*data*

MDPI

*Data Descriptor*

# A Low-Resolution Used Electronic Parts Image Dataset for Sorting Application

**Praneel Chand**

Centre for Engineering and Industrial Design (CEID), Waikato Institute of Technology, Hamilton 3200, New Zealand; praneelchand10@yahoo.co.nz or praneel.chand@wintec.ac.nz

**Abstract:** The accumulation of electronic waste (e-waste) is becoming a problem in society. Old parts and components are conveniently discarded instead of being recycled. Economic and environmental measures should be taken by individuals and organizations to enhance sustainability. This could include desoldering and reusing parts from electronic circuit boards. Hence, the purpose of the dataset presented in this paper is for the classification of used electronic parts in linear voltage regulator power supply circuits. The dataset presented in this paper comprises low-resolution ($30 \times 30$ pixels) grayscale images of major reusable electronic parts from a typical adjustable regulated linear voltage power supply kitset. The three major reusable parts are capacitors, potentiometers, and voltage regulator ICs. These are typically the most relatively expensive components. Data representing the parts are extracted from $960 \times 720$ pixel workspace images containing multiple parts. This permits the dataset to be used with multiple types of classifiers, such as lightweight shallow neural networks (SNNs), support vector machines (SVMs), or convolutional neural networks (CNNs). Classification accuracies of 93.5%, 94.9%, and 98.4% were achieved with SNNs, SVMs, and CNNs, respectively. Successful detection and classification of parts will permit a Niryo Ned robotic arm to pick and place parts in the desired locations. The dataset can be used by other academics and researchers working with the Niryo Ned robot and Matlab to handle electronic parts. It can be expanded to include relatively expensive components from other types of electronic circuit boards.

**Dataset:** https://github.com/praneelchand10/Electronics-Parts-Dataset.



**Keywords:** machine vision; image processing; electronic parts



**Citation:** Chand, P. A Low-Resolution Used Electronic Parts Image Dataset for Sorting Application. *Data* **2023**, *8*, 20. https://doi.org/10.3390/data8010020

Academic Editor: Joaquín Torres-Sospedra

Received: 28 October 2022
Revised: 5 January 2023
Accepted: 6 January 2023
Published: 14 January 2023



## 1. Summary

In circular economies [1], the reduction and elimination of waste is a key priority. This facilitates a robust system that is beneficial for the environment, businesses, and humans. Reusing and recycling devices and components should be promoted in all parts of an economy. In classroom and laboratory settings where access to materials can be limited, physical equipment and hardware parts used for practical activities can be recycled or reused [2].

Electrical engineering courses often rely on hardware components, such as resistors, capacitors, inductors, voltage regulators, and diodes, for project work. As an example, students are required to construct an electrotechnology product in the Electrical and Electronics Applications course at Waikato Institute of Technology [3,4]. The construction could be on a printed circuit board (PCB), Veroboard, or breadboard. After project work, the constructed boards are either thrown away or kept in storage (Figure 1). Functional components are discarded instead of being reused. In the circular economy concept, components on these circuit boards could be removed and collected as part of soldering practice lessons. Part sorting is a mundane task. This process could be achieved using an intelligent

automated sorting system. Hence, the data descriptor presented in this article is being utilized in a project that investigates vision-based detection, classification, and sorting of used electronic parts [5,6]. In particular, the classification of commonly used and relatively expensive electronic project parts, such as capacitors, potentiometers, and voltage regulator ICs, is investigated. These are the major reusable electronic parts from typical adjustable regulated linear voltage power supply kitsets. The project received funding from Waikato Institute of Technology.

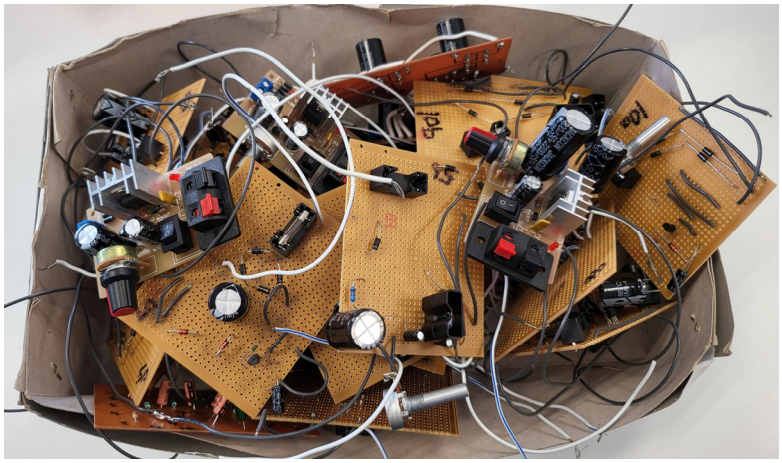


**Figure 1.** Collection of used power supply circuit boards.

A comprehensive literature review of various methods for detecting and classifying electronic parts is available in [6]. Two common applications of vision systems for electronic parts are PCB quality inspection [7–9] and loose electronic parts classification [10–12]. With the exception of [10] that utilized images from Kaggle [13], these methods produced their own customized datasets.

Deep learning [14] is the most commonly employed technique in these applications. Variations of the 'you only look once' (YOLO) deep learning algorithm [15] were applied in [11,12]. The performance of a customized CNN model is compared with AlexNet, GoogleNet, ShuffleNet, and SqueezeNet in [10]. Deep learning methods can perform object detection and classification via a single network model. However, these models typically rely on large datasets with high-resolution images. Consequently, they require powerful computing resources.

Traditional machine learning with SVM and principal component analysis (PCA) was evaluated in [9]. Scale-invariant feature transform (SIFT) [16] parameters extracted from raw images were applied to artificial neural networks (ANNs) and SVMs in [8]. Traditional machine learning approaches typically do not require powerful computing resources, such as deep learning methods. However, they tend to have lower classification accuracy than deep learning models.

There are datasets of electronic parts images publicly available on websites, such as Kaggle [13]. Alternatively, users can create their own datasets by sourcing images from electronic component supplier websites, such as Digi-Key [17] or RS Components [18]. However, in the case of used electronic components, there are no publicly available dataset images based on an internet search using Google. This requires the collection of data directly from the workspace environment where object sorting takes place. In [11], researchers collected electronic parts data using an overhead workspace camera for their custom application. Similarly, images of capacitors and inductors are collected using a high-resolution (3264 $\times$ 2488 pixels) camera in [12]. A very-high-resolution imaging device (8000 $\times$ 9000 pixels) collected images of PCBs from various electronics devices in [9]. The high resolution is needed to extract surface mount component images from the PCB images.

