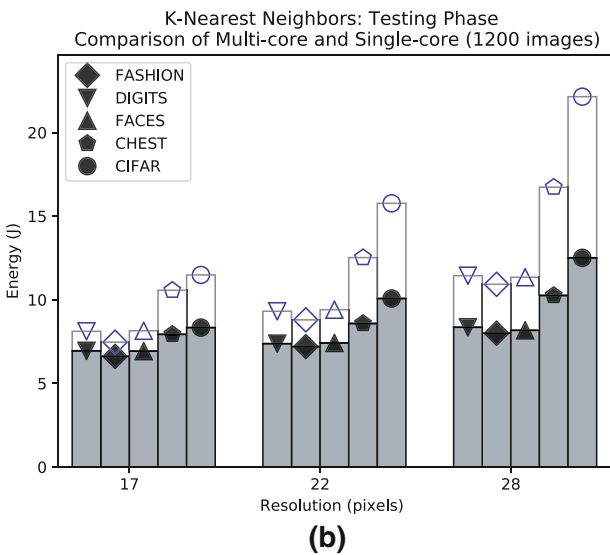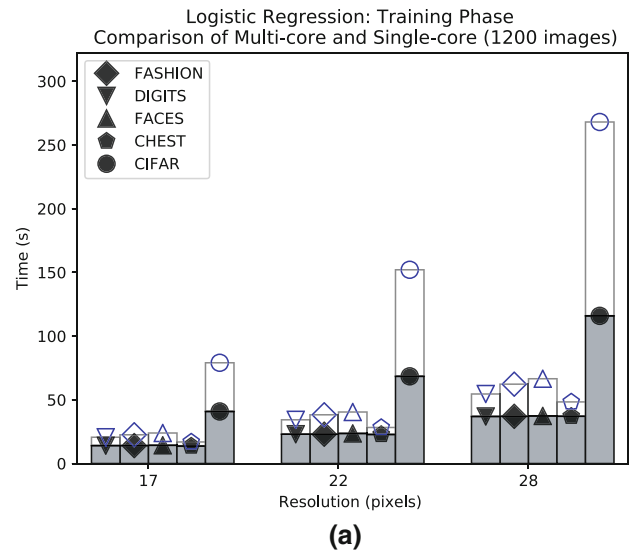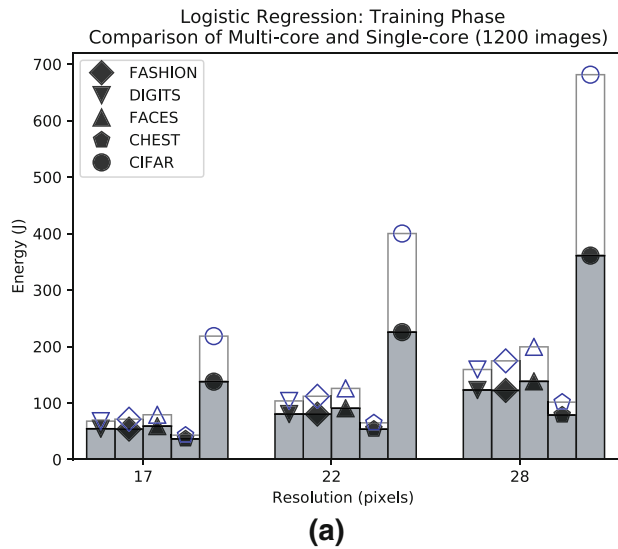**(a)**

(b)

(b)

Fig. 7 Comparison of energy consumption during the training phase of logistic regression using multi-core functionality (dark fill) and single-core functionality (no fill)

Fig. 8 Comparison of processing time during the training phase of logistic regression using multi-core functionality (dark fill) and single-core functionality (no fill)

important to note the dramatic differences in their individual energy consumption. The RPi not only has a CPU that is 20% faster than the BB, but it also boasts nearly twice as much RAM. This variance in hardware results in the RPi completing the experiments much faster than the BB. Consequently, because the BB had to run longer to complete each task, we observe that on average it consumes more energy. This conclusion is best demonstrated by Fig. 3a and b, which display the training phase of logistic regression when a single core is used. The RPi, which could generally train a logistic regression model for more than two classes in less than 4 min, consumed up to 450 J. By contrast, the BB, which generally took 15–19 min to train a logistic regression model, consumed up to

1400 J. This trend is consistent across all algorithms. More importantly, the RPi can achieve significantly higher performance compared to the BB, when the ML algorithm utilizes all the available four cores. In particular, the fast growth of low-cost, multi-core processors justifies their adoption at the IoT edge to lower energy consumption and enhance real-time processing.

