sustainability

MDPI

Article

An Empirical Study on Lightweight CNN Models for Efficient Classification of Used Electronic Parts

Praneel Chand [1,*] and Mansour Assaf [2]

[1] Sydney International School of Technology and Commerce, Sydney, NSW 2000, Australia
[2] School of Information Technology, Engineering, Mathematics and Physics, The University of the South Pacific, Suva 1168, Fiji; assaf_m@usp.ac.fj
* Correspondence: praneelchand10@yahoo.co.nz or praneel.c@sistc.edu.au

Abstract: The problem of electronic waste (e-waste) presents a significant challenge in our society as outdated electronic devices are frequently discarded rather than recycled. To tackle this issue, it is important to embrace circular economy principles. One effective approach is to desolder and reuse electronic components, thereby reducing waste buildup. Automated vision-based techniques, often utilizing deep learning models, are commonly employed to identify and locate objects in sorting applications. Artificial intelligence (AI) and deep learning processes often require significant computational resources to perform automated tasks. These computational resources consume energy from the grid. Consequently, a rise in the use of AI can lead to higher demand for energy resources. This research empirically develops a lightweight convolutional neural network (CNN) model by exploring models utilising various grayscale image resolutions and comparing their performance with pre-trained RGB image classifier models. The study evaluates the lightweight CNN classifier's ability to achieve an accuracy comparable to pre-trained red–green–blue (RGB) image classifiers. Experiments demonstrate that lightweight CNN models using 100 × 100 pixels and 224 × 224 pixels grayscale images can achieve accuracies on par with more complex pre-trained RGB classifiers. This permits the use of reduced computational resources for environmental sustainability.

Keywords: convolutional neural network (CNN); deep learning; used electronic components; computer vision; environmental sustainability; green AI; computational efficiency

check for updates

Citation: Chand, P.; Assaf, M. An Empirical Study on Lightweight CNN Models for Efficient Classification of Used Electronic Parts. *Sustainability* **2024**, *16*, 7607. https://doi.org/10.3390/su16177607

Academic Editor: Jesus C. Hernandez

Received: 7 August 2024
Revised: 29 August 2024
Accepted: 30 August 2024
Published: 2 September 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The growing issue of electronic waste (e-waste) poses a challenge in our society as old electronic items are often discarded instead of being recycled. To address this problem and promote sustainability, it is necessary for individuals and organisations to adopt the principles of circular economy [1]. One viable solution is to desolder and reuse parts, thus reducing waste accumulation (Figure 1).

An important part of the recycling process is the correct detection of objects. Automated vision-based techniques are commonly used to identify and locate objects in the workspace. These techniques often rely on artificial intelligence (AI) tools such as machine learning and deep learning [2–4]. Vision sensors can be mounted overhead in the environment or attached to the tool that is used to manipulate objects [5–7].

Depending on the location of the vision sensor, data are collected in the form of images for training a machine-learning model. The properties of the images in the dataset are a contributing factor in determining the complexity of the machine learning model. Images can be red–green–blue (RGB) colour or grayscale and have various resolutions. Low-resolution two-dimensional grayscale images can be converted into single-dimension arrays and input to shallow neural networks (SNNs) or support vector machines (SVMs) for classification [8]. Feature extraction techniques such as the bag of visual words (BoVW) [9], scale-invariant feature transform (SIFT) [10], or principal component analysis (PCA) [11] can

be applied to reduce the dimensionality of the data before input into the classifier. However, traditional machine learning models such as SNNs and SVMs have limited accuracy.

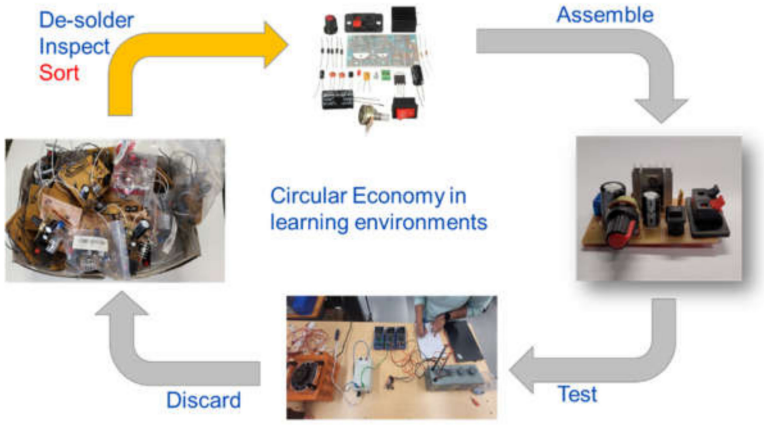


Figure 1. Overview of the circular economy concept in learning environments.

Higher accuracy is typically achieved with deep learning models such as convolutional neural networks (CNNs) [12,13]. Commonly used pre-trained deep learning models employ RGB colour image inputs with resolutions such as 224 × 224 pixels (e.g., ResNet-50, MobileNet-v2, VGG-16), 227 × 227 pixels (e.g., SqueezeNet, AlexNet), or 299 × 299 pixels (e.g., Inception, Xception) [14]. However, deep learning processes often require significant computational resources to perform tasks. These computational resources consume energy from the grid. Consequently, the ever-increasing use of AI in automated systems can place high demands on energy resources. Recently, the areas of green AI [15–17] and energy consumption in deep learning models [18,19] have been discussed. The consensus is that researchers and practitioners need to be accountable for the carbon emissions of AI models they develop. Important considerations such as precision/energy trade-off, algorithm design, and network architecture are reviewed in [15]. According to [16], algorithmic efficiency can mitigate the upcoming energy crisis. Targeted refinement of AI algorithms can produce meaningful reductions. The focus is on energy consumption during inference (post-training execution of AI models) in [18,19]. While object detection and image classification have moderate consumption, an increase in their future applications will increase energy uptake. Therefore, it is important to balance the complexity of deep learning models against the potential performance improvements.

One of the common applications of computer vision is inspecting the quality and integrity of printed circuit boards (PCBs) [10,11,20]. Deep neural network-based image classification techniques have been employed to detect Integrated Circuit (IC) components and verify proper positioning on the completed PCB product in [20]. An upper limit accuracy of 92.31% was achieved with a Siamese network based on the VGG-16 deep learning model. Deep learning has also been used to recognise tiny surface-mount electronic components on PCBs in [11]. The SqueezeNet model was selected due to its reduced number of parameters while preserving accuracy when compared with other models such as AlexNet and VGG-16. Furthermore, an optimised faster SqueezeNet architecture was designed to produce a true positive rate (TPR) of 99.99%.

Deep learning techniques have recently been utilised to detect and classify loose electrical and electronic components [8,13,21–24]. A customised CNN architecture has been developed to classify three categories of parts: resistors, diodes, and capacitors [13]. The model uses six convolution layers and four pooling layers. Performance was benchmarked against four common pre-trained deep learning architectures (AlexNet, GoogleNet, ShuffleNet, and SqueezeNet). An accuracy of 98.99% was achieved using the custom CNN model. This was superior to the accuracy of the pre-trained models (92.95% to 96.67%).

Variations of the 'you only look once' (YOLO [25]) deep learning architecture have also been used to classify electronic parts [21,22]. Fast and accurate real-time object detection is one of the major advantages of YOLO. Bounding boxes and class probabilities are directly predicted from full images in one evaluation. The YOLO-v3 and MobileNet architectures have been combined to classify four electronic components (three types of capacitors and an inductor) [22]. The basic network architecture is based on YOLO-v3, where the Darknet53 backbone is replaced with the MobileNet architecture. Experiments yielded a mean average precision (mAP) of 0.9521 with a dataset containing 1000 images.

Furthermore, the YOLO-v4 tiny architecture has also been used for real-time electronic component detection [21]. On its own, YOLO-v4 was able to classify twenty types of electronic components with an accuracy of 93.74%. In conjunction with a multiscale attention module (MAM) for filtering out redundant and erroneous features, the YOLO-v4 tiny network accuracy improved to 98.6%. A large dataset comprising 12,000 608 × 608 pixels RGB images was used for the classification task.

Faster R-CNN is also a potential deep learning model for detecting and classifying electronic parts [26]. In R-CNN, a region proposal network shares full-image convolutional features with a detection network. The fully convolutional network simultaneously predicts object bounds and objectness scores at each position. However, it has been shown that Faster R-CNN performs poorly in classifying small, relatively featureless objects [20]. Electronic components such as integrated circuits (ICs) achieved a precision and recall close to zero.

Many electronic component classifiers depend on large datasets for training models. This can be a tedious process. Hence, an approach based on a Siamese network utilising a reduced-size dataset has recently been developed [23]. An improved VGG-16 model was proposed as the feature extraction part of the Siamese network for small sample conditions. A novel correlation loss function improved the model's generalisation performance. A nearest neighbour algorithm was used to complete the classification work. The dataset comprised 17 different types of electronic components, with 182 images for each type. A 94% accuracy with 15 training samples was achieved in experimental results.

The application of CNNs for classifying electronic components in industrial settings has been discussed in [24]. Electronic components were classified on an assembly line using a fixed-mounted camera. An image dataset comprising 3994 images of 11 classes was created. This translates to approximately 363 images for each class. Both custom and pre-trained networks were developed and tested. The custom baseline model used 152 × 202 pixels RGB image inputs with two convolutional layers and two dense fully connected (FC) layers. Various settings were applied to the convolutional layer filters and FC neurons. The custom network achieved up to 96.59% accuracy. The best pre-trained network accuracies varied between 97.81% and 99.03%.

Previously, a robotic system has been developed to sort three types of electronic components (capacitors, potentiometers, and IC voltage regulators) [5,27]. A lightweight custom CNN model was used to classify components. Low-resolution 30 × 30 pixel grayscale images were input to the model. The model had three convolutional layers and one FC layer. An accuracy of 98.1% was achieved by the custom model.

This paper extends the work presented in [5] by investigating more lightweight models utilising various image resolutions and comparing performance with pre-trained RGB image classifier models. The purpose of the study is to empirically develop a lightweight CNN classifier that can achieve an accuracy close to pre-trained RGB classifiers for sustainable AI.

Many of the existing methods and systems for sorting electronic components do not consider factors to mitigate the complexity of their AI algorithms. There is an implicit assumption that processor power can always be increased to account for increased complexity. However, the proliferation of AI in automated systems can create an AI energy crisis [28].

2. Materials and Methods

2.1. Conceptual Framework

An overview of the proposed circular economy concept for learning environments is illustrated in Figure 1. It is possible to reuse electronic components from small-scale printed circuit boards (PCBs) made by students as part of their practical assessments. The current focus of the research is on sorting loose electronic components, which have already been removed from the PCB. Figure 2 shows the setup of a Niryo Ned robotic arm [29,30] and the corresponding workspace for sorting the components. A webcam is mounted above the centre of the workspace to capture images.



Figure 2. Robotic arm and workspace set up.

A broad overview of the sorting process is presented in Figure 3. The process begins with an image of the workspace being captured via the webcam. Next, the captured image is processed to detect the presence and position of objects within the workspace. The object images are then resized for CNN model input compliance. The CNN model then determines the object class, and the object is transferred from the workspace to a bin.

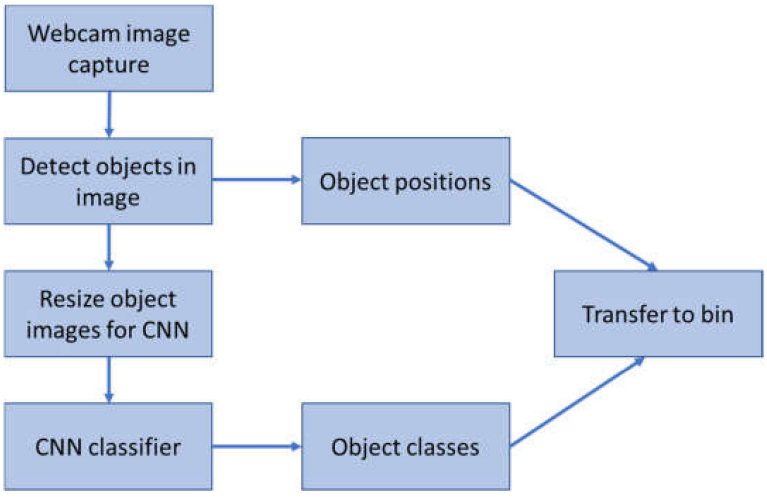


Figure 3. Broad overview of the sorting process.

2.2. Object Detection

Detecting the presence of objects within an image of the workspace is an important part of the pre-processing phase. This is used to collect images of individual electronic components for the dataset. It is also used to locate an object within the workspace during

the execution of the sorting task. A summary of the main stages of the object detection process [5] is illustrated in Figure 4.

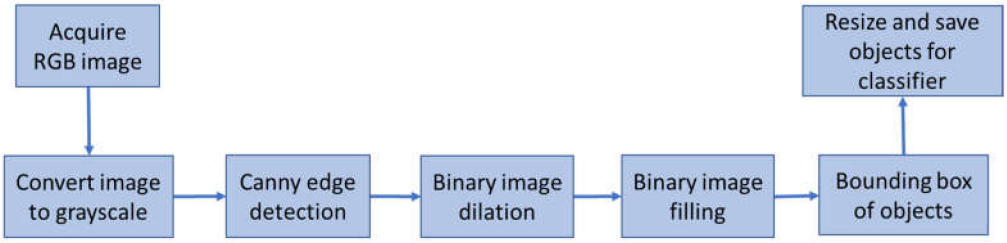


Figure 4. Main stages of the object detection process.

First, an RGB image of the entire workspace is captured at a resolution of 960 × 720 pixels (4:3 aspect ratio high-definition image). Following this, the captured image is converted to grayscale format. Next, the Canny edge detection algorithm is applied to the grayscale image to obtain a binary image which contains the outlines (edges) of any objects. The outlines of objects are dilated to improve connectivity. Following this, a flood fill algorithm is applied to the dilated binary image to produce solid shapes. Bounding boxes of solid objects are determined and pixels within the bounding box are extracted (from the grayscale and/or RGB image(s)). Lastly, the extracted portions of the images are resized based on the CNN classifier input requirements (Tables 1 and 2).

Table 1. Custom CNN model descriptions.