The data descriptor presented in this paper can be useful for academics and researchers developing Matlab-based control systems for the Niryo Ned robotic arm [19]. Other object classification algorithms and additional data can be generated for alternative sorting applications using the Niryo Ned robotic arm and conveyor belt system [20]. Additionally, the data could be used to identify parts for PCB placement or even detect faulty components if the dataset is expanded further. Extracting low-resolution (30 × 30 pixels) images of parts from higher-resolution workspace images (Figure 2) enables the dataset to be used with a variety of classifiers, such as shallow neural networks (SNNs), support vector machines (SVMs), or convolutional neural networks (CNNs). As outlined in [6], this is advantageous because the low-resolution images require less computational power. This permits the classifiers to be implemented using standard laptop computers.

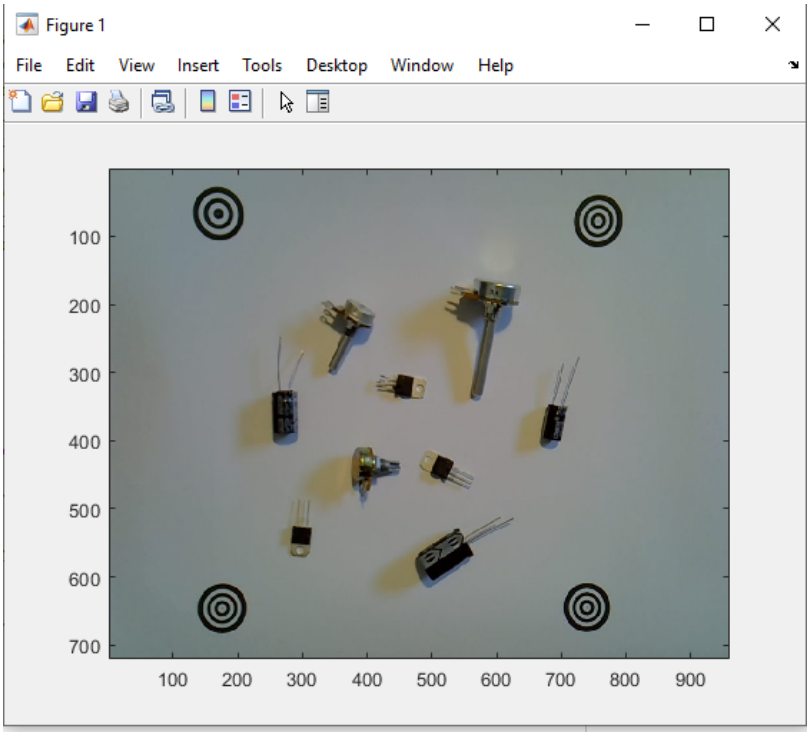
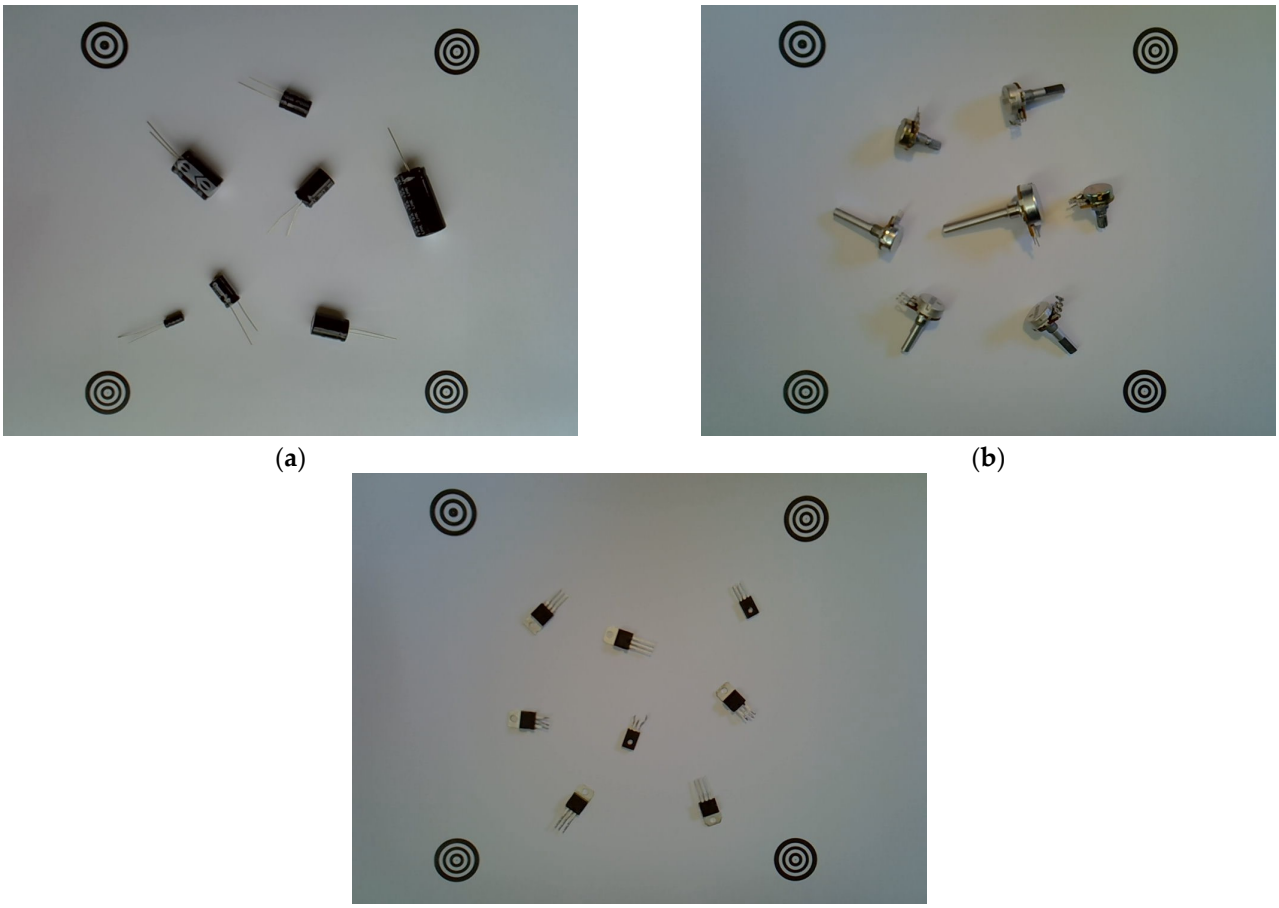



**Figure 2.** A 960 × 720 pixel image taken using Matlab.

## 2. Data Description

The current dataset contains over 1734 grayscale images extracted using an object detection process described in Section 3. There are three classes of objects: capacitor, potentiometer, and regulator. A sample of original 960 × 720 pixel images containing multiple instances of each object class in the workspace environment is shown in Figure 3. The capacitor object class (class 1) consists of electrolytic through-hole-type capacitors. Panel mount rotary potentiometers make up the potentiometer object class (class 2). The regulator object class (class 3) comprises linear voltage regulator integrated circuits (ICs) with a TO-220 casing. Further information about the objects in each class is available in [4]. Each object class has at least 578 images. This provides an even distribution of data across all three classes. A sample of grayscale images from the dataset extracted using the object detection process is shown in Figure 4. The dataset is organized primarily for ease of use with Matlab. It can be restructured for use with other software if required. In the grayscale images, each pixel has a numerical value between 0 and 255. Zero (0) corresponds to black, and 255 corresponds to white.



(**a**) (**b**)

(**c**)

**Figure 3.** Original 960 × 720 pixel images of the workspace. (**a**) Capacitors (class 1); (**b**) Potentiometers (class 2); (**c**) Regulators (class 3).

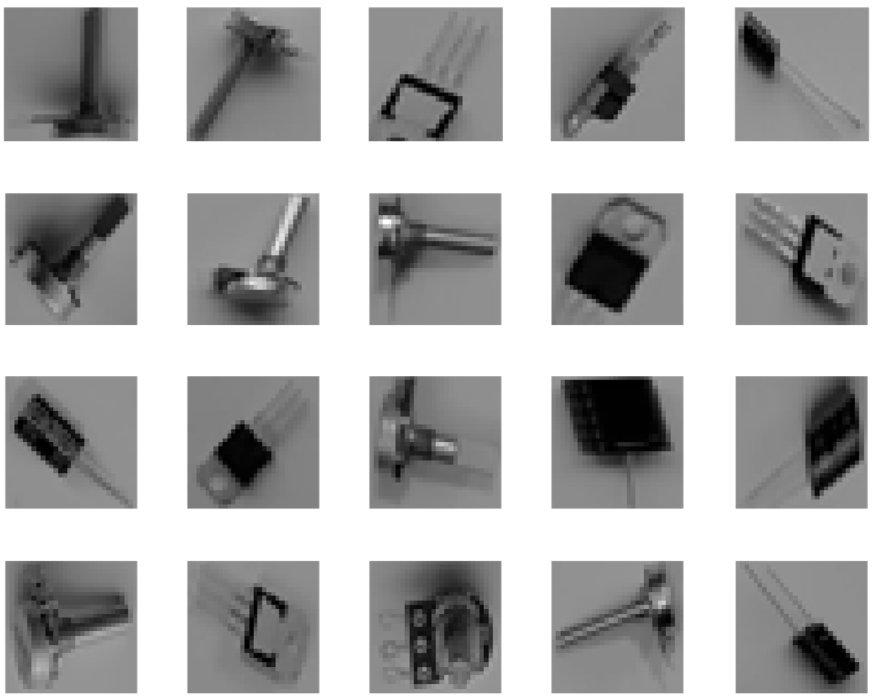


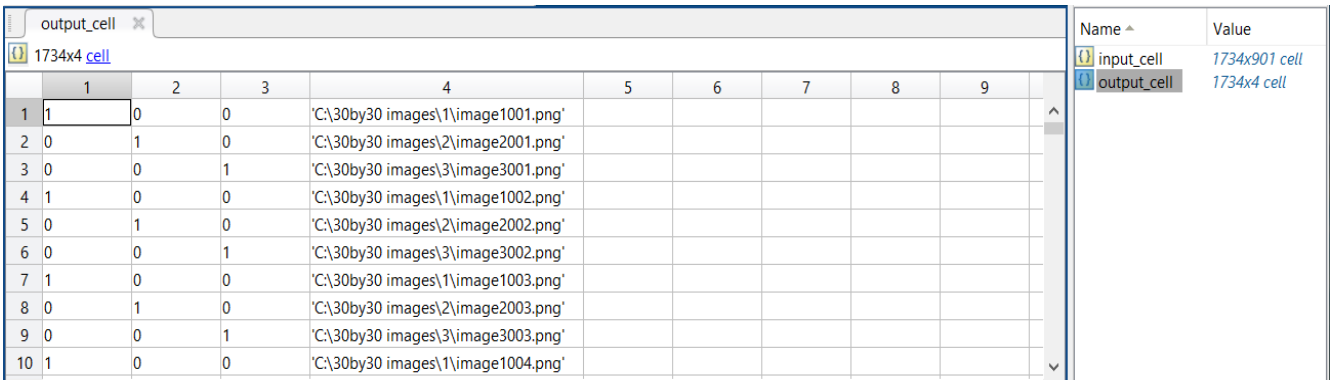
**Figure 4.** Samples of 30 × 30 pixel resolution grayscale object images from the dataset.