3.7.2 Guidelines

The following guidelines are primarily concerned with energy-performance trade-offs. Foremost, we observe that for small datasets it is rarely beneficial to increase image resolution. In most cases, doing so is detrimental to the

accuracy of the model and in all cases there is a significant increase in energy consumption as a result of additional pixel analysis. Second, we suggest to constrain dataset size to a minimum. While increasing the training set to tens of thousands of images would likely result in higher accuracy, for the small set increments associated with current IoT systems and traditional ML algorithms, adding additional images is not guaranteed to provide significant benefit. However, similar to increasing image resolution, increasing the dataset size or dimensionality always translates in higher energy consumption. Third, we suggest that images be captured in methods suited for sparse matrix representations so that they may benefit from the optimizations associated with sparse matrix formats. It should be noted that in addition to enhancing performance, these methods also can be applied to improve user privacy. For example, low-resolution and sparse images that do not reveal the person's identity could be captured by thermal cameras to achieve low-power and real-time activity recognition [36].

4 Modeling and predicting energy consumption

Our experimentation provided us with a sizable amount of data that can be used to model and predict the energy consumption. In this section, we utilize three statistical analysis techniques, discuss the drawbacks and benefits of each, and compare their performance in terms of prediction accuracy.

4.1 Random forest regression

In random forest regression, a multi-variate input x is used to estimate a continuous label y using the probability density function $p(y|v)$, where $y \in Y \subseteq \mathbb{R}^n$ [37]. In our case, the input contains the following features: device, resolution, number of images, color, number of dimensions, algorithm, and phase of the algorithm. All the features were coded to be categorical. For a feature with n possible values, we created $n - 1$ binary variables to represent it. For example, for 3 possible resolutions, we created 2 columns, r_1 and r_2, such that for a resolution of 17×17, $r_1 = 0$ and $r_2 = 0$. The full encoding for the features is summarized in Table 8.

Constructing a random forest model can be broken down into the following steps:

– Using random sampling, choose N samples from the training data.
– For each sample, randomly choose K features. Construct a decision tree using these features.

Table 8 Random forest regression encoding format

Feature encoding		
Feature	Possible values	#Columns to represent
Resolution	17×17, 22×22, 28×28	2
#Images	300, 600, 900, 1200, 1500	4
#Classes	2, 7, 10	2
Phase	Train, Test	1
Color	Yes, No	1
Algorithm	k-NN, SVM, LOG	2
Device	RPi, BB	1

– Repeat steps 1 and 2 for m times to generate m decision tree models.

The above process results in a random forest of m trees. To predict the y output of a new query point, pass the input to each of the m trees. For regression, the output is the average of m decision tree outputs. For classification, the output is the majority class label of the m decision tree outputs [38].

4.2 Gaussian process and linear regression

In addition to the random forest model, we also evaluate the accuracy of linear regression and Gaussian Process (GP). Linear regression is the most common predictive model to identify the relationship among an independent variable and a dependent variable [39]. The multiple linear regression line is fit to the data such that the sum of the squared errors is minimized. A Gaussian Process defines a distribution over functions which can be used for regression. The main assumption of GP is that the data is sampled from a multivariate Gaussian distribution. The function-space view of GP shows that a GP is completely specified by its mean function and co-variance function.

4.3 Results and discussion

To assess and understand how the proposed prediction model performs on datasets that are not part of the original data, we chose two new datasets and collected data on their energy consumption. The first dataset was drawn from Caltech-256 and was chosen because it contains a more challenging set of object categories [40]. From this dataset, we drew ten separate, mutually exclusive classes. The images within these sub-datasets are in color and of varying image resolutions. The second verification dataset contains images of flowers [41]. We chose this dataset because it contains five classes, which is a characteristic

the random forest was not trained to predict. The images within this dataset are also in color and of varying image resolutions. Following the same experimental protocols, we separated images from each of the datasets into the standard dataset sizes and resolutions. For both datasets, we only chose the 3-D images because our prior experiments demonstrated that the datasets with 3-D images had the highest variations in energy consumption. The characteristics of both datasets are summarized in Table 9.