Model Type	Name	Input Size	Convolution Filter Size F (Stride = 1)	Pooling Filter and Stride Size P
1	Model 10-F-P	10 × 10 grayscale	[2 4 6]	[1 2 3]
2	Model 20-F-P	20 × 20 grayscale	[2 4 6]	[1 2 3]
3	Model 30-F-P	30 × 30 grayscale	[2 4 6]	[1 2 3]
4	Model 100-F-P	100 × 100 grayscale	[2 4 6]	[2 3 4 5]
5	Model 224-F-P	224 × 224 grayscale	[2 4 6]	[6 7 8 9]

Table 2. Overview of the pre-trained CNN models [14].

Model	Input Size	Description
ResNet-50	224 × 224 RGB	177 layers deep, 96 MB size (Matlab). Rich feature representations for a wide range of images.
MobileNet-v2	224 × 224 RGB	155 layers deep, 13 MB size (Matlab). Rich feature representations for a wide range of images. Designed for low memory usage.
VGG-16	224 × 224 RGB	41 layers deep, 515 MB size (Matlab). Larger model with a large number of weight parameters.
GoogleNet	224 × 224 RGB	144 layers deep, 27 MB size (Matlab). Residual connections make it possible to train deeper networks.
EfficientNet-b0	224 × 224 RGB	290 layers deep, 20 MB size (Matlab). Layers based on the compound scaling method that uniformly scales the depth, width, and resolution of the network.

2.3. Deep Learning Classifier

Figure 5 illustrates the block diagram of the deep learning CNN approach used in this research. This process is similar to other CNN models discussed in Section 1. The electronic component dataset consists of three classes: capacitors, potentiometers, and IC voltage regulators. Images in the dataset are resized appropriately for input to the various CNN models. The pre-trained CNN models used in this research include ResNet-50, MobileNet-v2, VGG-16, GoogleNet, and EfficientNet-b0. Additionally, a custom CNN model derived

from [5] is utilised. The dataset is randomly divided into 70% training, 15% validation, and 15% test data. Consequently, the trained model can classify the test data.

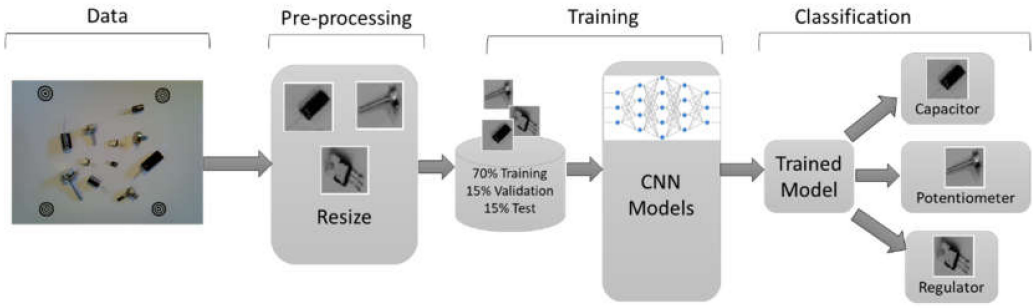


Figure 5. Block diagram of the deep learning CNN method.

2.4. Convolutional Neural Networks (CNNs)

As outlined in Section 1, CNNs are a widely used deep learning architecture for classification tasks due to their high performance (accuracy). The fundamental structure of a CNN comprises input, convolution, activation, pooling, fully connected, and output layers. A variety of models have been developed using various combinations of these layers. A simple series combination of the basic layers is illustrated in Figure 6. Many real-world CNN models (e.g., Table 2) can be much deeper and more complex, utilising components such as skip connections, batch normalisation, and residual blocks.

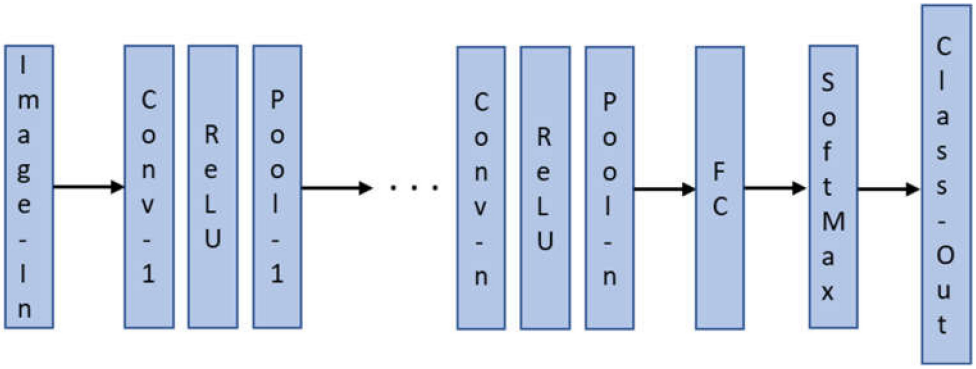


Figure 6. Fundamental structure of a CNN for image classification.

The CNN starts with an input layer that receives the raw image data. There is an input neuron for each pixel in the image. Convolutional layers are the core building blocks of the CNN. These layers extract local features from the input image by applying convolution operations. The convolution process involves multiple filters (or kernels) sliding across the image and computing dot products with local patches. For example, consider a grayscale image represented as an m × m matrix with x_{ij} the entry of the matrix for $i = 1, \ldots, m$ and $j = 1, \ldots, m$. Consider now a convolution filter H, which is taken as an $n \times n$ matrix (n is typically a small integer, such as 2,4,6, etc., that represents the filter size). The convolution of the image x with the filter H is another image matrix y given by Equation (1). For i and j that are out of range, x_{ij} is treated as zero.

$$y_{ij} = \sum_{a=1}^{n} \sum_{b=1}^{n} H_{ab} x_{(i+a)(j+b)} \quad (1)$$

In addition to filter size, two common parameters, padding and stride, also control the specific dimensions of the output image and the sliding movement of the filter over the input image. The output image is a feature map that highlights specific patterns (edges, textures, etc.) in the input image.

A nonlinear activation function, such as a rectified linear unit (ReLU) (Equation (2), is applied to remove negative values from the convolution process. After passing through the activation function, the feature map is downsampled via pooling layers. Common pooling methods include max pooling and average pooling, where a pooling filter size and stride are specified. The maximum value in each local region is selected in max pooling, while average pooling takes the average of the local region. For a feature map having dimensions $q_h \times q_w \times q_c$, the dimensions of the output obtained after a pooling layer are given by Equation (3).

$$ya_{ij} = f\left(y_{ij}\right) = \max\left(0, y_{ij}\right) \quad (2)$$

$$r_h \times r_w \times r_c = \left(q_h - p + 1\right)/s \times \left(q_w - p + 1\right)/s \times q_c \quad (3)$$

where q_h is the height of the feature map, q_w is the width of the feature map, q_c is the number of channels in the feature map, p is the pooling filter size, s is the stride of the pooling layer, r_h is the height of the downsampled feature map, r_w is the width of the downsampled feature map, and r_c is the number of channels in the downsampled feature map.

A fully connected (FC) layer is utilised for classification after the sequence of convolution layers, activation functions, and pooling layers. The FC layer connects every neuron to every neuron in the previous layer. Finally, the output layer consists of SoftMax activation units, providing class probabilities.