### *2.1. SNN and SVM Dataset Formats*

The SNN and SVM datasets are stored in Matlab workspace files as cell arrays. Each workspace file has two arrays: one representing the pixels of the grayscale images (input), and the other representing the object classes (output). The 30 × 30 pixel grayscale images (refer to Section 2.2 for more details, such as folders, filenames, etc.) are converted to 1 × 900 row vectors for Matlab compatibility. All n data samples are collated in an n × 900 array representing the input. The n rows of the array are ordered alphanumerically by filename in each folder (refer to Section 2.2). For the SNN dataset, the data are formatted for use with Matlab's Neural Pattern Recognition tool (nprtool). On the other hand, the SVM data are arranged in a format for use with the Matlab Classification Learner App. Both the SNN and SVM data files use the same input array format (n × 900 array). Each input array has an additional 901st column that contains information about the grayscale image filename. However, the output arrays vary, and the difference is summarized below:

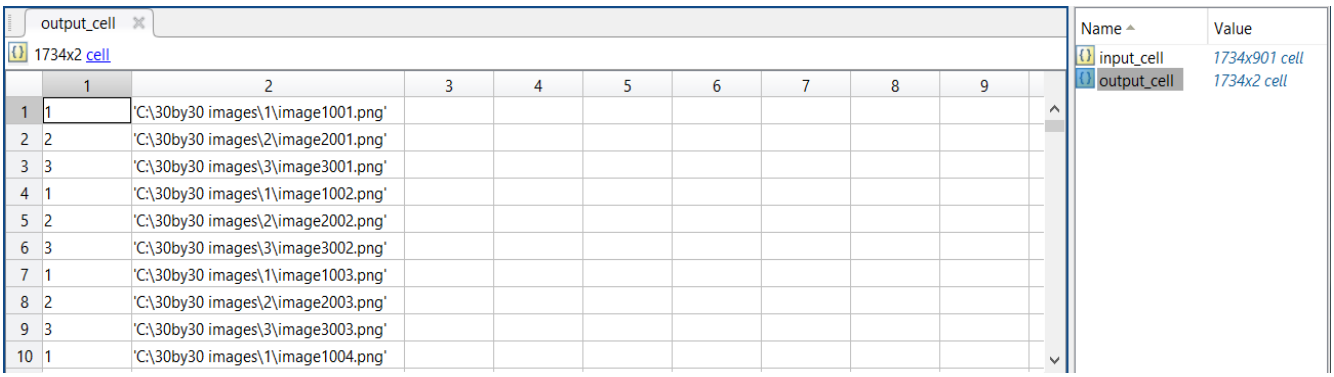
- The SNN data use an output array format of n × 3. The three columns are used for placing 0 s or 1 s representing the output class. An additional 4th column contains information about the grayscale image filename.
- For the SVM data, the output array has a format of n × 1. A single numerical value (1, 2, or 3) representing the object class is entered into the array. An additional 2nd column contains information about the grayscale image filename.

A portion of each output array format is illustrated in Figures 5 and 6. Figure 7 shows a portion of the input array data representing the grayscale images. If a user prefers to use a different software, they can use the portable network graphics (.png) image files described in Section 2.2 and convert them to their preferred software/hardware format requirements. Non-Matlab users can access the comma-separated values (.csv) format files to view the SNN and SVM data.



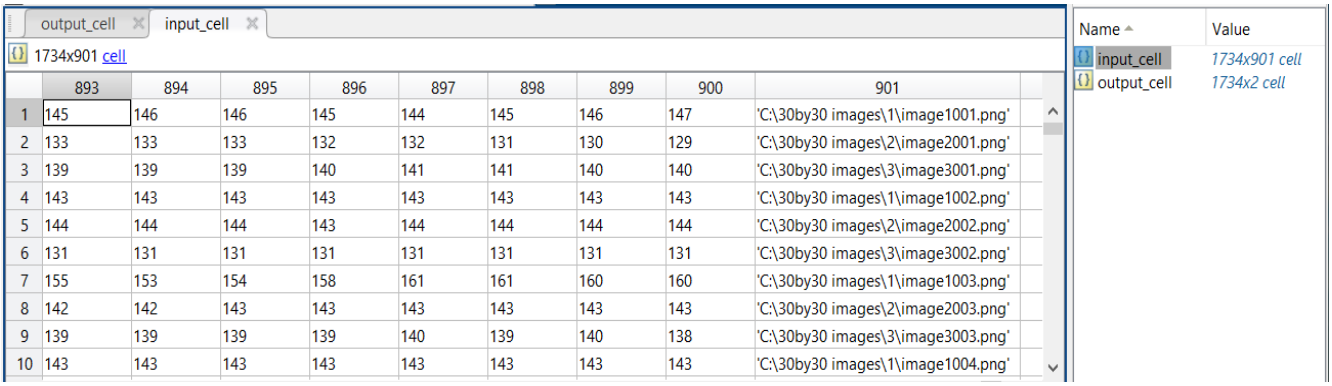

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 'C:\30by30 images\1\image1001.png' |
| 2 | 0 | 1 | 0 | 'C:\30by30 images\2\image2001.png' |
| 3 | 0 | 0 | 1 | 'C:\30by30 images\3\image3001.png' |
| 4 | 1 | 0 | 0 | 'C:\30by30 images\1\image1002.png' |
| 5 | 0 | 1 | 0 | 'C:\30by30 images\2\image2002.png' |
| 6 | 0 | 0 | 1 | 'C:\30by30 images\3\image3002.png' |
| 7 | 1 | 0 | 0 | 'C:\30by30 images\1\image1003.png' |
| 8 | 0 | 1 | 0 | 'C:\30by30 images\2\image2003.png' |
| 9 | 0 | 0 | 1 | 'C:\30by30 images\3\image3003.png' |
| 10 | 1 | 0 | 0 | 'C:\30by30 images\1\image1004.png' |

**Figure 5.** Output array of SNN data format.




| | 1 | 2 |
|---|---|---|
| 1 | 1 | 'C:\30by30 images\1\image1001.png' |
| 2 | 2 | 'C:\30by30 images\2\image2001.png' |
| 3 | 3 | 'C:\30by30 images\3\image3001.png' |
| 4 | 1 | 'C:\30by30 images\1\image1002.png' |
| 5 | 2 | 'C:\30by30 images\2\image2002.png' |
| 6 | 3 | 'C:\30by30 images\3\image3002.png' |
| 7 | 1 | 'C:\30by30 images\1\image1003.png' |
| 8 | 2 | 'C:\30by30 images\2\image2003.png' |
| 9 | 3 | 'C:\30by30 images\3\image3003.png' |
| 10 | 1 | 'C:\30by30 images\1\image1004.png' |