Table 11 demonstrates that linear regression is least successful at prediction. This algorithm shows $\times 4.2$ and $\times 5.9$ lower R^2 values compared to GP and random forest, respectively, especially when we attempted to extrapolate beyond the range of the sample data (Table 10). This is because linear regression requires assumptions that are invalidated by our data. Specifically, linear regression, as the name suggests, requires the relationship between the independent and dependent variables to be linear, which is not necessarily guaranteed by our data. A GP also requires certain assumptions about the data. For example, a GP requires that all finite dimensional distributions have a multivariate distribution. Specifically, each observation must be normally distributed. Considering our application, for a specific observation $\mathbf{x_i}$, where $\mathbf{x_i} = (x_1, \ldots x_n)$, $x_1 = 0$ if the image is 2-D, and $x_1 = 1$ if it is 3-D. This immediately precludes the normality assumption imposed by a Gaussian model.

We used k-fold cross validation with $k = 10$ to select the random forest that produces the maximum R-squared value and minimum RMSE. Performing k-fold cross validation with $k = 5$ or $k = 10$ has been shown empirically to yield test error rate estimates that suffer neither from excessively high bias nor from very high variance [42]. The random forest predicts the energy consumption of the

Caltech dataset with a R-squared value of 0.95 and a R-squared value of 0.79 for the Flowers dataset. The coefficient of determination, known as R-squared, can be interpreted as the percent of the variation in y that is explained by the variation in the predictor x. A value of 1 for R-squared indicates all of the data points fall perfectly on the regression line which means the predictor x accounts for all of the variation in y. In general, the closer the R-squared is to 1.0, the better the model.

The random forest model is capable of ameliorating the high error rates of the two previous models because it can capture the non-linearity in the data by dividing the space into smaller sub-spaces depending on the features. In addition, there is no prior assumption regarding the underlying distribution of the features. As a result of the bias-variance trade off, increasing the number of trees, as well as their depth, is almost always better. More trees reduce the variance; deeper trees reduce the bias. In practice, there are millions of training examples which will make it unrealistic and infeasible to train a random forest without limiting its depth. In fact, not controlling or tuning these parameters can lead to fully grown and un-pruned trees.

Some important hyper-parameters that require tuning are included in Table 10. To tune our hyper-parameters, we created a grid of values for each parameter and conducted a randomized grid search with cross validation to choose the optimal values. The most important hyper-parameter is the number of features to test at each split. Examining feature importance enables us to identify the impact of each feature on the accuracy of the model. This information can also be used to develop new features, exclude noisy features, and inform another model such as a neural network.

When determining which feature to split a tree on, the model considers a random permutation of the features. The number of features to consider was originally suggested as the total number of features divided by 3. However, recently, it has been empirically justified to use the number of features. When using large datasets, it might not be practical to consider thousands of features at each split. As a result, it has also been suggested to use the square root of the number of features or the log of the number of features.

Table 9 Characteristics of the verification datasets

Dataset	#Classes	#Dimensions	Color
Caltech-256	10	3	Yes
Flowers	5	3	Yes

Table 10 Hyper-parameters and tuned values of the random forest

Description	Value
The number of trees in the forest	800
The number of features to consider when looking for the best split	sqrt (#features)
The maximum depth of the tree	90
Whether bootstrap samples are used when building trees	True
The function to measure the quality of a split	MSE
The minimum number of samples required to be at a leaf node	4
The minimum number of samples required to split an internal node	10

