Deep Learning for Inexpensive Image Classification of Wildlife on the Raspberry Pi

Brian H. Curtin
Department of Electrical Engineering & Computer Science
U.S. Military Academy
West Point, NY, USA
brian.h.curtin.mil@mail.mil

Suzanne J. Matthews
Department of Electrical Engineering & Computer Science
U.S. Military Academy
West Point, NY, USA
suzanne.matthews@westpoint.edu

***Abstract*—Animal conservationists need unobtrusive methods of observing and studying wildlife in remote areas. Many commercial options for wildlife observation are expensive, obtrusive, or sub-optimal in remote environments. In this paper, we explore the viability of a Raspberry Pi-based camera system augmented with a deep learning image recognition model for detecting wildlife of interest. Unlike traditional sensor nodes that would have to transmit every captured image, localized image recognition enables only pictures of desired animals to be transferred to the user. For the purposes of this study, we use TensorFlow and Keras to create a convolutional neural network that runs on a Raspberry Pi 3B+. We trained the model on nearly 3,600 images gathered from publicly available image databases that are split into three classes. Our experiments suggest that our system can detect snow leopards with between 74 percent and 97 percent accuracy. We believe that our results show the viability of employing deep learning image recognition models on the Raspberry Pi to create an inexpensive system to observe wildlife.**

***Index Terms*—Raspberry Pi, single board computer, deep learning, convolutional neural network, TensorFlow, Keras**

I. Introduction

Wildlife conservationists often require inexpensive methods for monitoring wildlife. This is especially true in remote areas, where direct human observation is not always possible. In cases where the animal in question is shy of humans, camera-equipped systems allow conservationists and scientists to gain a richer picture of animal habits and behavior. Conservationists currently use a combination of traps, cameras and robots to observe wildlife. However, traps are often inappropriate for megafauna. While robots (such as unmanned ground/aerial vehicles) are a useful substitute for human observers [1], [2], they can inadvertently startle animals and/or cause stress or harm [1], [3]. While camouflage and selecting appropriate material in robotic design can minimize disruption to species [2], this in turn increases the expense of the system.

Modern trail cameras represent a more cost-effective option for observing wildlife and typically cost between $100.00 to $400.00 per camera. Sophisticated models can detect motion and transmit pictures and video over cellular networks. However, such cameras may be inappropriate in remote environments where cellular signal is not available. Most crucially, trail cameras take pictures of whatever triggers the camera, requiring the scientist to manually shift through all pictures to determine if a particular species of interest was detected.

Inexpensive single board computers such as the $35.00 Raspberry Pi offer a more cost-effective option for observing wildlife. Researchers have successfully demonstrated how single board computers like the Raspberry Pi can create cost-effective acoustic monitoring devices. For example, Solo [4] is a bio-acoustic monitoring device built on the Raspberry Pi. Researchers have also used the Raspberry Pi to monitor birds in Kenya [5] and marine mammals off the coast of Brazil [6].

More recently, a Raspberry Pi-based wireless sensor network for observing an area for wildlife was proposed [7], [8]. However, a key weakness of the proposed system is that all captured images are automatically sent to the user. This presents a problem if the images being captured do not contain the animal of interest, forcing the conservationist to sift through extraneous images. Furthermore, excessive transmission of data can deplete battery life.

In this paper, we explore the efficacy of the Raspberry Pi to automatically and visually detect wildlife using deep learning. We focus on snow leopards for the purposes of this project. Snow leopards are especially interesting for conservationists due to their notorious shyness. We build a deep learning model that attempts to classify images as containing snow leopards, humans, or "other" (empty background). To mimic how a conservationist with no domain-based knowledge would build such a model, we deliberately built our model using publicly available software, databases and tools. Our model is based on a convolutional neural network (CNN) built using TensorFlow and Keras. The model is trained using images procured from the publicly available ImageNet [9] and VIPeR [10] databases.

Our experiments show that we can detect snow leopards with at least 74 percent accuracy, and that image classification occurs at acceptable speeds for real-time processing. Our results suggest that a Raspberry Pi-based camera system augmented with a deep learning model can assist in conservation efforts. Since our image recognition technology is built with free software and tools, adding a deep learning model to the

Funding is provided by U.S. Army Futures Command, CCDC Armaments

Fig. 1. Raspberry Pi 3B+ single board computer.

Pi does not increase the overall cost of the system. We hope our work will assist conservationists interested in exploring the Raspberry Pi for camera-based monitoring of wildlife.

II. Background and Related Work

The Raspberry Pi (Figure 1) is a popular single board computer that was initially released in 2012 and has been widely used for numerous applications, including computer science education [11]–[14], robotics [15], sensor networks [7], [8], [15] and internet-of-things [16]. More recently, researchers have begun to explore the Raspberry Pi for image recognition.

For example, researchers [17], [18] successfully deployed the Daugman algorithm [19] for iris detection on Raspberry Pis. However, the Daugman algorithm itself does not depend on machine learning. Researchers have also used Raspberry Pis extensively with the Open Source Computer Vision (OpenCV) package and Adaptive Boosting (AdaBoost) for various applications, include augmenting street lights with object detection [20], traffic sign recognition [21], face recognition [22], and human emotion recognition [23]. Raspberry Pis have also been used for cloud-based image recognition for use in law enforcement [24]. The Raspberry Pi in the aforementioned project transfers captured images to the cloud, which serves as the actual site of image recognition.

Unlike most prior work, we use a convolutional neural network (CNN) to generate our image recognition model. While CNNs are an extremely popular for image classification [25], [26], there is very little prior work deploying CNNs on Raspberry Pis, largely due to the limited processing power of older models of the Raspberry Pi. To mitigate this issue, researchers typically train their CNN on a separate (more powerful) machine prior to off-loading the model onto the Raspberry Pi for use. For example, researchers from Brazil [27] explored CNNs for aerial person detection on the Raspberry Pi 2. While CNNs showed substantial promise, the researchers concluded that the Raspberry Pi's processing power was insufficient for many of the applications they explored [27]. Another set of researchers [28] compared different types of CNNs on the Raspberry Pi 2 for traffic sign classification. The researchers concluded that a CNN built with a combination of TensorFlow [29], [30] and Keras [31] yielded the best results on the Raspberry Pi. However, they also cautioned readers that the processing power of the Raspberry Pi 2 limited its ability support memory-intensive models [28].

In this paper, we deliberately use free or widely available components to mimic how a conservationist will use the system. For example, the use of the inexpensive Raspberry Pi as a platform drastically reduces the cost of the camera sensor system, making it easy for scientists to cheaply deploy units. Based on the results of prior work, we choose the Raspberry Pi Model 3B+ (shown in Figure 1) as our single board computer of choice. The Raspberry Pi 3B+ has a 1.4 GHz ARM A53 Processor, 1 GB of RAM and integrated wireless and Bluetooth capabilities.

The wide availability of open-source materials make it possible for conservationists to train their own image recognition models for Raspberry Pis. We choose TensorFlow [29], [30] and Keras [31] for their relative ease of use, freely availability, and (as of August 2018) native support on the Raspberry Pi. TensorFlow implements a variety of machine learning and deep learning algorithms (including CNNs), and allows users to access a variety of tools in order to create models and deploy machine learning robustly and easily. Keras is a high-level neural network API that runs on top of TensorFlow and allows users with little experience to quickly code and create models using different sources of data.

We train the model using pictures from the publicly available ImageNet [9], [32] and VIPeR [10] databases. ImageNet [9], [32] is a well established image database for learning applications that is curated by Stanford University. ImageNet sorts images by category, with each category containing hundreds to thousands of pictures. The repository allows users to easily find, download, and utilize large numbers of images for use in supervised learning of different classes. The Viewpoint Invariant Pedestrian Recognition (VIPeR) [10], [33] is an image data set containing pictures of pedestrians (human) from different viewpoints and under varying lighting conditions. The data set is provided by the University of Santa Cruz to evaluate the performance of models in modern surveillance systems. We strongly believe that scientists can reproduce our results and cheaply train their own models by using the database and procedures described in this paper.

III. Methods

To mimic how a conservationist would build the model, we use an online tutorial [34] to build a CNN. The tutorial includes code that allows for dynamic creation of an image recognition model using a CNN with two convolutional layers. For the purposes of this study, we focus on distinguishing between snow leopards and humans. We also have a third category ("other") that accounts for situations where the Raspberry Pi detects motion, but the captured image contains nothing of interest.