**Figure 6.** Output array of SVM data format.




| | 893 | 894 | 895 | 896 | 897 | 898 | 899 | 900 | 901 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 145 | 146 | 146 | 145 | 144 | 145 | 146 | 147 | 'C:\30by30 images\1\image1001.png' |
| 2 | 133 | 133 | 133 | 132 | 132 | 131 | 130 | 129 | 'C:\30by30 images\2\image2001.png' |
| 3 | 139 | 139 | 139 | 140 | 141 | 141 | 140 | 140 | 'C:\30by30 images\3\image3001.png' |
| 4 | 143 | 143 | 143 | 143 | 143 | 143 | 143 | 143 | 'C:\30by30 images\1\image1002.png' |
| 5 | 144 | 144 | 144 | 143 | 144 | 144 | 144 | 144 | 'C:\30by30 images\2\image2002.png' |
| 6 | 131 | 131 | 131 | 131 | 131 | 131 | 131 | 131 | 'C:\30by30 images\3\image3002.png' |
| 7 | 155 | 153 | 154 | 158 | 161 | 161 | 160 | 160 | 'C:\30by30 images\1\image1003.png' |
| 8 | 142 | 142 | 143 | 143 | 143 | 143 | 143 | 143 | 'C:\30by30 images\2\image2003.png' |
| 9 | 139 | 139 | 139 | 139 | 140 | 139 | 140 | 138 | 'C:\30by30 images\3\image3003.png' |
| 10 | 143 | 143 | 143 | 143 | 143 | 143 | 143 | 143 | 'C:\30by30 images\1\image1004.png' |

**Figure 7.** Input array of SNN and SVM data formats representing the grayscale images.

### *2.2. CNN Dataset Format*

For CNN use, the dataset is stored as portable network graphics (.png) image files in folders representing each object class. The parent folder ("30by30 images") has three subfolders as shown in Figure 8a.

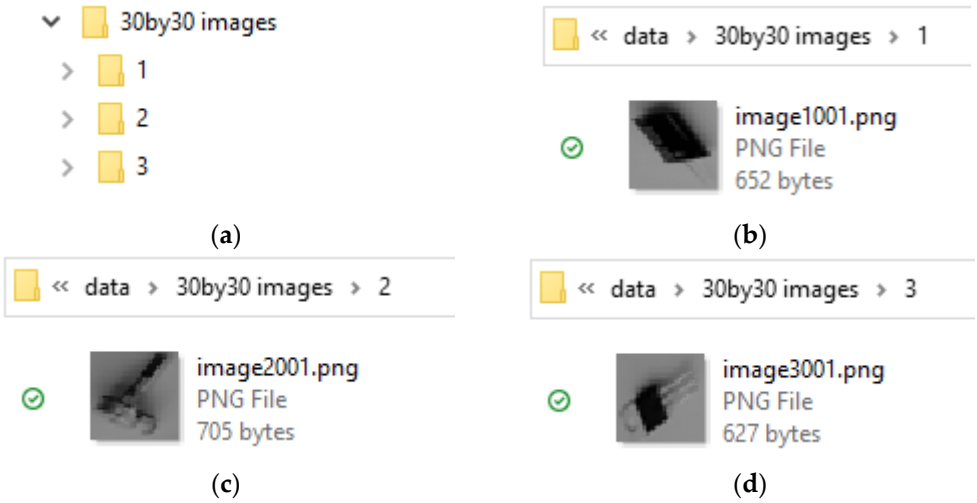



**Figure 8.** Dataset format for CNN. (**a**) Folder structure; (**b**) Object class 1 folder (capacitors); (**c**) Object class 2 folder (potentiometers); (**d**) Object class 3 folder (regulators).

The first subfolder ("1") contains images of the capacitors (object class 1). As shown in Figure 8b, the filenames have the format image1xxx.png where xxx represents the image number. A similar notation is used by the other two subfolders as shown in Figure 8c,d. The subfolder name and the first digit of the filename correspond to the object class, and this can easily be extracted when the image files are batch-loaded using Matlab or any other software.

## 3. Methods

Figure 9 illustrates the setup of the workspace environment used to acquire object data. An overhead camera (Logitech HD C270) is mounted in the center of the workspace to capture images. The camera is positioned at a height of 0.37 m. This height can be adjusted because the four circular boundary markers are used to calibrate pixel distances. The workspace environment assumes a fixed lighting condition of approximately 350 lux, which is the typical level for focused activity [21].

The collected data are used by an object classification system to sort electronic parts [6]. An overview of the system is presented in Figure 10. The captured camera image is passed into the object detection process (Section 3.1) for detecting instances of unclassified objects

and their locations. The unclassified object images (regions of interest, ROI) are resized to conform to the classification process requirements. After resizing, the classification process matches the unclassified object images to an object class it has been trained to recognize. Training the classifier is the primary purpose of the dataset presented in this paper (Section 3.2). After classification, an object can be moved from the workspace to a desired target location.

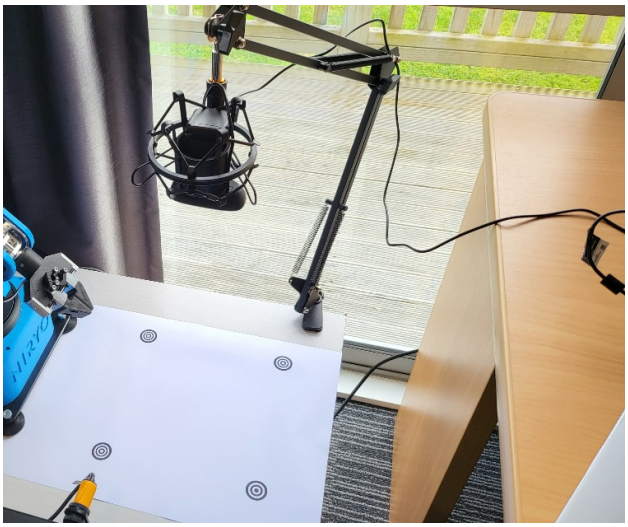


**Figure 9.** Workspace environment setup for collecting object data.

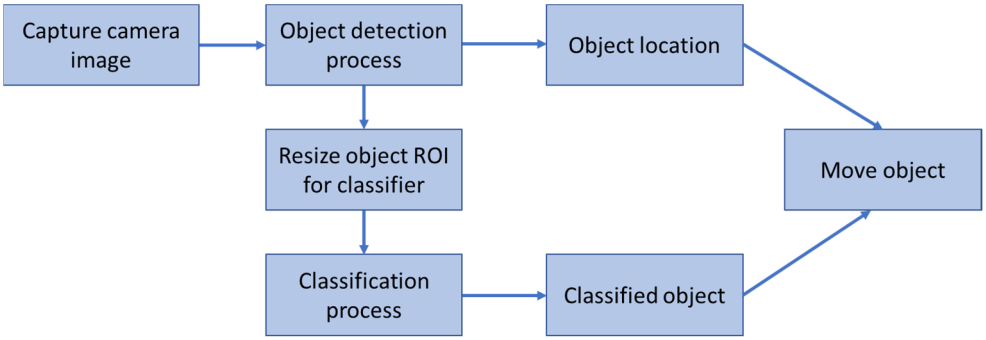



**Figure 10.** Overview of object classification system [6].

Insufficient data in classifier training can lead to overfitting, which makes the model's generalization ability worse. Therefore, the dataset needs to be augmented to improve the diversity of the sample. Conventional methods, such as flipping images and rotating images, were used to augment the dataset. Other complex preprocessing, such as contrast enhancement or blur processing, was not needed.

### *3.1. Object Detection Process*

An overview of the object detection process is provided in Figure 11. A variety of image processing algorithms are applied to the original 960 × 720 pixel workspace color image to extract grayscale object images. The first part of the process is to convert the RGB color image to grayscale using the weighted method (1) [22]. Next, edge detection algorithms can be applied to determine the boundaries (outlines) of objects within images [23]. The Canny edge detection algorithm performed the best in detecting shape outlines out of the available algorithms in Matlab (Sobel, Canny, Prewitt, and Roberts). Canny is less likely to be fooled

by noise and more likely to detect true weak edges due to its use of two thresholds. For this application, the values of the high and low thresholds are 0.1 and 0.04, respectively.

$$\text{gray} = 0.299R + 0.587G + 0.114B \quad (1)$$

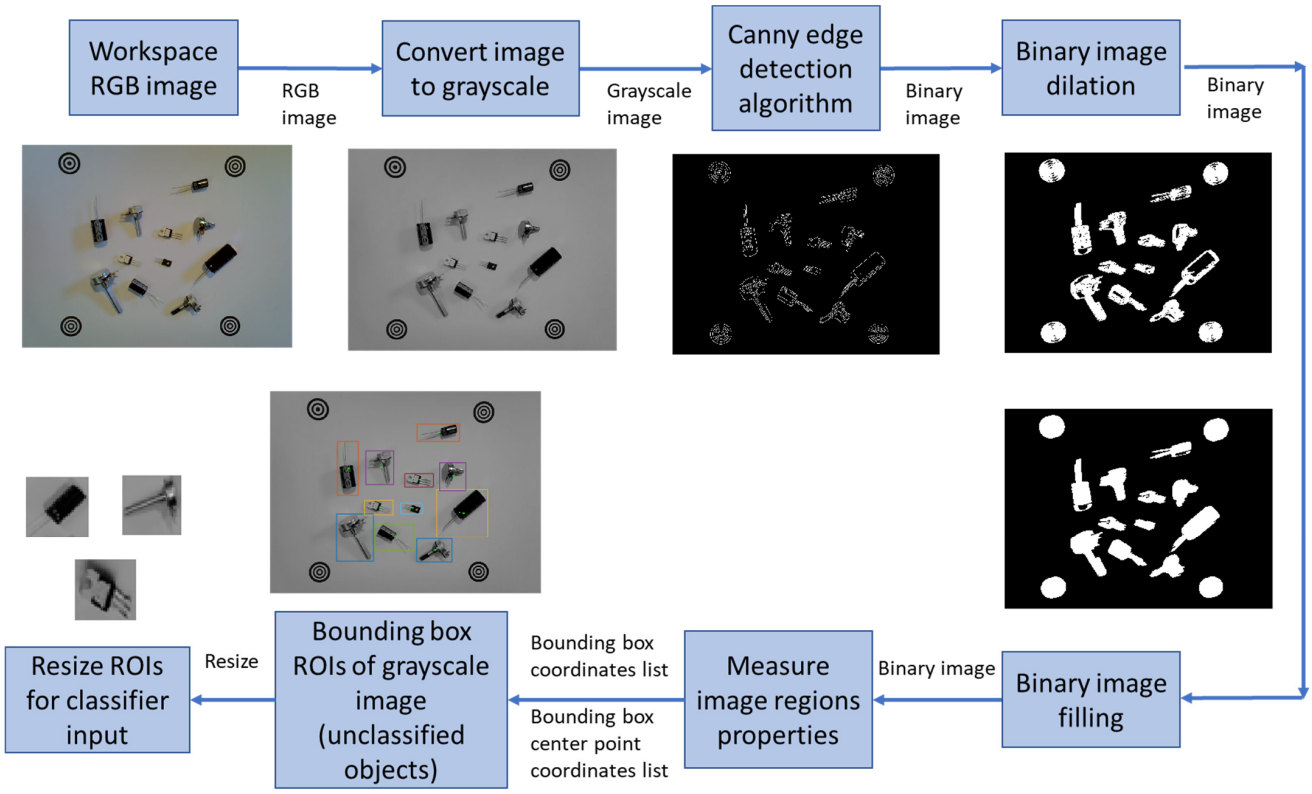



**Figure 11.** Object detection process.

A binary image is output from the Canny edge detection algorithm. This binary image is then dilated to further improve connectivity between the edges. A rectangular structuring element filter that enlarges the edges of the binary image by 10 pixels is applied. Edge connectivity is important as the next stage involves flood-filling the binary image to form filled (solid) shapes representing the detected objects. After flood-filling, the binary image is further processed by measuring the properties of the image regions. The "BoundingBox" property provides the positions and sizes of the smallest boxes containing each detected object. This represents the ROIs or unclassified objects. Green markers are applied to the bounding box centers to identify the location of objects in the workspace.

### *3.2. Validation*

Prior to input to the classification process, the validity of the data is checked in two ways:

1. The size of each unclassified object image is checked against an estimated size threshold representing the dimensions of the smallest component to be detected. This eliminates small images that may have been erroneously detected due to noise or tiny holes in components, such as voltage regulators. The pick and place task assumes that objects are physically separated and do not overlap.
2. Pixel grayscale shades are checked against a threshold value. Because all the objects in the current sorting task produce dark shades of gray or black in their grayscale images, an 80% or above shade of gray (i.e., <51 grayscale pixel value) is arbitrarily set as the threshold. At least 5% of the unclassified object image pixels need to

meet the threshold. Otherwise, the image is flagged for manual inspection or can be automatically ignored.