Fig. 9 Feature importance (a.k.a., 'Gini importance' and 'mean decrease impurity') for the random forest model. The feature importance is defined as the total decrease in node impurity averaged over all trees of the ensemble
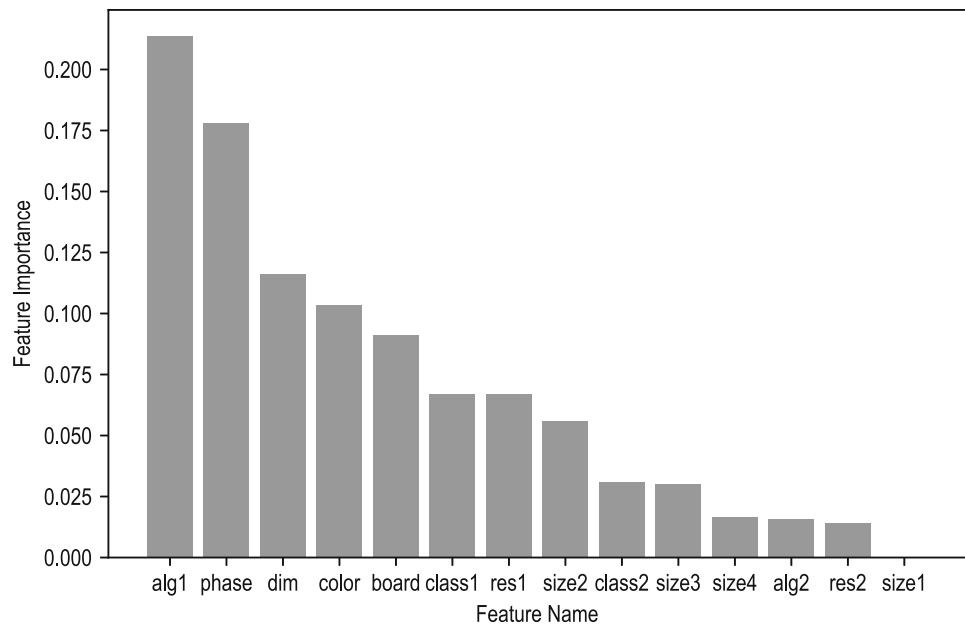

Table 11 R^2 comparison

ML model	Flowers	CALTECH-256	Original datasets
Linear regression	0.34	0.30	− 0.15
Gaussian process	0.50	0.63	0.49
Random forest	0.79	0.95	0.74

Our model contains 14 features as described in Table 8. Their importance is highlighted in Fig. 9. We found that *size2* (the second column of the encoding for size) is computed to have about ×5 higher importance than *size4*, while their 'true' importance is very similar. When the dataset has two (or more) correlated features, then from the point of view of the model, any of these correlated features can be used as the predictor, with no concrete preference of one over the others.

When using the impurity-based ranking (for determining which feature to split on), once one of the features is used at the split, the importance of other correlated features is significantly reduced. When our goal is to reduce overfitting, it makes sense to remove features that are mostly duplicated by other features. However, this might lead one to the incorrect conclusion that one of the variables is a stronger predictor compared to other features in the same group when in fact, they share a very similar relationship with the response variable. It is important to note the difficulty of interpreting the importance of correlated variables is not specific to random forests, but applies to most model-based feature selection methods.

Table 12 RMSE of the predictions achieved by the random forest model when predicting the energy consumption of k-NN, SVM, and LOG

Algorithm	Phase	Dataset	RMSE	Range	N_RMSE
k-NN	Train	O	0.162	4.409	0.036
k-NN	Train	F	8.518	11.55	0.737
k-NN	Train	C	8.400	11.91	0.705
k-NN	Test	O	1.389	14.14	0.098
k-NN	Test	F	5.464	45.28	0.120
k-NN	Test	C	4.989	44.43	0.112
SVM	Train	O	27.508	196.81	0.139
SVM	Train	F	30.980	348.01	0.089
SVM	Train	C	26.509	334.57	0.079
SVM	Test	O	1.561	12.53	0.124
SVM	Test	F	5.004	30.19	0.165
SVM	Test	C	4.399	28.92	0.152
LOG	Train	O	170.54	1305	0.130
LOG	Train	F	181.08	776.67	0.233
LOG	Train	C	157.74	1117.46	0.141
LOG	Test	O	0.021	0.248	0.086
LOG	Test	F	4.465	6.476	0.689
LOG	Test	C	4.379	6.241	0.701

We examined RMSE to quantitatively measure the performance of the random forest model. To place it in the context of the data the model is predicting, the RMSE must be normalized. One such method of normalization involves dividing the RMSE by the range. Table 12 highlights the range, RMSE, and normalized RMSE, separated by

algorithm and phase for each dataset. Datasets are coded as follows: original (O), Flowers (F), and Caltech (C).

In order to evaluate the random forest's performance in terms of the two validation datasets, we took an average of their normalized RMSEs for each algorithm and phase. The average normalized RMSE for k-NN, SVM, and logistic regression during the training and testing phases are as follows: 72.1%, 11.6%, 8.4%, 15.8%, 18.7%, and 69.5%, respectively. Among these values, SVM has the lowest average error rate at 12.1%. This error is approximately $\times 3.4$ less than what was exhibited by k-NN and logistic regression, on average. The model performs poorly for k-NN's training phase and logistic regression's testing phase on the verification datasets because of the extreme polarity and variation in the data for these algorithm and phase combinations. For example, for the Flowers dataset, the maximum value for logistic regression testing is 6.61 J, while the minimum is 0.13 J. This variation may cause the random forest to poorly predict this configuration. As anticipated, the random forest outputs the lowest prediction accuracy for the Flowers dataset because the training set for the random forest did not include a dataset with five classes.

5 Conclusion

As IoT systems become increasingly powerful, edge computing has become more practical compared to offloading data processing to cloud platforms. This trend has unlocked enormous potential in sectors focused on real-time computing, as it allows IoT systems to quickly and reliably process data while consuming lower energy overall. This is particularly useful for IoT systems involved in image classification, where the timely processing of data is critical for prompt decision making. Our experiments sought to explore the relationships between energy consumption, processing duration, and accuracy, versus various parameters including dataset size, image resolution, algorithm, phase, and hardware characteristics. We benchmarked two IoT devices in order to reliably identify and study the parameters affecting energy consumption. Our studies show distinct quadratic and linear relationships between dataset size and image resolution with respect to energy consumption. Choosing a lower resolution speeds up the execution time and reduces energy consumption significantly, while maintaining the accuracy of a model trained on a higher resolution. If even a slight change in accuracy is crucial for a given system, then selecting a lower resolution frees energy that can be allocated towards increasing dataset size.

In addition to being a lengthy process, energy profiling requires an accurate and programmable power measurement tool. Consequently, we propose a random forest model to predict energy consumption given a specific set of features. While we demonstrated that our model can predict energy consumption with acceptable accuracy for inputs with previously unseen characteristics, it relied on most of the remaining parameters being similar. To address this concern, this model would greatly benefit from additional training on datasets with completely different characteristics. Additionally, we searched a small hyper-parameter space when tuning the random forest. Thus, future work could be focused on testing more values. In order to further increase the versatility of the model, future work could be focused on collecting data from additional hardware devices to reflect the diversity in the IoT community. Additionally, in its current state, this model can only be used statically. Should a user rely on this model for a specific task, prior to deployment, they would need to test each algorithm against the task to find the impact of each on energy consumption. Future work may also focus on creating a system that dynamically changes the algorithm based on the real-time data returned from the model. Furthermore, future work could also focus on adding more ML algorithms to the model. Pairing this concept with the addition of a model trained on multiple hardware devices would result in a robust system capable of performing tasks optimally and making the best use of its limited computational resources, especially energy.

References

1. Hung, M.: Leading the IoT, Gartner insights on how to lead in a connected world. In: Gartner Reseach, pp. 1–29. https://www.gartner.com/imagesrv/books/iot/iotEbook_digital.pdf (2018)
2. Kumar, A., Goyal, S., Varma, M.: Resource-efficient machine learning in 2 kb ram for the internet of things. In: International Conference on Machine Learning, pp. 1935–1944 (2017)
3. Naha, R.K., Garg, S., Georgakopoulos, D., Jayaraman, P.P., Gao, L., Xiang, Y., Ranjan, R.: Fog computing: survey of trends, architectures, requirements, and research directions. IEEE Acess **6**, 47980–48009 (2018)
4. Ni, J., Zhang, K., Lin, X., Shen, X.S.: Securing fog computing for internet of things applications: challenges and solutions. IEEE Commun. Surv. Tutor. **20**(1), 601–628 (2017)
5. Lin, J., Yu, W., Zhang, N., Yang, X., Zhang, H., Zhao, W.: A survey on internet of things: architecture, enabling technologies, security and privacy, and applications. IEEE Internet Things J. **4**(5), 1125–1142 (2017)

6. Wei, J.: How wearables intersect with the cloud and the internet of things: considerations for the developers of wearables. IEEE Consum. Electron. Mag. **3**(3), 53–56 (2014)

7. Metcalf, D., Milliard, S.T.J., Gomez, M., Schwartz, M.: Wearables and the internet of things for health: wearable, interconnected devices promise more efficient and comprehensive health care. IEEE Pulse **7**(5), 35–39 (2016)

8. Amirtharaj, I., Groot, T., Dezfouli, B.: Profiling and improving the duty-cycling performance of linux-based iot devices. J. Ambient Intell. Humaniz. Comput. **30**, 1–29 (2018)

9. Centre for Energy-Efficient Telecommunications: the power of wireless cloud. https://ceet.unimelb.edu.au/publications/ceet-white-paper-wireless-cloud.pdf (2013)

10. Krizhevsky, A., Sutskever, I., Hinton, G. E.: Imagenet Classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)

11. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9 (2015)

12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)

13. Shafique, M., Theocharides, T., Bouganis, C.-S., Hanif, M.A., Khalid, F., Hafiz, R., Rehman, S.: An overview of next-generation architectures for machine learning: roadmap, opportunities and challenges in the IoT era. In: Design, Automation & Test in Europe Conference & Exhibition. IEEE, pp. 827–832 (2018)

14. ARM Ltd., Deploying neural networks on Android-based mobile and embedded devices. https://developer.arm.com/solutions/machine-learning-on-arm/developer-material/how-to-guides/optimizing-neural-networks-for-mobile-and-embedded-devices-with-tensorflow/deploying-neural-networks-on-android-based-mobile-and-embedded-devices-single-page

15. Cui, W., Kim, Y., Rosing, T.S.: Cross-platform machine learning characterization for task allocation in IoT ecosystems. In: IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), pp. 1–7 (2017)

16. Carbajales, R.J., Zennaro, M., Pietrosemoli, E., Freitag, F.: Energy-efficient internet of things monitoring with low-capacity devices. In: IEEE World Forum on Internet of Things: 14–16 December, Milan, Italy: Proceedings. Institute of Electrical and Electronics Engineers (IEEE), pp. 305–310 (2015)

17. Lane, N. D., Bhattacharya S., Georgiev, P., Forlivesi, C., Kawsar, F.: An early resource characterization of deep learning on wearables, smartphones and internet-of-things devices. In: Proceedings of the International Workshop on Internet of Things Towards Applications. ACM, pp. 7–12 (2015)

18. Top 10 Open Source Linux and Android SBCs, Aug 2014. https://www.linux.com/news/top-10-open-source-linux-and-android-sbcs

19. Top 10 Best Open-Spec Hacker SBCs, Jun 2016. https://www.linux.com/news/raspberry-pi-stays-top-survey-81-open-spec-sbcs

20. BCM2837 ARM Peripherals. https://web.stanford.edu/class/cs140e/docs/BCM2837-ARM-Peripherals.pdf

21. BeagleBone Black Wireless. https://beagleboard.org/black-wireless

22. Eclipse Foundation, Key Trends from the IoT Developer Survey, 2018. https://iot.eclipse.org/resources/iot-developer-survey/iot-developer-survey-2018.pdf

23. Dang, D.: Reducing the Cost, Power and Size of Connectivity in IoT Designs, Feb, 2018. http://www.ti.com/lit/wp/sway013/sway013.pdf

24. Dezfouli, B., Amirtharaj, I., Li, C.-C.: EMPIOT: an energy measurement platform for wireless IoT devices. J. Netw. Comput. Appl. **121**, 135–148 (2018)

25. Mahmud, M., Kaiser, M.S., Hussain, A., Vassanelli, S.: Applications of deep learning and reinforcement learning to biological data. IEEE Trans. Neural Netw. Learn. Syst. **29**(6), 2063–2079 (2018)

26. Manning, C., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge (2008)

27. Harrell, F.E.: Ordinal Logistic Regression, in Regression modeling strategies, pp. 311–325. Springer, Berlin (2015)

28. Wang, L.: Support Vector Machines: Theory and Applications, vol. 177. Springer, Berlin (2005)

29. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)

30. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)

31. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, arXiv preprint arXiv:1708.07747 (2017)

32. Krizhevsky, A.: Learning multiple layers of features from tiny images. University of Toronto, Technical Report (2009)

33. Kermany, D.S., Goldbaum, M., Cai, W., Valentim, C.C., Liang, H., Baxter, S.L., McKeown, A., Yang, G., Wu, X., Yan, F., et al.: Identifying medical diagnoses and treatable diseases by image-based deep learning. Cell **172**(5), 1122–1131 (2018)

34. Huang, G.B., Ramesh, M., Berg, T., Learned-Miller, E.: labeled faces in the wild: a database for studying face recognition in unconstrained environments. University of Massachusetts, Amherst. Technical Report, vol. 07–49 (October 2007)

35. Wasserman, L.: All of Nonparametric Statistics. Springer, New York (2006)

36. Griffiths, E., Assana, S., Whitehouse, K.: Privacy-preserving image processing with binocular thermal cameras. Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. **1**(4), 133:1–133:25 (2018)

37. Suter, Y., Rummel, C., Wiest, R., Reyes, M.: Fast and uncertainty-aware cerebral cortex morphometry estimation using random forest regression. In: International Symposium on Biomedical Imaging (ISBI). IEEE, pp. 1052–1055 (2018)

38. Feng, Y., Wang, S.: A forecast for bicycle rental demand based on random forests and multiple linear regression. In: 16th International Conference Computer and Information Science (ICIS). IEEE, pp. 101–105 (2017)

39. Seber, G.A., Lee, A.J.: Linear Regression Analysis, vol. 329. Wiley, New York (2012)

40. Griffin, G., Holub, A., Perona, P.: Caltech-256 object category dataset, California Institute of Technology, Technical Report 7694. (2007). http://authors.library.caltech.edu/7694

41. Mamaev, A.: Flowers Dataset, Reviewed Dataset from Kaggle. https://www.kaggle.com/alxmamaev/flowers-recognition

42. James, G., Witten, D., Hastie, T., Tibshirani, R.: An Introduction to Statistical Learning. Springer, New York (2013)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Salma Abdel Magid is a Master's Student in the Department of Computer Engineering at Santa Clara University. She received her Bachelor of Science degree in Computer Science in 2017 from SCU. Currently, she works on integrating machine learning with resourceconstrained embedded devices as a research assistant at the SCU's Internet of Things Research Lab. Prior to this, she worked at Abbott (previously Saint Jude Medical) as a Software Engineer on the Nanostim Pacemaker team.
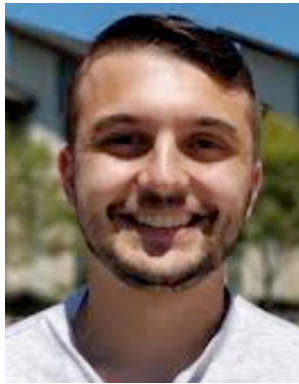
Francesco Petrini is a student at Santa Clara University pursuing a major in Computer Science and Engineering with a minor in Mathematics. He currently works as an undergraduate researcher where his interests are focused primarily on edge and fog computing.

Behnam Dezfouli is an Assistant Professor at the Department of Computer Science and Engineering, Santa Clara University (SCU), US. Before joining SCU, he was a Postdoctoral Research Scientist and a Vising Assistant Professor at the University of Iowa, US, during 2015–2016. He served as a Postdoctoral Research Fellow and a Research Fellow for University Technology Malaysia, and Institute for Infocomm Research, Singapore, during 2014 and 2012, respectively. He received his Ph.D. in Computer Science from University Technology Malaysia in 2014. His research interests include Internet of Things, Cyber-Physical Systems, and Wireless Softwaredefined Networking. He is an Associated Editor of IEEE Communications Letters journal. He is currently the PI of several IoT projects funded by Cypress Semiconductor, City of San Jose, and Santa Clara Valley Water District.

Terms and Conditions