At a high level, the code first creates a sequential model and adds a 2-D convolution layer with 32 feature layers and a 3×3 feature detector with a stride length of 1. A second 2-D convolutional layer with 64 feature layers with a 3×3 feature detector is added with a rectified linear unit (ReLU)

Fig. 2. Sample images used for training (human, snow leopard, and other).

activation function. Max pooling is added to both layers to reduce the total set of features. The layers are flattened and used as input to a linear/dense layer with 128 input units and a ReLU activation function. The model then drops 20 percent of the inputs and is fed into a last linear/dense output layer. The last layer uses a softmax activation function to support three different dimensions (representing "snow leopard", "human" and "other"). The model is compiled with the ADADELTA optimizer [35] and a categorical cross-entropy loss function, where the latter is used to support the three image classes.

We generate a dataset of images using $1,262$ pictures of humans pedestrians from the VIPeR database [10], and $1,568$ pictures of snow leopards and 768 pictures of natural background environments (for the "other" category) procured from ImageNet [9] database. The natural backgrounds used for our classification are mainly green forest, since most of the collected snow leopard pictures are from zoos, where they are typically kept in forest-like enclosures. The collected images are split into three categories (classes): "snow leopard", "human" and "other". Figure 2 shows an sample image from each of the three image classes.

The collected image sets are split into three further sets: a training set containing 80 percent of each category, a testing set containing 10 percent of the images from each category, and a validation set containing the remaining 10 percent of images. There is no overlap between the images in the training, testing and validation sets. A training data generator is instantiated with a random variety of image sheer, rescaling, zooming, and horizontal flipping to make the training images more diverse. A test data generator is also instantiated and linked to the directory containing the test images. Unlike the training images, the test images are only re-scaled to fit the model and are not modified any further. The model is trained and tested over 10 epochs with 250 steps and 150 validation steps.

We build the model on a Dell Latitude 7350 laptop that has a 64-bit Intel(R) Core(TM) M-5Y71 1.2 GHz CPU, 8 GB of RAM and runs the Windows 10 operating system. Anaconda [36], a free popular Python package for scientific computing, is used in conjunction with the TensorFlow and Keras packages. We also use the Jupyter [37] package to create image recognition models easily with the Keras API and TensorFlow back-end. The model takes about 35 minutes to build and is approximately 19 MB in size. Lastly, we export the completed model to the Raspberry Pi 3B+ for experimentation. We note that as an additional step, TensorFlow and Keras need to be installed on the Raspberry Pi.

$$Accuracy = \frac{TruePositive + TrueNegative}{Total} \quad (1)$$

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \quad (2)$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \quad (3)$$

We measure the quality of our results using accuracy, precision, and recall. Accuracy (Equation 1) is the number of correct identifications divided by the total number of predictions. Precision (Equation 2) is the total number of correct positive identifications of an class divided by the total predicted positive identifications of that class, both correct and incorrect. Lastly, Recall (Equation 3) or Sensitivity is the total number of correct positive identifications of a class divided by the total number of elements in that class.

IV. Results

To assess the efficacy of the approach, we run two different set of experiments on the Raspberry Pi. In both sets, we run the model on the Raspberry Pi 3B+ against images in the validation set to classify each image as either "snow leopard", "human" or "other". In the first set of experiments, we download images in the validation set directly to the Raspberry Pi 3B+ prior to classification; this set of experiments is referred to as "pre-downloaded". In the second set of experiments, we attach an Adafruit Raspberry Pi Camera [38] to the Raspberry Pi to capture "live" images and classify each image in real-time. This second set of experiments is referred to as "live capture". In both sets of experiments, the model prints the classification to the console. We note that in a fully integrated system, logic can be added to enable the Pi to automatically delete, save, or transmit images to the user. Lastly we project the cost to build the Raspberry Pi camera sensor node, measure its power usage, and estimate its battery life.

A. Pre-downloaded image experiments

In our first set of experiments, we test the model against the validation set of 122 images of humans, 150 images of snow leopards, and 75 images of natural backgrounds for a total of 347 images. The model is tested on the Raspberry Pi 3B+ and the corresponding results are shown in Table I. In this intial set of experiments, we are able to achieve 97 percent accuracy in identifying snow leopards, 99 percent accuracy in identifying human subjects, and 96 percent accuracy in identifying empty background pictures. Recall is very high across all three classes, while precision is very high for snow leopards and humans.

While our initial set of results are promising, we notice that there are several limitations to this experiment. First, the VIPeR images in the human validation set consist of direct front and side profiles and people without any background.

TABLE I
PRE-DOWNLOADED IMAGE TESTING RESULTS

	Snow Leopard	Human	Other
Accuracy	0.97	0.99	0.96
Precision	0.99	1.00	0.85
Recall	0.94	0.96	0.99

Fig. 3. An image the model incorrectly classifies at "other".

In contrast, the "snow leopard" and "other" categories have images with greater diversity, including different lighting conditions and larger backgrounds. As a result, when the model misclassifies images, it tends to group them in the "other" category. We note that the "other" category has the lowest precision value at 85 percent.

Figure 3 gives an example of an image that our model misclassifies as "other". In this particular image, the snow leopard is hiding behind a tree branch. We notice that our model struggles to identify images that are majority green foliage or in cases where the subject is not taking up the majority of the image frame.

We measure the amount of time image classification takes on the Raspberry Pi on the pre-downloaded image set. As noted in Section II, much prior work [27], [28] expressed concern about the limited processing power of older Raspberry Pi models. However, our results over five runs show that it takes only 29.2 seconds on average to classify 347 images on the Raspberry Pi 3B+. We note that roughly 55 percent of that time is devoted to starting up Tensorflow and Keras on the Pi and loading the model. Once the model is loaded and the TensorFlow/Keras backend starts up, it takes only 13 seconds to classify all 347 images, or roughly 0.03 seconds per image. The alacrity at which images are classified suggests that the Pi 3B+ has sufficient processing power for real-time classification of images.

B. "Live capture" image experiments

We also test the model using "live" image capturing using an Adafruit Raspberry Pi camera. Again, all experiments are conducted on a Raspberry Pi 3B+. As actual snow leopards are unavailable for our assessment, we simulate "live" capture by printing out 40 images from each class of the validation set and holding each up to the camera at a distance of approximately 4 inches. The output of the decision is recorded for each image and Table II shows the accuracy, precision and recall achieved.

TABLE II
"LIVE" IMAGE TESTING RESULTS

	Snow Leopard	Human	Other
Accuracy	0.74	0.77	0.72
Precision	0.57	0.95	0.63
Recall	0.97	0.41	0.53

On the "live capture" set of experiments, we achieve 74 percent accuracy for snow leopards, 77 percent accuracy with humans, and 72 percent accuracy for the "other" category. Snow leopards have the highest recall with 97 percent, but the lowest precision value, suggesting that there were humans or empty background that were categorized as snow leopards. Conversely, the human class had a very high precision value, but low sensitivity, reflecting that many were misclassified as being snow leopards or other. In a real-world scenario (where the Pi is set to transmit all images classified as snow leopards), we hypothesize that the system would transmit most snow leopard pictures, plus several additional images that were incorrectly classified as snow leopards.

There are several limitations to this experiment. First, the model is attempting to classify pictures of print-outs of collected images rather than raw subjects. The loss in resolution makes it harder for the model to identify colors and lines as accurately as if it were dealing with an actual subject. Furthermore, the resolution of the pictures taken by the Adafruit Raspberry Pi camera [38] may be of lower quality than those used to train the model. However, we note that higher quality cameras are available for the Raspberry Pi such as the Pimoroni CAM008 [39]. Lastly, the images held up to the camera may not have been perfectly flush with the limits of the camera causing it to analyze the background of the paper which may have led to misidentification. Field testing is ultimately needed to fully assess the quality of a Raspberry Pi camera sensor node.

Lastly, we also measure the amount of time "live" testing takes on the Raspberry Pi 3B+. The system, once started, takes a picture every second and runs it against the loaded model for classification. This process continues in a loop until the classification mechanism is exited or the system is shut down. Consistent with our pre-loaded images experiments, loading the model and starting up TensorFlow/Keras takes up the majority of the time, or approximately 16.3 seconds. However, we note that this is a one-time start-up cost. Once the camera starts taking pictures, the classification process is instantaneous, taking roughly 0.12 seconds to classify each picture. Our experiment, while imperfect, reinforces our hypothesis that the Raspberry Pi 3B+ is capable of doing real-time image classification in the wild.

C. Projected Cost of Sensor Node

The cost to build our Raspberry Pi-based camera system is shown in Table III. The proposed camera system consists

TABLE III
ESTIMATED COST OF RASPBERRY PI CAMERA SYSTEM

Item	Cost
Raspberry Pi 3B+	$35.00
Raspberry Pi Camera Module	$5.00
16 GB microSD Card	$4.50
4 AA Batteries	$2.00
AA Battery Holder with Micro USB Cable	$3.00
Total	**$49.50**

of a Raspberry Pi 3B+, a Raspberry Pi camera module, a 16 GB microSD card, and a power source such as four AA batteries in a holder to provide power. The total estimated cost of the camera sensor node is $\$50.00$. This is a more realistic option for mass deployment of cameras compared to the trail cameras which can cost upwards of $\$100.00$. Using our method, conservationists can deploy two Raspberry Pis with image detection capability for every trail camera that does not have any image recognition capability.

We measured the power consumption of our camera system using a KillAWatt [40]. The Raspberry Pi running without image recognition consumes 0.08 Amperes of current. When it is taking pictures and running an image recognition model (as in the "live" testing environment) it consumes 0.10 Amperes. With a power supply of four Alkaline AA batteries ($\approx 2,000$ milliampere hours each), the Raspberry Pi is estimated to last for 10 hours in "live" mode. This is long enough for a single day or night of observation. We note that the Raspberry Pi will likely last longer with more expensive rechargeable batteries, enabling conservationists to leave the camera systems isolated for longer periods of time. A weather-proof enclosure can be constructed from readily available parts, and there are tutorials [41] online to aid a conservationist in creating one.

V. CONCLUSIONS AND FUTURE WORK

This paper discusses the construction of a deep learning image recognition model to aid with inexpensive wildlife conservation efforts in remote environments. Our prototype is created using open source software and methods and leverages the $\$35.00$ Raspberry Pi 3B+ single board computer to make deployment as inexpensive as possible. To reproduce our work, a conservationist needs only a laptop and the necessary hardware (Table III) to assemble a Raspberry Pi camera sensor node. We use a simple convolutional neural network whose code is available online [34] to build our model, reflecting the process that a conservationist with no domain knowledge of machine learning would follow. Despite the relative simplicity of our model, the Raspberry Pi 3B+ detects snow leopards with 97 percent accuracy with images from the validation set downloaded onto the Pi, and 74 percent accuracy when tested with "live" capture from an attached Adafruit Pi camera. Each camera sensor node costs approximately $\$50.00$ to build. Our power experiments suggest that the unit can be powered with four AA batteries over a ten hour period.

We note that our work is not without limitations. Currently, our model is trained to recognize humans, snow leopards, and empty backgrounds reflecting an outdoor, wooded environment containing mostly green foliage. Almost all the snow leopard pictures were taken inside of zoos or enclosed areas which keep the snow leopards in such environments. However, the natural environment of snow leopards is mountainous, rocky and mostly barren. It is very difficult to observe snow leopards in their natural habitat so the number of images available is limited. We also note that the model had difficulty recognizing snow leopards if the animal was obscured or far away, as shown in Figure 3. Lastly, better battery life can likely be obtained by integrating the system with a motion sensor, so that the camera only begins recording when motion is detected.

However, we believe our results are very promising. The use of open source deep learning tools in conjunction with the Raspberry Pi suggests that any conservationist can use our methods on their personal device to create a custom image recognition model designed for animals they are interested in. They can also tailor the backgrounds to resemble what they would likely encounter in the environment they are deploying the cameras in. Lastly, they can add their own photos into the machine learning algorithm making for a better model. Our run-time experiments also show that the latest model of the Raspberry Pi (3B+) has sufficient processing power to classify images in real-time. While starting up TensorFlow/Keras and model loading can take nearly 30 seconds, this is a one-time cost. Image classification itself takes a fraction of a second to complete.

There are many avenues of future work. First, we plan on improving our model by augmenting it with object detection techniques to focus on animals of interest, reducing the potential of misclassification when animals are far away or hiding. We also plan to augment our model to detect additional classes in more varied environments and add motion sensing to help further conserve battery life. Most importantly, we hope to get our model working on the recently released Raspberry Pi Zero [42]. The challenge with this device is its extremely limited memory and processing power, which severely limits the size and complexity of models that can be loaded on it. However, the Raspberry Pi Zero only costs $\$10.00$, consumes a fraction of the power of the Raspberry Pi 3B+, and would represent an even more inexpensive option for conservationists.

Lastly, while this paper has focused on applications in conservation efforts, we note that our results have implications for any problem requiring inexpensive real-time image recognition. We strongly believe that our results suggest that CNNs can be successfully used for real-time image recognition on Raspberry Pis, and represent a useful tool for conservation.

ACKNOWLEDGMENT

This paper summarizes the work of an honors project completed by the first author, while he was an undergraduate student at the U.S. Military Academy. The second author was the faculty advisor on the project. Both authors contributed to the writing of the paper. The views expressed in this paper

are those of the authors and do not reflect the official policy or position of the Department of the Army, Department of Defense or the U.S. Government.

REFERENCES

[1] D. Grémillet, W. Puech, V. Garçon, T. Boulinier, and Y. Le Maho, "Robots in ecology: welcome to the machine," *Open Journal of Ecology*, vol. 2, no. 02, p. 49, 2012.

[2] A. van Wynsberghe and J. Donhauser, "The dawning of the ethics of environmental robots," *Science and engineering ethics*, vol. 24, no. 6, pp. 1777–1800, 2018.

[3] M. A. Ditmer, J. B. Vincent, L. K. Werden, J. C. Tanner, T. G. Laske, P. A. Iaizzo, D. L. Garshelis, and J. R. Fieberg, "Bears show a physiological but limited behavioral response to unmanned aerial vehicles," *Current Biology*, vol. 25, no. 17, pp. 2278–2283, 2015.

[4] R. C. Whytock and J. Christie, "Solo: an open source, customizable and inexpensive audio recorder for bioacoustic research," *Methods in Ecology and Evolution*, vol. 8, no. 3, pp. 308–312, 2017.

[5] C. wa Maina, D. Muchiri, and P. Njoroge, "Cost effective acoustic monitoring of biodiversity and bird populations in kenya," *bioRxiv*, p. 072546, 2016.

[6] M. Caldas-Morgan, A. Alvarez-Rosario, and L. R. Padovese, "An autonomous underwater recorder based on a single board computer," *PloS one*, vol. 10, no. 6, p. e0130297, 2015.

[7] B. H. Curtin, R. H. David, E. D. Dunham, C. D. Johnson, N. Shyamkumar, T. A. Babbitt, and S. J. Matthews, "Designing a raspberry pi sensor network for remote observation of wildlife," in *Proceedings of the 6th Annual Symposium on Hot Topics in the Science of Security*, ser. HotSoS '19. New York, NY, USA: ACM, 2019, pp. 17:1–17:2.

[8] A. Alejos, M. Ball, C. Eckert, M. Ma, H. Ward, P. Hanlon, and S. J. Matthews, "Exploring the raspberry pi for data summarization in wireless sensor networks: Poster," in *Proceedings of the 5th Annual Symposium and Bootcamp on Hot Topics in the Science of Security*, ser. HoTSoS '18. New York, NY, USA: ACM, 2018, pp. 18:1–18:1.

[9] Stanford Vision Lab, "Imagenet," 2016, http://www.image-net.org/ (accessed May 10, 2019).

[10] Jack Baskin School of Engineering, University of California Santa Cruz, "Viewpoint invariant pedestrian recognition," 2007, https://vision.soe.ucsc.edu/node/178 (accessed May 12, 2019).

[11] J. R. Byrne, L. Fisher, and B. Tangney, "Computer science teacher reactions towards raspberry pi continuing professional development (cpd) workshops using the bridge21 model," in *2015 10th International Conference on Computer Science Education (ICCSE)*. Cambridge, UK: IEEE, July 2015, pp. 267–272.

[12] J. Kawash, A. Kuipers, L. Manzara, and R. Collier, "Undergraduate assembly language instruction sweetened with the raspberry pi," in *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, ser. SIGCSE '16. New York, NY, USA: ACM, 2016, pp. 498–503.

[13] X. Zhong and Y. Liang, "Raspberry pi: An effective vehicle in teaching the internet of things in computer science and engineering," *Electronics*, vol. 5, no. 3, 2016.

[14] S. J. Matthews, J. C. Adams, R. A. Brown, and E. Shoop, "Portable parallel computing with the raspberry pi," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '18. New York, NY, USA: ACM, 2018, pp. 92–97.

[15] S. J. Matthews, "Harnessing single board computers for military data analytics," in *Military Applications of Data Analytics*, K. Huggins, Ed. CRC Press, Taylor & Francis, 2018, ch. 4, pp. 63–77.

[16] M. Maksimović, V. Vujović, N. Davidović, V. Milošević, and B. Perišić, "Raspberry pi as internet of things hardware: performances and constraints," *design issues*, vol. 3, no. 8, 2014.

[17] F. R. G. Cruz, C. C. Hortinela, B. E. Redosendo, B. K. P. Asuncion, C. J. S. Leoncio, N. B. Linsangan, and W. Chung, "Iris recognition using daugman algorithm on raspberry pi," in *2016 IEEE Region 10 Conference (TENCON)*, Nov 2016, pp. 2126–2129.

[18] Z. Kunik, A. Bykowski, T. Marciniak, and A. Dbrowski, "Raspberry pi based complete embedded system for iris recognition," in *2017 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*, Sep. 2017, pp. 263–268.

[19] J. Daugman, "How iris recognition works," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 1, pp. 21–30, Jan 2004.

[20] J. A. Galindo and M. V. Caya, "Development of street lighting system with object detection," in *2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology,Communication and Control, Environment and Management (HNICEM)*, Nov 2018, pp. 1–5.

[21] E. Bilgin and S. Robila, "Road sign recognition system on raspberry pi," in *2016 IEEE Long Island Systems, Application and Technology Conference (LISAT)*. IEEE, April 2016.

[22] A. Ben Thabet and N. Ben Amor, "Enhanced smart doorbell system based on face recognition," in *2015 16th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, Dec 2015, pp. 373–377.

[23] Suchitra, Suja P., and S. Tripathi, "Real-time emotion recognition from facial images using raspberry pi ii," in *2016 3rd International Conference on Signal Processing and Integrated Networks (SPIN)*, Feb 2016, pp. 666–670.

[24] M. Sajjad, M. Nasir, K. Muhammad, S. Khan, Z. Jan, A. K. Sangaiah, M. Elhoseny, and S. W. Baik, "Raspberry pi assisted face recognition framework for enhanced law-enforcement services in smart cities," *Future Generation Computer Systems*, 2017.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[26] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[27] D. C. D. Oliveira and M. A. Wehrmeister, "Towards real-time people recognition on aerial imagery using convolutional neural networks," in *2016 IEEE 19th International Symposium on Real-Time Distributed Computing (ISORC)*, May 2016, pp. 27–34.

[28] C. F. Silva and C. A. Siebra, "An investigation on the use of convolutional neural network for image classification in embedded systems," in *2017 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, Nov 2017, pp. 1–6.

[29] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[30] TensorFlow, "Tensorflow: An end-to-end open source machine learning platform," 2016, https://www.tensorflow.org/ (accessed May 10, 2019).

[31] F. Chollet *et al.*, "Keras," 2015, https://keras.io (accessed May 10, 2019).

[32] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. IEEE, 2009, pp. 248–255.

[33] D. Gray, S. Brennan, and H. Tao, "Evaluating appearance models for recognition, reacquisition, and tracking," in *Proc. IEEE International Workshop on Performance Evaluation for Tracking and Surveillance (PETS)*, vol. 3, no. 5. Citeseer, 2007, pp. 1–7.

[34] R. Khandelwal, "Building powerful image classification convolutional neural network using keras," 2019, https://medium.com/datadriveninvestor/building-powerful-image-classification-convolutional-neural-network-using-keras-a1839d0ff298 (last accessed May 10, 2019).

[35] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[36] Anaconda Inc., "Anaconda," 2012, https://www.anaconda.com/ (accessed May 10, 2019).

[37] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. E. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. B. Hamrick, J. Grout, S. Corlay *et al.*, "Jupyter notebooks-a publishing format for reproducible computational workflows." in *ELPUB*, 2016, pp. 87–90.

[38] Ebay, "Camera module board 5mp webcam video 1080p 720p for raspberry pi," 2019, https://www.ebay.com/i/381181830171?chn=ps (accessed Sep. 27, 2019).

[39] D.-K. Electroncics, "Pimoroni cam008," 2019, https://www.digikey.com/product-detail/en/pimoroni-ltd/CAM008/1778-1218-ND/9521978 (accessed May 14, 2019).

[40] P3 International, "Kill a watt meter," 2018, http://www.p3international.com/products/p4400.html (accessed May 10, 2019).

[41] South Somerset Weather. (2013) Raspberry pi trail camera. [Online]. Available: http://www.afraidofsunlight.co.uk/weather/trailcam_info.html

[42] Raspberry Pi Foundation, "Raspberry pi zero," 2018, https://www.raspberrypi.org/products/raspberry-pi-zero/ (accessed May 12, 2019).