To validate the dataset, classifiers were implemented using SNN, SVM, and CNN. The dataset was randomly divided into 70% training (1214 images), 15% validation (260 images) and 15% test (260 images). Five-fold cross-validation was used in the training process. A Windows 10 HP ProBook 450 G7 laptop running Matlab 2021a was used to implement the various classifiers. The hardware configuration had an Intel i7-10510U processor and 16 GB RAM. Full details of the classification models, results, and analysis are available in the cosubmitted journal paper [6]. Furthermore, a comparison with other classification methods is also presented in [6]. In this data descriptor paper, a summary of the best SNN, SVM, and CNN models is presented in Table 1 for validation purposes.

**Table 1.** Summary of the best classification models.

| Model | Main Parameters | Dataset Test Accuracy % |
|---|---|---|
| SNN | 900 inputs<br>Single hidden layer with 80 neurons<br>3 outputs<br>Scaled conjugate gradient training method | 93.5 |
| SVM | 900 inputs<br>3-class one vs all approach<br>Cubic kernel function | 94.9 |
| SVM + PCA | 900 inputs reduced to 20 components with PCA<br>3-class one vs all approach<br>Cubic kernel function | 94.6 |
| CNN | 30 × 30 pixel grayscale image input<br>3 convolution layers<br>2 pooling layers<br>1 fully connected layer | 98.4 |

All classifiers achieved over 90% accuracy with the dataset. Classifier accuracy improves as the complexity of the network increases. For the SNN model, a single hidden layer with 80 neurons achieved the best result over the tested range of 40 to 120 neurons. The SVM model was tested with four kernel functions (linear, quadratic, cubic, and medium Gaussian). Cubic kernel function performed the best. Principal component analysis (PCA) was applied to reduce the number of support vectors for the SVM model. The number of components was varied between 10 and 50. Classifier performance was least impaired with 20 components. The classification result was similar to using the full 900 inputs for support vectors. Deep learning via a lightweight CNN model achieved the best classification accuracy. The CNN model uses 30 × 30 pixel grayscale images with three convolution layers, two pooling layers, and one fully connected layer. It does not need to perform object detection because this is achieved via the process described in Section 3.1. This permits the same dataset to be used across multiple types of classifiers.

Other deep learning models that perform object detection and classification with a single network rely on higher-resolution color images and a larger number of network layers [9–12]. Hence, these other classifiers are inherently more complex and require heavier computational power. Their accuracies are in the range of 95.21% to 99.99%.

Figure 12 illustrates sample results from the trained classifier models when presented with new multiobject scenes (cap = capacitor, pot = potentiometer, and reg = regulator). The SVM and CNN models classify all objects successfully. There is one misclassification when using the SNN model. Based on the results presented in Table 1 and Figure 12, the applicability of the dataset for training classification models is justified.

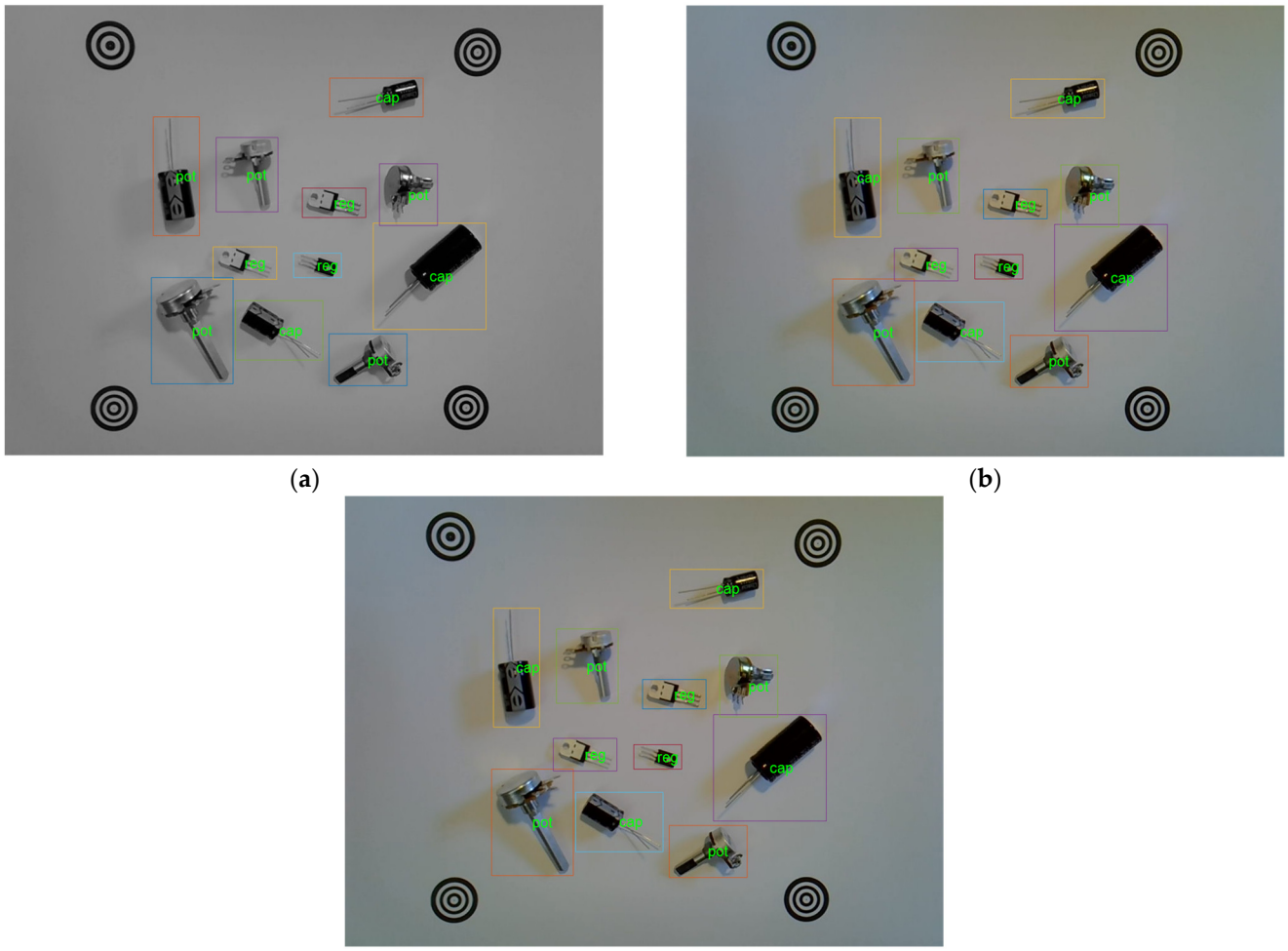



**Figure 12.** Sample results from trained classifier models. (**a**) SNN; (**b**) SVM; (**c**) CNN.

## 4. User Notes

This dataset was derived from a workspace environment based on the Niryo Ned robotic arm. It relies on the use of an overhead camera to take images of the workspace. Computational complexity is reduced by using low-resolution grayscale images. This permits a variety of classifiers, such as SNN, SVM, and CNN, to be implemented using standard laptop computers. Classification accuracies of 93.5%, 94.9%, and 98.4% were achieved with SNNs, SVMs, and CNNs, respectively. These accuracies compare favorably with other similar classes of problems that tend to utilize higher-resolution images. The current implementation assumes that objects are scattered in the workspace such that the robotic arm can relatively easily grasp objects. The Niryo Ned robotic system also comes with a conveyor belt. Because individual objects are detected in the workspace, this dataset can also be applied to classify objects entering the Niryo conveyor belt via overhead camera detection.

**Funding:** This research received partial funding from the Waikato Institute of Technology Contestable Fund.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Data are available at (Github link) under Creative Commons Attribution 4.0 International license.

**Acknowledgments:** The author greatly appreciates the assistance of Niryo and The Brainary for their technical support with the Niryo Ned robotic arm. The author also appreciates Sunil Lal for machine learning model suggestions and proofreading.

**Conflicts of Interest:** The author declares no conflict of interest.

## References

1. Arruda, E.H.; Melatto, R.A.P.B.; Levy, W.; Conti, D.d.M. Circular economy: A brief literature review (2015–2020). *Sustain. Oper. Comput.* **2021**, *2*, 79–86. [CrossRef]
2. Chand, P.; Sepulveda, J. Automating a Festo Manufacturing Machine with an Allen-Bradley PLC. *J. Mechatron. Robot.* **2021**, *5*, 23–32. [CrossRef]
3. Chand, P.; Foulkes, M.; Kumar, A.; Ariyarathna, T. Using Simulated Work-Integrated Learning in Mechatronics Courses. In Proceedings of the 2021 IEEE International Conference on Engineering, Technology & Education (TALE), Wuhan, China, 5–8 December 2021; pp. 01–06.
4. Arduino-Tech. DIY LM317 Adjustable Regulated Voltage Module Suite Kit DC/AC Input. Available online: https://www.arduino-tech.com/diy-lm317-adjustable-regulated-voltage-module-suite-kit-dc-ac-input/ (accessed on 20 October 2022).
5. Chand, P. Investigating Vision Based Sorting of Used Items. In Proceedings of the 2022 IEEE International Conference on Artificial Intelligence in Engineering and Technology (IICAIET), Kota Kinabalu, Malaysia, 13–15 September 2022; pp. 01–05.
6. Chand, P.; Lal, S. Vision-based Detection and Classification of Used Electronics Parts. *Sensors* **2022**, *22*, 9079. [CrossRef] [PubMed]
7. Reza, M.A.; Chen, Z.; Crandall, D.J. Deep Neural Network–Based Detection and Verification of Microelectronic Images. *J. Hardw. Syst. Secur.* **2020**, *4*, 44–54. [CrossRef]
8. Goobar, L. *Machine Learning Based Image Classification of Electronic Components*; KTH: Stockholm, Sweden, 2013.
9. Xu, Y.; Yang, G.; Luo, J.; He, J. An Electronic Component Recognition Algorithm Based on Deep Learning with a Faster SqueezeNet. *Math. Probl. Eng.* **2020**, *2020*, 2940286. [CrossRef]
10. Atik, I. Classification of Electronic Components Based on Convolutional Neural Network Architecture. *Energies* **2022**, *15*, 2347. [CrossRef]
11. Guo, C.; Lv, X.-L.; Zhang, Y.; Zhang, M.-L. Improved YOLOv4-tiny network for real-time electronic component detection. *Sci. Rep.* **2021**, *11*, 22744. [CrossRef] [PubMed]
12. Huang, R.; Gu, J.; Sun, X.; Hou, Y.; Uddin, S. A Rapid Recognition Method for Electronic Components Based on the Improved YOLO-V3 Network. *Electronics* **2019**, *8*, 825. [CrossRef]
13. Acharya, S. Electronics Components and Devices. Available online: https://www.kaggle.com/datasets/aryaminus/electronic-components (accessed on 20 October 2022).
14. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016; p. 800.
15. Jiang, P.; Ergu, D.; Liu, F.; Cai, Y.; Ma, B. A Review of Yolo Algorithm Developments. *Procedia Comput. Sci.* **2022**, *199*, 1066–1073. [CrossRef]
16. Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 91–110. [CrossRef]
17. Digi-Key. Passives. Available online: https://www.digikey.co.nz/en/products/super-category/passives/21 (accessed on 20 October 2022).
18. RS-Components. Passive Components. Available online: https://nz.rs-online.com/web/c/passive-components/ (accessed on 20 October 2022).
19. Chand, P. Developing a Matlab Controller for Niryo Ned Robot. In Proceedings of the 2022 International Conference on Technology Innovation and Its Applications (ICTIIA), Virtual, 23–25 September 2022; pp. 01–05.
20. Niryo. NED User Manual. Available online: https://docs.niryo.com/product/ned/v4.0.0/en/index.html (accessed on 30 June 2022).
21. Level. Appropriate Lighting Levels. Available online: https://www.level.org.nz/energy/lighting-design/appropriate-lighting-levels/ (accessed on 21 October 2022).
22. Qi, L.; Wang, L.; Huo, J.; Shi, Y.; Gao, Y. GreyReID: A Novel Two-stream Deep Framework with RGB-grey Information for Person Re-identification. *ACM Trans. Multimed. Comput. Commun. Appl.* **2021**, *17*, 1551–6857. [CrossRef]
23. Nixon, M.S.; Aguado, A.S. *Feature Extraction and Image Processing for Computer Vision*, 4th ed.; Elsevier Academic Press: London, UK, 2019; p. 650.