2.5. Performance Metrics

The success of model classifications is dependent on the number of samples identified as true positive and true negative. False positives and false negatives lead to incorrect classifications. Performance can be summarised in the form of a confusion matrix (Figure 7) [31]. Accuracy measures how often the classifier correctly predicts the class labels (Equation (4)). Precision quantifies the proportion of true positive predictions among all positive predictions (Equation (5)). Recall (sensitivity) calculates the proportion of true positive predictions among all actual positive instances (Equation (6)). The F-Score is the harmonic mean of precision and recall (Equation (7)).

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad (4)$$

$$Precision = \frac{TP}{TP + FP} \quad (5)$$

$$Recall = \frac{TP}{TP + FN} \quad (6)$$

$$F - Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (7)$$

		Target Class	
		Positive	Negative
Output Class	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

Figure 7. Performance summarised in a confusion matrix.

Another performance measurement parameter is the receiver operating characteristic (ROC) curve. This curve assesses the effectiveness of machine learning classification algorithms by plotting the true positive rate (recall) against the false positive rate at various classification thresholds. Figure 8 illustrates an example of ROC curves.

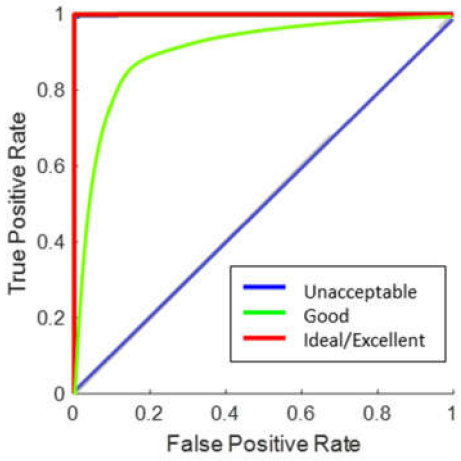


Figure 8. ROC curve examples.

2.6. Baseline Custom CNN Model

The baseline custom CNN model is a feedforward neural network utilising a two-dimensional grayscale image input. Inspired by [32], it has three convolution layers, two pooling layers, one fully connected layer, one softmax, and an output classification layer (Figure 9). This configuration is highly lightweight in comparison to the pre-trained models listed in Table 2. The basic structure remains the same regardless of the input image size. The number of filters in the convolution layers Conv-1, Conv-2, and Conv-3 is set to 10, 20, and 40, respectively. In previous research [5], there was little variation in accuracy around this setting.

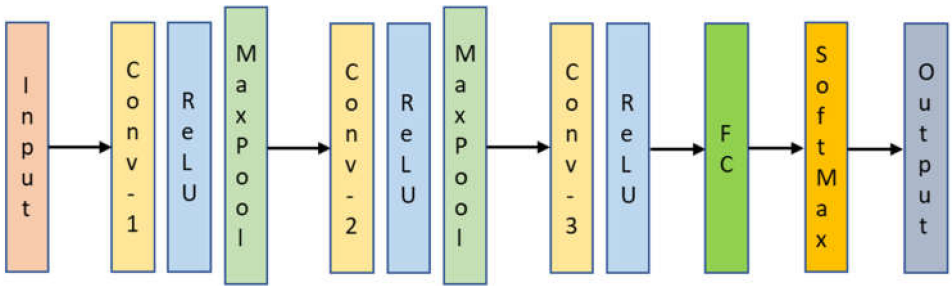


Figure 9. Layers of the baseline custom CNN model.

Table 1 details the different custom CNN models tested in this research. The five model types are categorised based on input image size ranging from 10 × 10 pixels to 224 × 224 pixels. In each model type, the convolution filter size in all convolution layers is varied simultaneously (2, 4, or 6). The stride in all convolution layers is set to 1. Additionally, the pooling layer filter and stride sizes are kept the same in all pooling layers and adjusted based on the values given in Table 1. For example, model type 4 has a 100 × 100 pixels input. Within model type 4, Model 100-4-3 has a convolution filter size of 4 and a pooling filter and stride size of 3.

2.7. Pre-Trained Models

Several pre-trained CNN models are available for image classification as outlined in Section 1. These models have been trained using millions of images and have several output classes. Transfer learning [33] can be applied to adapt these pre-trained models to other image classification tasks such as electronic parts identification. By leveraging the learned features from the pre-trained model, the updated model can learn more quickly and effectively on the new classification task. Transfer learning helps prevent overfitting and allows the reuse of general features across related tasks. Since this research is investigating

the merits of low-resolution images, a selection of the 224 × 224 pixel-size input image models has been selected (Table 2). During the transfer learning process, the last layers of the CNN model (typically the classification layers) are replaced with new layers for the new task. Most of the other layers are typically 'frozen' (i.e., kept the same) since the problem domain is similar.

3. Results

3.1. Datasets and Configuration

In this research, the classification of electronic components is performed using deep learning models with a variety of input image resolutions. The image database contains an even distribution of images, as outlined in Table 3. There are a total of 1734 images in each image set. The images in each category have been extracted from the same original images of the workspace (Figure 2). A visual comparison of the level of detail in each image set is provided in Figure 10. The lower-resolution images (30 × 30 grayscale and lower) lack detail, and this may impact classification accuracy.

Table 3. Image database details (refer to Section 2.3 'Deep Learner Classifier' for class details).

Image Set Name	Description	Class 1 Quantity	Class 2 Quantity	Class 3 Quantity
Gray1	10 × 10 grayscale	578	578	578
Gray2	20 × 20 grayscale	578	578	578
Gray3	30 × 30 grayscale	578	578	578
Gray4	100 × 100 grayscale	578	578	578
Gray5	224 × 224 grayscale	578	578	578
Colour1	224 × 224 RGB	578	578	578

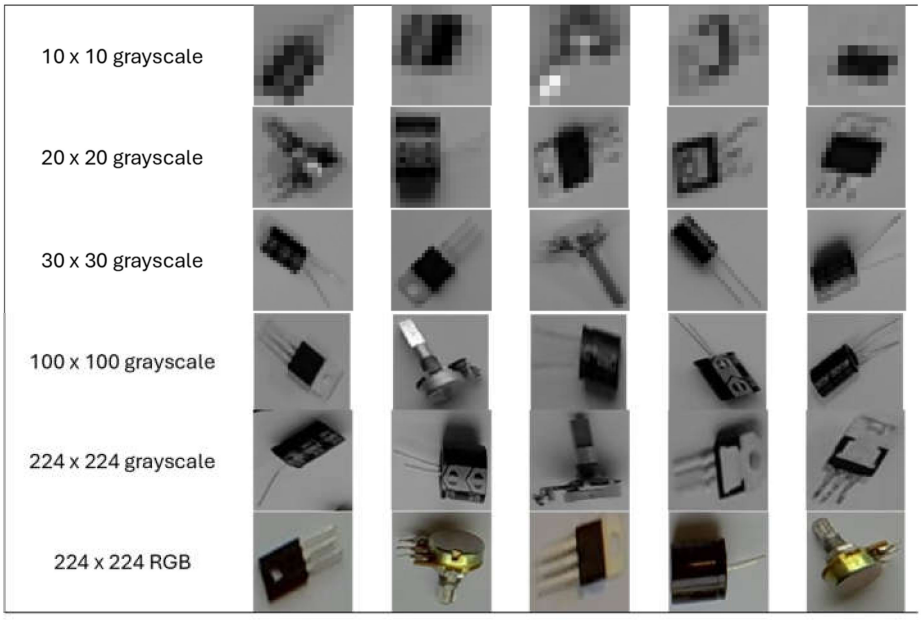


Figure 10. Visual comparison of sample images.

Each image set was randomly divided into 70% training (1214 images), 15% validation (260 images), and 15% test (260 images). A Windows 11 Dell Inspiron 7630 laptop (Syndey, Australia) running Matlab 2021a was used to implement the various classifiers. The hardware configuration included an Intel i7-1360P processor and 16 GB RAM. The CNN model training parameters are outlined in Table 4. These values have been selected based on similar image classification tasks implemented using Matlab. The validation frequency has been determined using Equation (8) [34].

Table 4. CNN model training parameters.

Parameter	Value
Mini-batch size	10
Optimize method	stochastic gradient descent with momentum (sgdm)
Initial learning rate	0.01
Maximum epochs	7
Validation frequency	121 (Equation (8))

$$\text{Validation frequency} = \text{floor}\left(\frac{Training\ Images}{Mini-batch\ size}\right) \quad (8)$$

3.2. Custom CNN Models

Figure 11 shows the test accuracy of the various custom CNN models listed in Table 1. Convolution filter size 2 can produce the highest classification accuracy in all model types. For the lower-resolution image model types (Model 1, Model 2, and Model 3), pooling filter and stride size 2 can produce the highest classification accuracy. In the higher-resolution image model types (Model 4 and Model 5), the pooling filter and stride size need to be increased to improve accuracy. The results indicate that there is an 'empirically optimal' pooling filter and stride size for convolution filter size 2 in all model types. The model with the highest accuracy within each model type is listed in Table 5.

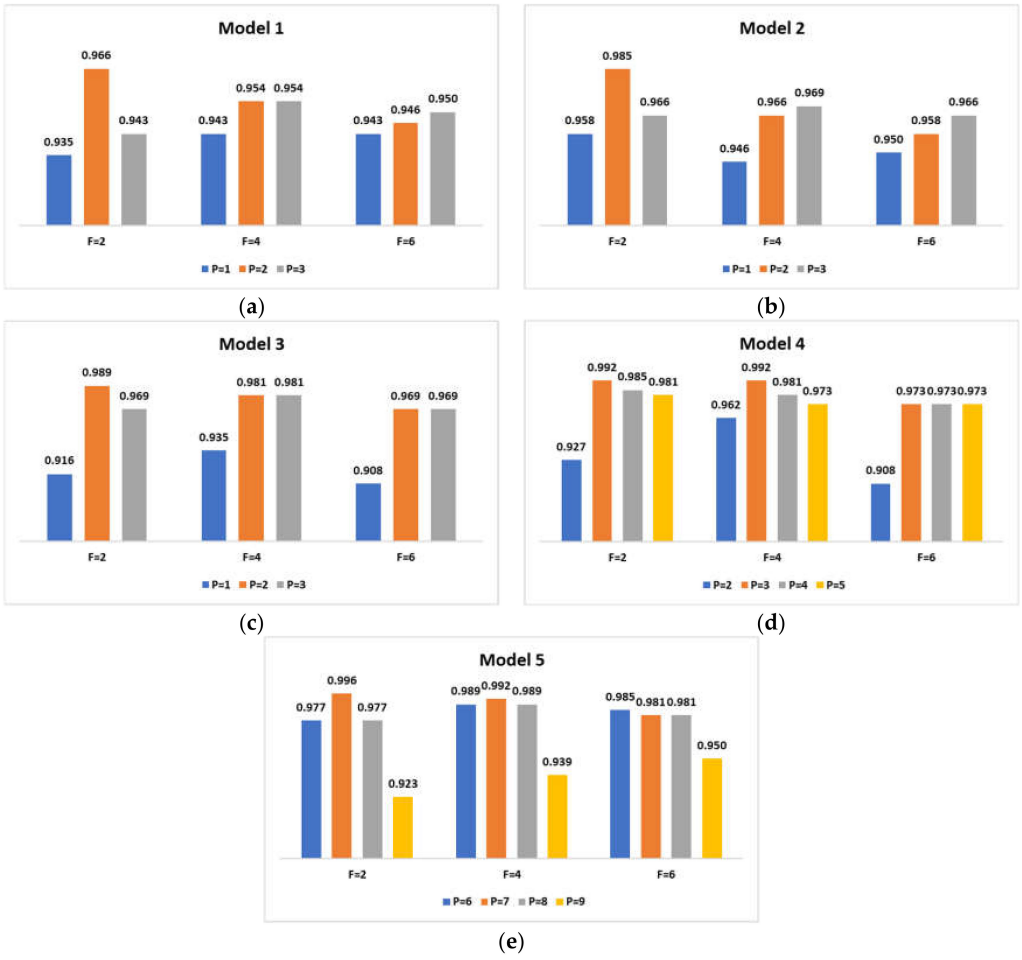


Figure 11. Test accuracy of the various custom CNN models listed in Table 1. (**a**) Model 1; (**b**) Model 2; (**c**) Model 3; (**d**) Model 4; (**e**) Model 5.

Table 5. Summary of the best model within each model type.

Model Type	Name	Accuracy
Model 1	Model 10-2-2	0.9655
Model 2	Model 20-2-2	0.9847
Model 3	Model 30-2-2	0.9885
Model 4	Model 100-2-3	0.9923
Model 5	Model 224-2-7	0.9962

Further analysis of the models in Table 5 is provided in Figures 12 and 13 and Table 6. Figure 12 illustrates the confusion matrices for each of the models. The confusion matrices illustrate the precision (at the right end of each row) and recall (at the bottom end of each column) for each class, along with the accuracy (bottom right diagonal entry). Figure 13 shows the corresponding ROC curves and the improvement as image resolution increases. Table 6 lists the average precision, recall, and F-score for each model.

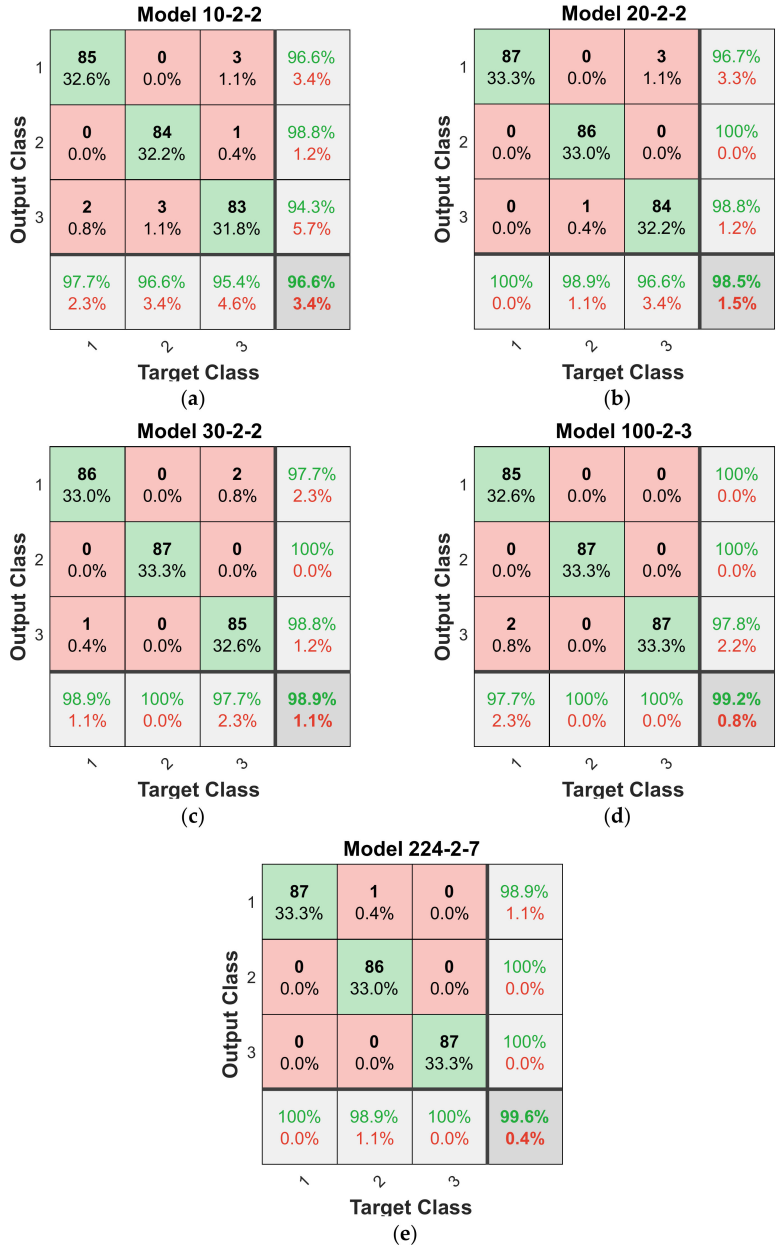


Figure 12. Confusion matrices of the best custom CNN models listed in Table 5. (**a**) Model 10-2-2; (**b**) Model 20-2-2; (**c**) Model 30-2-2; (**d**) Model 100-2-3; (**e**) Model 224-2-7.

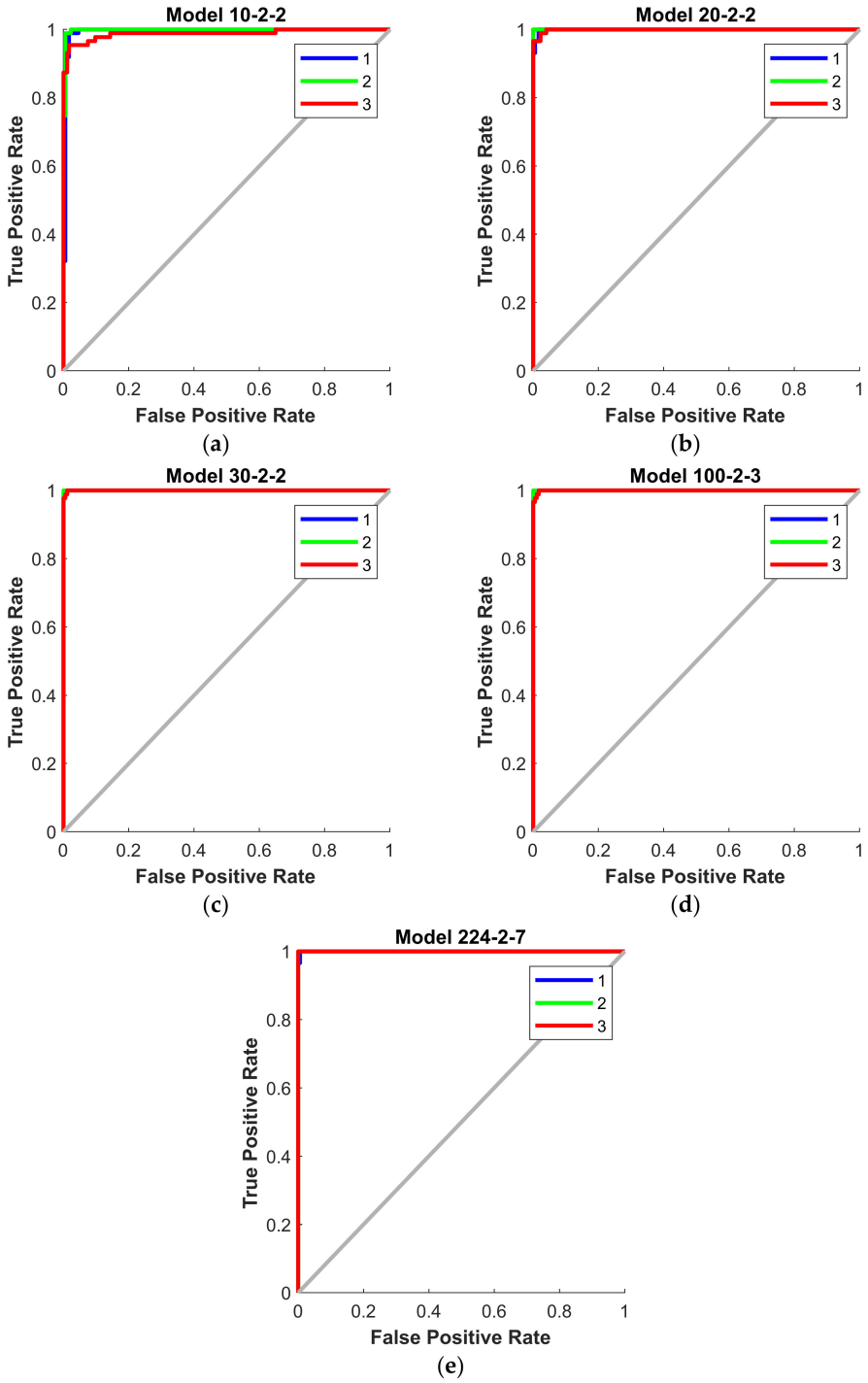


Figure 13. ROC curves of the best custom CNN models listed in Table 5. (**a**) Model 10-2-2; (**b**) Model 20-2-2; (**c**) Model 30-2-2; (**d**) Model 100-2-3; (**e**) Model 224-2-7.

Table 6. Additional performance metrics of the best model within each model type.

Model Name	Precision	Recall	F-Score
Model 10-2-2	0.9657	0.9655	0.9655
Model 20-2-2	0.9849	0.9847	0.9847
Model 30-2-2	0.9885	0.9885	0.9885
Model 100-2-3	0.9923	0.9923	0.9923
Model 224-2-7	0.9962	0.9962	0.9962

3.3. Pre-Trained CNN Models

Figure 14 illustrates the confusion matrices for each of the pre-trained CNN models listed in Table 2. The corresponding ROC curves are similar to Figure 13c,d. All pre-trained models produced unity precision and recall values for class 2. The small number of misclassifications were related to classes 1 and 3. Table 7 lists the accuracy, precision, recall, and F-score for each model. The values are the averages across all classes.

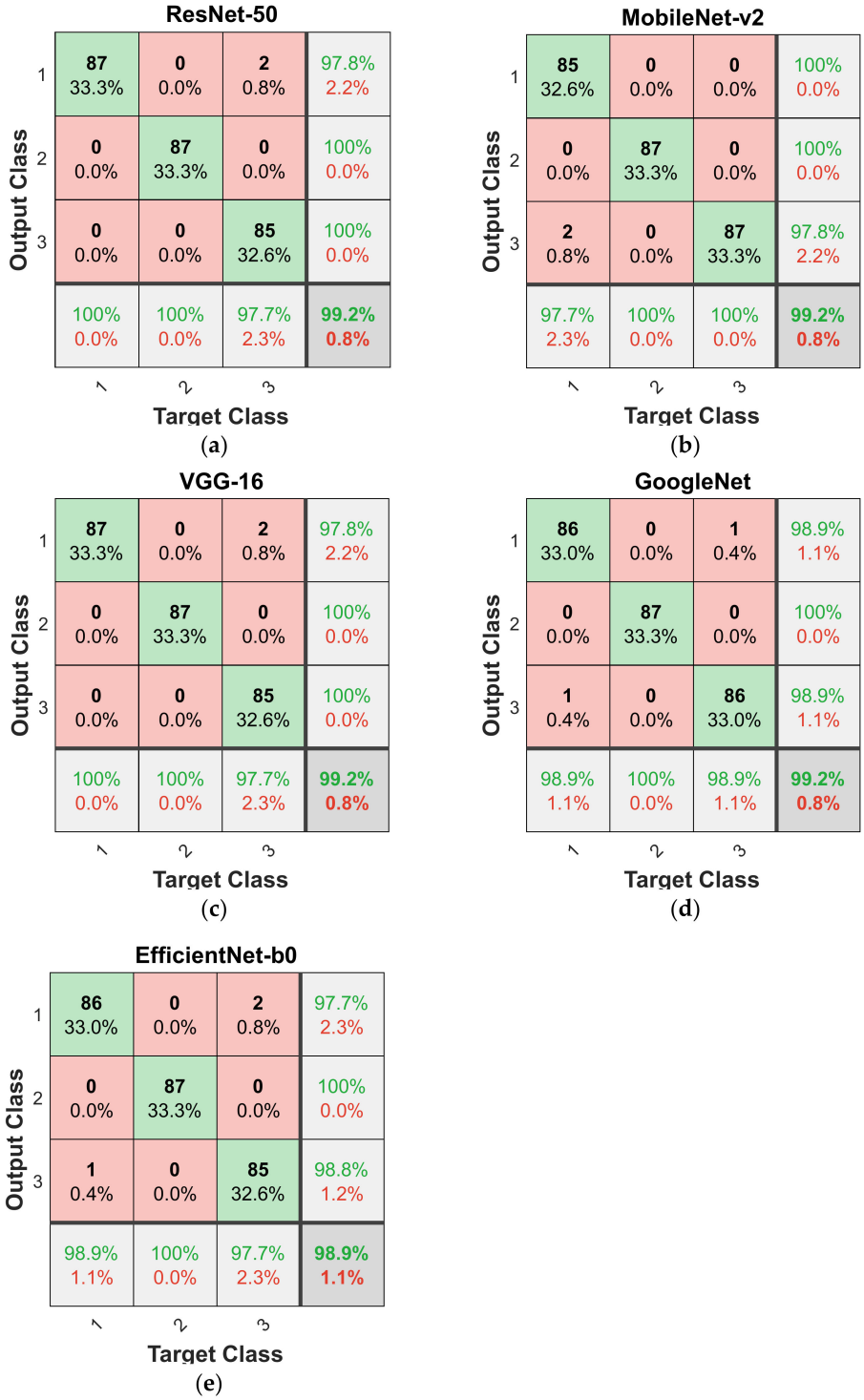


Figure 14. Confusion matrices of the pre-trained CNN models listed in Table 2. (**a**) ResNet-50; (**b**) MobileNet-v2; (**c**) VGG-16; (**d**) GoogleNet; (**e**) EfficientNet-b0.

Table 7. Performance metrics of the pre-trained CNN models.

Name	Accuracy	Precision	Recall	F-Score
ResNet-50	0.9923	0.9925	0.9923	0.9923
MobileNet-v2	0.9923	0.9925	0.9923	0.9923
VGG-16	0.9923	0.9925	0.9923	0.9923
GoogleNet	0.9923	0.9923	0.9923	0.9923
EfficientNet-b0	0.9885	0.9885	0.9885	0.9885

4. Discussion

4.1. Custom and Pre-Trained CNN Models Comparison

The custom and pre-trained CNN model results are compared based on accuracy and model complexity. Model complexity is characterised by size and execution time. Size is determined by the memory space required to store the Matlab-based model. Execution time is calculated as the total time taken to execute ten iterations of classification on the test dataset. Table 8 details the corresponding values of the various models. The execution time and size of each custom CNN model are much smaller than those of the pre-trained CNN models. This is due to the custom models utilising grayscale image inputs instead of RGB image inputs. The use of grayscale images leads to the lightweight baseline model shown in Figure 9. The accuracy of Model 30-2-2, Model 100-2-3, and Model 224-2-7 compare favourably with the accuracies of the pre-trained CNN models. This permits the use of reduced computational resources during the inference process, where repeated calls to the classification algorithm are typically made.

Table 8. Comparison of custom and pre-trained CNN models.

Model Name	Accuracy	Execution Time (s)	Size (KB)
Model 10-2-2	0.9655	1.36	23
Model 20-2-2	0.9847	1.39	32
Model 30-2-2	0.9885	1.49	43
Model 100-2-3	0.9923	1.81	74
Model 224-2-7	0.9962	2.96	28
ResNet-50	0.9923	53.73	85,709
MobileNet-v2	0.9923	18.44	8294
VGG-16	0.9923	146.78	487,424
GoogleNet	0.9923	25.07	21,709
EfficientNet-b0	0.9885	30.74	14,746

Accuracy and execution time are combined into a single score using Equation (9). Execution time is normalised using a maximum value of 150 to account for the highest execution time in Table 8. Since accuracy is of primary importance, it is given a weighting of 0.99 (or 99%). Table 9 ranks each model based on descending score values. A higher score implies a better model. Two custom CNN models, Model 224-2-7 and Model 100-2-3, have the highest scores. The top-scoring pre-trained CNN model (MobileNet-v2) is in third place.

Table 9. Ranking of CNN models based on descending score values.

Model Name	Score
Model 224-2-7	0.9960
Model 100-2-3	0.9923
MobileNet-v2	0.9911
GoogleNet	0.9907
ResNet-50	0.9888
Model 30-2-2	0.9885
EfficientNet-b0	0.9865
Model 20-2-2	0.9848
VGG-16	0.9826
Model 10-2-2	0.9658

$$\text{Score} = 0.99 \times \text{Accuracy} + 0.01\left(\frac{150 - \textit{Execution Time}}{150}\right) \quad (9)$$

4.2. Comparison with Other Studies

Grayscale images are employed by the lightweight custom CNN models presented in this paper. This inherently reduces the complexity of the classification models and facilitates implementation on standard laptop computers. However, classification accuracy should be on par with other research studies that tend to use RGB images. Table 10 compares the three best lightweight custom CNN models with the properties of a selection of other deep learning models from Section 1. The accuracy of the lightweight custom CNN models compares favourably with the models developed by other researchers.

Table 10. Comparison of lightweight custom CNN models with other deep learning models.

Reference	Dataset Properties	Classes	Model Complexity	Accuracy
Model 224-2-7	Grayscale, 224 × 224 pixels, 1734 images	3	Lightweight custom CNN with 7 layers	99.6%
Model 100-2-3	Grayscale, 100 × 100 pixels, 1734 images	3	Lightweight custom CNN with 7 layers	99.2%
Model 30-2-2	Grayscale, 30 × 30 pixels, 1734 images	3	Lightweight custom CNN with 7 layers	98.85%
Atik (2022) [13]	RGB, 227 × 227 × 3 pixels, 5332 images	3	Custom CNN with 13 layers	98.99%
Huang et al. (2019) [22]	RGB, 416 × 416 × 3 pixels, 43,160 images	4	YOLO-V3-Mobilenet with 30 layers	95.21% mAP
Hozyn (2023) [24]	RGB, 152 × 202 × 3 pixels, 3994 images	11	Custom CNN with 11 layers	96.5%
Cheng et al. (2022) [23]	RGB, 224 × 224 × 3, 3094 images	17	Siamese network with VGG-16, 41 layers	94%
Guo et al. (2021) [21]	RGB, 608 × 608 × 3 pixels, 12,000 images	20	YOLOv4-tiny + MAM with 24 layers	98.6% mAP
Xu et al. (2020) [11]	RGB, 112 × 112 × 3 pixels, 40,000 images	22	Faster SqueezeNet with 23 layers	99.999% TPR when FPR = 10^{-6}

Most of the models surveyed in Section 1 and Table 10 primarily focus on improving accuracy without considering model complexity. Atik [13] did not compare the execution time of their custom CNN model with pre-trained models. The VGG-16-based Siamese network model developed by Cheng, Wang, and Wu [23] focuses on reducing the dataset requirements for training a model but does not consider execution time.

Variations in input image resolution and class categories limit the ability to directly compare computation volume and speed across the various models. However, the results from the previous section indicate that RGB image input models are likely to be more complex and require increased computational resources. How much computational resources are saved cannot be directly quantified as this varies for different processors and computer configurations. However, execution times on the same processor/computing device configuration have been compared, and reduced execution time implies lower complexity, which in turn implies reduced computational resource requirements.

Hożyń [24] compared the execution time of their custom CNN model with pre-trained models. The ratio of their custom model's execution time to VGG-16 and ResNet-50 is 0.1792 and 0.2035, respectively. In this research, the ratio of the lightweight custom CNN model's (Model 224-2-7) execution time to VGG-16 and ResNet-50 is 0.0202 and 0.0551, respectively. Lower ratio values for the lightweight custom CNN model imply less computational complexity and lower computational requirements.

Some of the other models surveyed can classify more than ten types of components. It would be useful to investigate if lower-resolution grayscale images can be used to achieve similar results when the number of classes is increased.

5. Conclusions

This paper presents an empirical study on classifying used electronic parts with lightweight custom CNN deep learning models for efficient classification. The classification of used parts forms part of a recycling process in a circular economy for environmental sustainability. Moreover, the high uptake of AI automated systems is placing increasing demands on energy resources. Therefore, the development of deep learning systems should balance complexity against potential performance improvements. In this respect, the lightweight custom CNN model employs grayscale images to reduce model complexity. Various resolutions of grayscale images have been tested. Lightweight models utilising 100×100 pixels and 224×224 pixels grayscale images can achieve the same accuracy as pre-trained CNN models such as ResNet-50, MobileNet-v2, VGG-16, and EfficientNet-b0. When scoring the tested deep learning models with 1% weight for execution time and 99% weight for accuracy, two of the lightweight custom CNN models ranked highest. Similar metrics for model complexity, such as floating point operations (FLOPs), could be used to score and rank deep learning models for balancing complexity and performance (accuracy).

The broader implications of this study highlight the increasing energy demands associated with the advancement of AI automated systems. As global energy consumption inevitably continues to rise, it is imperative to consider the impact of deploying advanced automation technologies. The development and deployment of deep learning models must prioritize energy efficiency for sustainable energy utilisation.

For practitioners, it is recommended to adopt lightweight models that maintain high accuracy while reducing computational complexity and energy consumption. This approach not only supports environmental sustainability but can also enhance the scalability and accessibility of AI solutions. Policymakers should incentivise research and development in energy-efficient AI technologies and establish regulations that promote the use of sustainable practices in the tech industry.

Future work will involve extending the number of part classes that the lightweight custom CNN models can detect. Further investigation into the use of various resolution grayscale images and their performance evaluation will be conducted. Balancing the performance and complexity of deep learning models is crucial for sustainable AI development. By adopting energy-efficient practices and supporting policies, sustainable AI can be realised.

Author Contributions: Conceptualization, P.C.; methodology, P.C.; software, P.C. and M.A.; validation, P.C. and M.A.; formal analysis, P.C.; investigation, P.C. and M.A.; resources, P.C. and M.A.; data curation, P.C.; writing—original draft preparation, P.C.; writing—review and editing, P.C. and M.A.; visualization, P.C.; project administration, P.C.; funding acquisition, P.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflicts 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. Janiesch, C.; Zschech, P.; Heinrich, K. Machine learning and deep learning. *Electron. Mark.* **2021**, *31*, 685–695. [CrossRef]
3. Sharma, N.; Sharma, R.; Jindal, N. Machine Learning and Deep Learning Applications-A Vision. *Glob. Transit. Proc.* **2021**, *2*, 24–28. [CrossRef]

4. Sarker, I.H. Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions. *SN Comput. Sci.* **2021**, *2*, 420. [CrossRef] [PubMed]
5. Chand, P.; Lal, S. Vision-based Detection and Classification of Used Electronics Parts. *Sensors* **2022**, *22*, 9079. [CrossRef]
6. Wang, Y.; Zhou, Y.; Wei, L.; Li, R. Design of a Four-Axis Robot Arm System Based on Machine Vision. *Appl. Sci.* **2023**, *13*, 8836. [CrossRef]
7. Sun, R.; Wu, C.; Zhao, X.; Zhao, B.; Jiang, Y. Object Recognition and Grasping for Collaborative Robots Based on Vision. *Sensors* **2024**, *24*, 195. [CrossRef]
8. 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. 1–5.
9. Ionescu, R.T.; Popescu, M. Object Recognition with the Bag of Visual Words Model. In *Knowledge Transfer between Computer Vision and Text Mining: Similarity-Based Learning Approaches*; Springer International Publishing: Cham, Switzerland, 2016; pp. 99–132.
10. Goobar, L. *Machine Learning Based Image Classification of Electronic Components*; KTH: Stockholm, Sweden, 2013.
11. 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]
12. Alzubaidi, L.; Zhang, J.; Humaidi, A.J.; Al-Dujaili, A.; Duan, Y.; Al-Shamma, O.; Santamaría, J.; Fadhel, M.A.; Al-Amidie, M.; Farhan, L. Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. *J. Big Data* **2021**, *8*, 53. [CrossRef]
13. Atik, I. Classification of Electronic Components Based on Convolutional Neural Network Architecture. *Energies* **2022**, *15*, 2347. [CrossRef]
14. Mathworks. Pretrained Deep Neural Networks. Available online: https://au.mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks.html (accessed on 18 March 2024).
15. Verdecchia, R.; Sallou, J.; Cruz, L. A systematic review of Green AI. *WIREs Data Min. Knowl. Discov.* **2023**, *13*, e1507. [CrossRef]
16. Tripp, C.; Bensen, E.; Hayne, L.; Perr-Sauer, J.; