*electronics*

Article

# Low-Cost Image Search System on Off-Line Situation

**Mery Diana [1,2,*], Juntaro Chikama [2], Motoki Amagasaki [2] and Masahiro Iida [2]**

[1] Agency for the Assessment and Application of Technology, Jakarta 10340, Indonesia
[2] Graduate School of Science and Technology, Kumamoto University, Kumamoto 860-8555, Japan; chikama@arch.cs.kumamoto-u.ac.jp (J.C.); amagasaki@cs.kumamoto-u.ac.jp (M.A.); iida@cs.kumamoto-u.ac.jp (M.I.)
* Correspondence: mery@arch.cs.kumamoto-u.ac.jp

Received: 15 November 2019; Accepted: 10 January 2020; Published: 14 January 2020

**Abstract:** Implementation of deep learning in low-cost hardware, such as an edge device, is challenging. Reducing the complexity of the network is one of the solutions to reduce resource usage in the system, which is needed by low-cost system implementation. In this study, we use the general average pooling layer to replace the fully connected layers on the convolutional neural network (CNN) model, used in the previous study, to reduce the number of network properties without decreasing the model performance in developing image classification for image search tasks. We apply the cosine similarity to measure the characteristic similarity between the feature vector of image input and extracting feature vectors from testing images in the database. The result of the cosine similarity calculation will show the image as the result of the searching image task. In the implementation, we use Raspberry Pi 3 as a low-cost hardware and CIFAR-10 dataset for training and testing images. Base on the development and implementation, the accuracy of the model is 68%, and the system generates the result of the image search base on the characteristic similarity of the images.

**Keywords:** low-cost system; edge site; CNN; general average pooling layer (GAP); cosine similarity; image search

## 1. Introduction

Recently, many researchers have investigated in the merit of edge computing. Edge computing [1] becomes one of the optimizing solutions to reduce network latency and save bandwidth in cloud computing [2–7]. Since the edge device is near or close to the source of data, such as a sensor, it gives a chance for real-time processing [8]. Scalability and privacy are also other benefits of edge computing to avoid bottleneck because of connecting devices to the cloud and keeping the privacy of user data from public internet usage [9].

The implementation of deep learning in edge devices is challenging. Since the computation of deep learning requires high computing performance [10], edge computing gives many researchers another chance to run the deep learning model in a limited resource such as the edge device. The complexity of the model will increase the memory usage and execution time, which in turn increases power consumption. As a result of this condition, a deep learning model should fit with the limitation of memory on the microcontroller and limitation of resource in the edge device for computing the task such as real-time classification processing [11]. Another solution to tackle this issue is by reducing the size or number of parameter usages in building the deep learning model [12].

One of the popular deep learning techniques in classification is the convolutional neural network (CNN). CNN shows excellent performance in image classification [13,14] because it captures the high-level input data represented by the implementation of several convolutional filters. In the previous study [15], we developed a classical CNN model in order to characterize the similarity between tiny images from the CIFAR-10 dataset. Classifying the tiny image, such as the CIFAR-10 dataset, is also challenging to prove the performance of the CNN as an image classifier. Small images, such as the CIFAR-10 dataset, do not provide a high-quality image, but show a cleanly labeled subset [16]. In image classification for image search task, cosine similarity is used to filter the most similar images. By calculating the distance between feature vectors from testing images with the reference feature vectors generated from the classical CNN model, the system could characterize over 90% for a similar image.

Considering the processing power, energy storage, and memory capacity in edge devices [5,17,18], decreasing the number of neural network properties will decrease the hardware resource usage in system implementation. The low complexity of the CNN model will be running in an embedded processor such that the microcontroller in the edge device becomes an important point. Since the CNN layer uses stacks of the layer with a different function to support the classification task, applying the general average pooling (GAP) layer will reduce the number in the network because the GAP has fewer parameters than fully connected layers which will thus reduce computational load and overfitting risk [19–21].

In this study, we propose a deep learning model to address these challenges. We build a CNN model using the GAP layer to reduce the parameter resource since the system will be running on a low-cost edge device such as Raspberry Pi 3. The GAP layer will replace two fully connected layers in the CNN model [15]. We also propose the image search task by using the CIFAR-10 dataset for training and testing images to show how the proposed model works on the edge device. Then we use cosine similarity to define the most similar images from our database and show the result of an image search task on the Raspberry Pi 3 display.

## 2. Related Works

Considering the benefits of computing in edge devices, several researchers have studied the implementation of deep learning in edge sites. A study by Neto et al. [22] classified edge devices for supporting data processing. They classified the device into five classes in which the third and fourth classes are suitable to run the machine learning algorithm since its powerful GPU (Graphics Processing Unit) is in parallel processing of neural networks. This study conducted real experiments to obtain the performance of each device in processing data from sensors and reduce network delays and bandwidth using Raspberry Pi series and BeagleBone. As a result, in the smart pole application, the response time of MySQL run in the server was similar to the response time of SQLite on Raspberry Pi 2 and BeagleBone. Another experiment in this study is the smart place project in which an automatic air conditioner is built using CNN MobileNet on different computers like a notebook, Raspberry Pi Zero, Raspberry Pi 2, and Raspberry Pi 3. As a result, Raspberry Pi 3 shows that response time is faster than another Raspberry Pi series.

Li, Zhou, and Chen [2] built Edgent to prove the capability of an edge device to run deep learning techniques for specific applications. Edgent is a deep neural network (DNN) framework in the edge device. This study implements the image recognition application using CIFAR-10 as the dataset and AlexNet as the model on Raspberry Pi 3. As a result, Edgent shows the effectiveness of enabling on-demand in edge devices. Another related study by Gauswami and Trivedi [23] discusses the implementation of deep learning techniques on Raspberry Pi. This study uses the CNN for gender detection. The result shows that the purposed system is working on various challenging levels of datasets and gives the best performance in gender detection for each database.

Several studies were conducted to evaluate the CNN as a neural network model in low-edge devices such as the Raspberry Pi platform. A study by Foley and O'Reilly [24] aimed to discover how the object detection algorithm operates in a low-end device such as Raspberry Pi. They compared three CNN algorithms for object detection, namely SSD MobileNet, Inception v2, and Tiny YOLO. Results showed that the lower number of CNN gave a lower computational overhead, then decreased the computation time of the image processing. Nikouei et al. [25,26] proposed a lightweight convolutional neural network (L-CNN) by using the depth wise separable convolution. The aim of the L-CNN was to reduce the number of parameters in the network without affecting the quality of the output so that it could fit with edge devices such as Raspberry Pi. The result shows that the L-CNN achieved a satisfactory frame per second value and met the goals of the design.

## 3. Classification Using Image Data

### *3.1. Overview of Our System*

In this study, we purpose a deep learning technique using the CNN model that has modified by using the general average pooling layer to reduce the number the parameter in the learning process. This system aims to work on low power devices such as Raspberry Pi 3 and uses Keras [27] and Tensor flow [28] as a design framework. For the implementation of the system, we applied cosine similarity measurement to characterize a similar image from CIFAR-10 [29] as a result of the image searching task. Training and testing images use the CIFAR-10 dataset.

### *3.2. CIFAR-10 Dataset*

Recently, many applications of machine learning have used the CIFAR-10 dataset, especially for supervised learning processes. Image recognition, image classification, and image detection use CIFAR-10 to evaluate the system performance of deep learning models for tiny images. CIFAR-10 has ten image label categories. Every label has 6000 images for each class. The representative images for each class and mapping label feature of CIFAR-10 are shown in Figure 1 and Table 1 as follows.

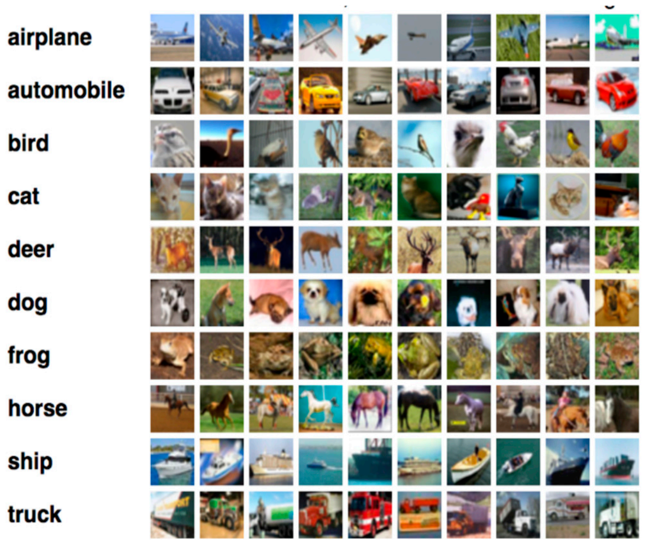


**Figure 1.** CIFAR-10 image example [29].

**Table 1.** Mapping the label feature of CIFAR-10 [15].

| Label. | Category |
|---|---|
| 0 | Airplane |
| 1 | Automobile |
| 2 | Bird |
| 3 | Cat |
| 4 | Deer |
| 5 | Dog |
| 6 | Frog |
| 7 | Horse |
| 8 | Ship |
| 9 | Truck |

CIFAR-10 dataset contains 50,000 training images and 10,000 testing images. We used the mapping of each class category in CIFAR-10 as a label in the data processing. Then we also used an image from the CIFAR-10 dataset as the input of the image search task.

## *3.3. CNN Model*

### 3.3.1. Small CNN for Raspberry Pi 3

The convolutional neural network (CNN) is a technique in supervised learning. CNN consists of several convolutional layers as structures and requires an amount of dataset in the training process to achieve the best accuracy. In many studies, the CNN shows better performance in image classification [30–32].

Since the system will run in low power devices such as Raspberry Pi 3, the number of parameters and the stack of layers have to reduce in the CNN model. In our previous study, we built a classical CNN model [15]. Table 2 shows the model architecture of the classical CNN model and the small CNN model. The previous model consists of two 2D convolutional layers for generating the feature vector, three fully connected layers, and the Softmax layer as the classifier.

**Table 2.** Classical convolutional neural network (CNN) [15] vs. small CNN.

| **Classical CNN [12].** | **Small CNN** |
|---|---|
| conv2d-1, ReLU, bnorm1<br>maxpooling2d-1<br>conv2d-2, ReLU, bnorm2<br>maxpooling2d-2<br>FC1, ReLU, dropout-1<br>FC2, ReLU, dropout-2<br>FC3<br>Softmax | conv2d-1, ReLU, bnorm1<br>maxpooling2d-1<br>conv2d-2, ReLU, bnorm2<br>maxpooling2d-2<br>General Average Pooling (GAP)<br>FC3<br>Softmax |

In this study, we proposed a small CNN, which is appropriate with the Raspberry Pi 3 capability as an edge device. As shown in Table 2 above, the main layers of the architecture consist of two 2D convolutional layers, one layer of general average pooling, and one layer of a fully connected layer. The GAP layer replaces the two fully connected layers in the previous model.

### 3.3.2. Global Average Pooling

Global average pooling is one of the most well-known pooling used in the CNN model. Generally, pooling layers in convolutional layers aim to strengthen the translational invariance and reduce the dimension of the feature maps [21]. In image classification of the Caltech101 dataset, global average pooling obtains excellent accuracy [33].

General average pooling is utilized to replace the fully connected layer, which is prone to overfitting. In the classification task using the CNN model, the feature map generated by the convolutional filter becomes the input to a fully connected layer and Softmax layer as the classifier of the output. Since fully connected layers are prone to overfitting, Lin, Chen, and Yan [20] used global average pooling to replace them. The feature map as the output of the general average pooling was fed to the Softmax layer by taking the average of feature maps from the output of the convolutional layers.

## *3.4. Database*

### 3.4.1. SQLite

In the application of deep learning technologies such as the CNN in edge devices, we have to consider the lite database to store essential data and key information. The database should be fit with edge device capacity. Moreover, the purposed system will work as the off-line system in the image search task, and all processing of the database will run on edge devices without connection to other means of access such as internet connectivity to the cloud. SQLite becomes one of the solutions in edge site implementation because it is light. The SQLite is the library based embedded DBMS, which can integrate easily with the existing source code [34]. It contains simple data types and has dynamic querying capability [35], so it could operate smoothly. We use the SQL function to store the primary data and build the database.

Figure 2 shows the block diagram of SQLite. By using the SQL command processor, SQLite becomes convenient for designing the lite database. In this study, we used the SQLite library that is available in the python environment. Since SQLite is a serverless engine [36], it tremendously fits for small embedded hardware applications such as Raspberry Pi 3. Applying SQLite in the application will reduce overhead related to network calls and simplify database administration [37]. After developing the database, we wrote and read the data on the database easily without server configuration.

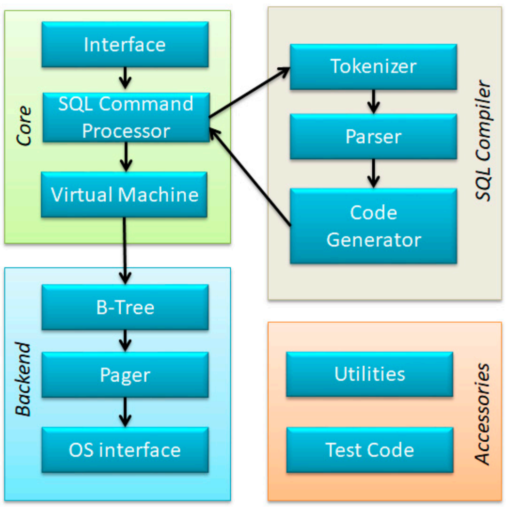


**Figure 2.** Block diagram of SQLite [36].

### 3.4.2. Database Construction

In our system, we have several data including original files that contain the generated feature vectors from testing images in the model and raw images of each class from the CIFAR-10 dataset. These data and files are stored in the database. Since we used the lite database, the size is reasonable with the Raspberry Pi 3 memory allocation.

We constructed the database, as shown in Table 3. We used id, name, feature, and raw_image as the title for each row in the database. Id becomes the primary key for the database, feature contains the generated feature vectors from the testing image, and raw_image contains the original image from CIFAR-10 as feature vectors that were generated by the small CNN model. These files are stored in the database as a blob type to support the lite storage and off-line system.

**Table 3.** Database construction.

| id. | Name | Feature | raw_image |
|---|---|---|---|
| 1 | aeroplane_s_000002.png | b'5.46540737152099xxxxx... | b'\x89PNG\r\n\x1a\n\x00\xx.. |
| 2 | aeroplane_s_000040.png | b'-5.6080293655395xxxxx ... | b'\x89PNG\r\n\x1a\n\x00\xx.. |
| 3 | aeroplane_s_000045.png | b'-2.4921542406082xxxxx ... | b'\x89PNG\r\n\x1a\n\x00\xx.. |
| . | . | . | . |
| . | . | . | . |
| . | . | . | . |
| 100 | aerial_ladder_xxxx.png | b'-5.6080293655395xxxx ... | b'\x89PNG\r\n\x1a\n\x00\xx.. |

## 4. Evaluation

After preparing the dataset, we trained our model using the training images from the CIFAR-10 dataset. Since the system requires feature vectors that have been generated by the CNN model as feature vector references in the database, we tested the model using the testing images from CIFAR-10. The files of feature vectors and raw images of CIFAR-10 were loaded in the SQLite database. In the implementation, we ran the system in Raspberry Pi 3 and executed the image search task. We used ten images from CIFAR-10 for each class as the image input. Then, we evaluated the system performance in finding and showing similar images as the result of the image search task on Raspberry Pi 3.

### *4.1. Evaluation Model*

Figure 3 shows the architecture of the CNN model. In this model, we signed the general average pooling before the fully connected layer to replace the two fully connected layers in the previous study. After training the small CNN model, we achieved model accuracy of 68%. The model will generate feature vectors from the testing images for each class of the CIFAR-10 dataset. In the implementation, the model will predict the class of testing images by showing the label of the class, for example, 0 for the airplane. We used the predicted label for reducing the computation time between the feature vectors of the input image with the feature vectors that are stored in the database.

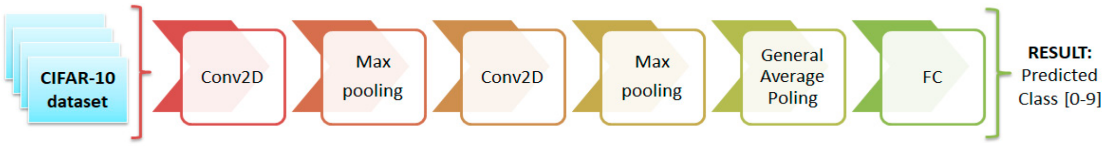


**Figure 3.** CNN model with general average pooling (GAP) layer.

### *4.2. Evaluation Condition*

In the previous study, we discovered that the classical CNN model could determine the tiny images, such as images of the CIFAR-10 dataset by using cosine similarity. In this study, we modified the classical model architecture by removing two fully connected layers and adding one general average layer. The accuracy increased to 68% compared to the previous accuracy of 56%.

Figure 4 shows the main schema for the image search task as an application of CNN in image classification. Given the condition of edge devices such as Raspberry Pi 3, the SQLite database is working well and has an appropriate amount related to its size. The file containing the feature vector and the raw images are stored in the database. By storing these files in the database, it supports the system to run as an off-line system in an edge site.

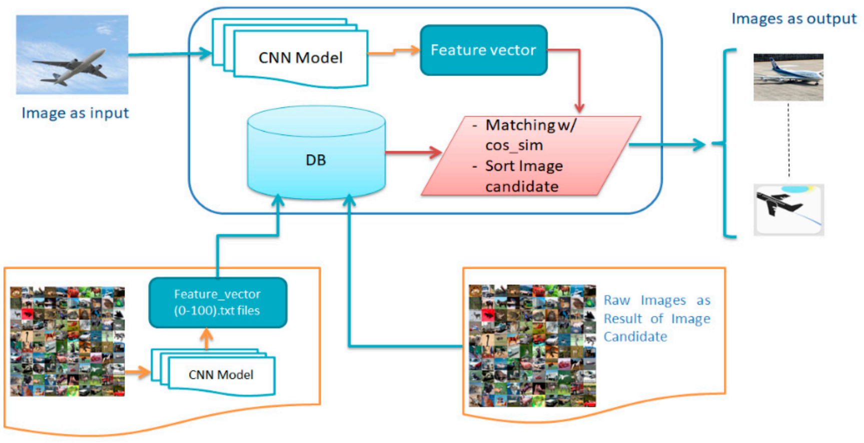


**Figure 4.** Image search with CNN.

Since the tiny images, such as the CIFAR-10 dataset, are a challenge in classification, we used the cosine similarity, as shown in Equation (1), to obtain the most similar images. Also, the convolutional filters in the CNN model can generate the feature vectors. It becomes a solution to use the cosine similarity for image search tasks by calculating the cosine between the reference feature vectors and feature vectors of image input.

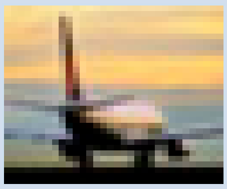
$$cosine\_similarity = \cos\theta = \frac{p.q}{\parallel p \parallel \parallel q \parallel} = \frac{\sum_{i=1}^{n} p_i q_i}{\sqrt{\sum_{i=1}^{n} p_i^2}\sqrt{\sum_{i=1}^{n} q_i^2}} \tag{1}$$

Cosine similarity operates between two dot vectors. Vector $p$ and $q$ represent the feature vectors for generating feature vector from the dataset as a reference and generating feature vectors from the image input in the image search task. The result of the cosine similarity calculation will be on range 0 to 1. The highest value of cosine similarity indicates that two vectors are almost similar.

### 4.3. Results

We investigate the ability of the low-cost system in Raspberry Pi 3 by examining several images as the image input. Images of representation for each class from the CIFAR-10 dataset are selected randomly. Table 4 shows the representative images. After predicting the category of the input image, the system will continuously calculate the cosine similarity between the input image feature vectors with the reference feature vectors from the database.

**Table 4.** Input image and results.

| Input Image | Category | cos_sim | Id of Image | Output Image |
|---|---|---|---|---|
|  | Airplane | 100. | 0 | |
| | | 54.236 | 2 | |
| | | 53.36 | 1 | |
| | | 44.20 | 6 | |
| | | 43.33 | 3 | |

As a result, the system shows the image result base on the value of the cosine similarity calculation. The system generates five images as the results for each image search task. After getting the calculation result of cosine similarity between the feature vector of the image input and feature vectors in the database, then the system will arrange the Id of the images. In displaying the image results in the Raspberry Pi 3 display, the system shows the images from the highest value of the cosine similarity, which means from the most similar image.

### *4.4. Discussion*

Based on the results, there are several points to be considered. First is that the usage of the general average pooling layer is not only avoiding the overfitting but also reducing the number of parameters used in our model, the small CNN. The general average layer replaces the two fully connected layers. It reduces the number of resource usage in our small CNN model without decreasing the model performance in classification and image search tasks. This fact also tackled the memory limitation issue in low-cost hardware implementation.

The second point is by using a general average pooling layer equally reduces the size of the generated feature vectors. In the previous study, the size of the feature vectors was 196. After applying the general average layer to the small CNN model, we obtained 64 feature vectors. It showed a positive impact on our system regarding the memory limitation in the edge device. Since this system works as an off-line system not connected to any public or local access, all of the required data, such as generated feature vectors as reference for image search tasks, are stored in the lite database, SQLite. The less data stored in the database, the smaller the size of the memory usage for database implementation on the Raspberry Pi 3. In this study, the size of the database is 480 Kb.

In addition, using the CIFAR-10 dataset in the image search task is also a method to prove the capability of the small CNN, especially in image classification with tiny images, because of the limited information from available images in the CIFAR-10 dataset. In this study, model accuracy is improving. It reached 68% accuracy compared with the previous study which had 56% accuracy. The small CNN model showed good performance with the CIFAR-10 dataset. In future study, we would like to enlarge the number of files in the database and enhance the small CNN model performance for low-cost hardware implementation.

## 5. Conclusions

As the conclusion for this study, we discovered that a small CNN could be run in Raspberry Pi 3 as an environment to implement the image search task. For reducing the time in the image search task, we use the predicted class. The cosine similarity calculates the distance between the feature vector of the input image and feature vectors in the database. As a result, this system shows five similar images as the result of the input image.

The small CNN model uses Keras, Tensorflow, Sqlite database, sckit-learn library, and some python libraries such as pandas and numpy to build the model architecture. For training and testing, the system works in different environments. In the training of the model, the system runs on the personal computer in the laboratory. The environment of this step is a personal computer with Quadro M5000 and Tesla K40c GPU. After preparing the small CNN model and database, the system runs on Raspberry Pi 3 as an edge device which completed with the mini display.

**Author Contributions:** Conceptualization, M.D., M.A., and M.I.; methodology, M.D. and J.C.; validation, M.A.; formal analysis, M.A. and M.I.; supervision, M.A. and M.I.; writing—original draft, M.D.; writing—review and editing, M.D., J.C., M.A., and M.I. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Acknowledgments:** M.D. gratefully acknowledges Research and Innovation in Science and Technology Project (RISET-Pro) from KEMENRISTEKDIKTI (Ministry of Research, Technology, and Higher Education of the Republic of Indonesia) for scholarship support during this study.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Shi, W.; Cao, J.; Li, Y.; Xu, L. Edge Computing: Vision and Challenges. *IEEE Internet Things J.* **2016**, *3*, 11. [CrossRef]
2. Li, E.; Zhou, Z.; Chen, X. Edge Intelligence: On-Demand Deep Learning Model Co-Inference with Device-Edge Synergy. In Proceedings of the 2018 Workshop on Mobile Edge Communications, Budapest, Hungary, 20 August 2018; pp. 31–36.
3. Murshed, M.G.S.; Murphy, C.; Hou, D.; Khan, N.; Ananthanarayanan, G.; Hussain, F. Machine Learning at the Network Edge: A Survey. *arXiv* **2019**, arXiv:1908.00080.
4. Jiang, Z.; Chen, T.; Li, M. Efficient Deep Learning Inference on Edge Devices. In Proceedings of the ACM Conference on Systems and Machine Learning (SysML'18), Stanford, CA, USA, 15–16 February 2018; p. 3.
5. Kang, J.; Eom, D.-S. Offloading and Transmission Strategies for IoT Edge Devices and Networks. *Sensors* **2019**, *19*, 835. [CrossRef] [PubMed]
6. Zhang, X.; Wang, Y.; Lu, S.; Liu, L.; Xu, L.; Shi, W. OpenEI: An Open Framework for Edge Intelligence. In Proceedings of the 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), Dallas, TX, USA, 7–9 July 2019; pp. 1840–1851.
7. Han, Y.; Wang, X.; Leung, V.C.M.; Niyato, D.; Yan, X.; Chen, X. Convergence of Edge Computing and Deep Learning: A Comprehensive Survey. *arXiv* **2019**, arXiv:1907.08349.
8. Anglano, C.; Canonico, M.; Guazzone, M. Profit-aware Resource Management for Edge Computing Systems. In Proceedings of the 1st International Workshop on Edge Systems, Analytics, and Networking (EdgeSys'18), Munich, Germany, 10 June 2018; pp. 25–30.
9. Chen, J.; Ran, X. Deep Learning With Edge Computing: A Review. *Proc. IEEE* **2019**, *107*, 1655–1674. [CrossRef]
10. Lin, L.; Liao, X.; Jin, H.; Li, P. Computation Offloading towards Edge Computing. *Proc. IEEE* **2019**, *107*, 1584–1607. [CrossRef]
11. Lai, L.; Suda, N. Enabling Deep Learning at the loT Edge. In Proceedings of the 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Marrakech, Morocco, 19–21 March 2018; pp. 1–6.
12. Véstias, M.P. A Survey of Convolutional Neural Networks on Edge with Reconfigurable Computing. *Algorithms* **2019**, *12*, 154. [CrossRef]
13. Zhou, Z.; Chen, X.; Li, E.; Zeng, L.; Luo, K.; Zhang, J. Edge Intelligence: Paving the Last Mile of Artificial Intelligence with Edge Computing. *Proc. IEEE* **2019**, *107*, 1738–1762. [CrossRef]
14. Plastiras, G.; Terzi, M.; Kyrkou, C.; Theocharides, T. Edge Intelligence: Challenges and Opportunities of Near-Sensor Machine Learning Applications. In Proceedings of the 2018 IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP), Milan, Italy, 10–12 July 2018; pp. 1–7.
15. Diana, M.; Chikama, J.; Amagasaki, M.; Iida, M.; Kuga, M. Characteristic Similarity Using Classical CNN Model. In Proceedings of the 2019 34th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC), Jeju, Korea, 23–26 June 2019; pp. 1–2.
16. Recht, B.; Roelofs, R.; Schmidt, L.; Shankar, V. Do CIFAR-10 Classifiers Generalize to CIFAR-10? *arXiv* **2018**, arXiv:1806.00451.
17. Sultana, F.; Sufian, A.; Dutta, P. Advancements in Image Classification using Convolutional Neural Network. In Proceedings of the 2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN), Kolkata, India, 22–23 November 2018; pp. 122–129.
18. Yazici, M.; Basurra, S.; Gaber, M.M. Edge Machine Learning: Enabling Smart Internet of Things Applications. *Big Data Cogn. Comput.* **2018**, *2*, 26. [CrossRef]
19. Gu, J.; Wang, Z.; Kuen, J.; Ma, L.; Shahroudy, A.; Shuai, B.; Liu, T.; Wang, X.; Wang, L.; Wang, G.; et al. Recent Advances in Convolutional Neural Networks. *Pattern Recogn.* **2018**, *77*, 24. [CrossRef]
20. Lin, M.; Chen, Q.; Yan, S. Network in Network. *arXiv* **2013**, arXiv:1312.4400.
21. Zhang, B.; Zhao, Q.; Feng, W.; Lyu, S. AlphaMEX: A smarter global pooling method for convolutional neural networks. *Neurocomputing* **2018**, *321*, 13. [CrossRef]
22. Neto, A.R.; Soares, B.; Barbalho, F.; Santos, L.; Batista, T.; Delicato, F.C.; Pires, P.F. Classifying Smart IoT Devices for Running Machine Learning Algorithms. In Proceedings of the 2018 Anais do XLV Seminário Integrado de Software e Hardware, Natal, Brazil, 22–24 July 2018; p. 12.

23. Gauswami, M.H.; Trivedi, K.R. Implementation of machine learning for gender detection using CNN on raspberry Pi platform. In Proceedings of the 2018 2nd International Conference on Inventive Systems and Control (ICISC), Coimbatore, India, 19–20 January 2018; pp. 608–613.
24. Foley, D.; O'Reilly, R. An Evaluation of Convolutional Neural Network Models for Object Detection in Images on Low-End Devices. In Proceedings of the 26th Irish Conference on Artificial Intelligence and Cognitive Science, Dublin, Ireland, 6–7 December 2018; pp. 350–361.
25. Nikouei, S.Y.; Chen, Y.; Song, S.; Xu, R.; Choi, B.-Y.; Faughnan, T.R. Smart Surveillance as an Edge Network Service: From Harr-Cascade, SVM to a Lightweight CNN. In Proceedings of the 2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC), Philadelphia, PA, USA, 18–20 October 2018; pp. 256–265.
26. Nikouei, S.Y.; Chen, Y.; Song, S.; Xu, R.; Choi, B.-Y.; Faughnan, T.R. Real-Time Human Detection as an Edge Service Enabled by a Lightweight CNN. In Proceedings of the 2018 IEEE International Conference on Edge Computing (EDGE), San Francisco, CA, USA, 2–7 July 2018; pp. 125–129.
27. Keras. Available online: https://keras.io/ (accessed on 19 July 2019).
28. Tensorflow. Available online: https://www.tensorflow.org/ (accessed on 22 August 2019).
29. The CIFAR-10 Dataset. Available online: https://www.cs.toronto.edu/~{}kriz/cifar.html (accessed on 19 July 2019).
30. Park, K.; Kim, D.-H. Accelerating Image Classification using Feature Map Similarity in Convolutional Neural Networks. *Appl. Sci.* **2018**, *9*, 108. [CrossRef]
31. Pena, D.; Forembski, A.; Xu, X.; Moloney, D. Benchmarking of CNNs for Low-Cost, Low-Power Robotics Applications. In Proceedings of the RSS 2017 Workshop: New Frontier for Deep Learning in Robotics, Boston, MI, USA, 15 July 2017; pp. 1–5.
32. Wang, C.; Xi, Y. *Convolutional Neural Network for Image Classification*; Johns Hopkins University: Baltimore, MD, USA, 2015; p. 7.
33. Yu, D.; Wang, H.; Chen, P.; Wei, Z. Mixed Pooling for Convolutional Neural Networks. In Proceedings of the Rough Sets and Knowledge Technology: 9th International Conference (RSKT 2014), Shanghai, China, 24–26 October 2014; pp. 364–375.
34. Dam, T.Q.; Cheon, S.; Won, Y. On the IO Characteristics of the SQLite Transactions. In Proceedings of the 2016 IEEE/ACM International Conference on Mobile Software Engineering and Systems (MOBILESoft), Austin, TX, USA, 16–17 May 2016; pp. 214–224.
35. Bhosale, S.T.; Patil, M.T.; Patil, M.P. SQLite: Light Database System. *Int. J. Comput. Sci. Mob. Comput.* **2015**, *4*, 882–885.
36. SQLite. Available online: https://www.sqlite.org/index.html (accessed on 1 August 2019).
37. Allen, G.; Owens, M. *The Definitive Guide to SQLite: Take Control of This Compact and Powerful Tool to Embed Sophisticated SQL Databases within Your Applications*, 2nd ed.; The Expert's Voice in Open Source; Apress: New York, NY, USA, 2010; ISBN 978-1-4302-3225-4.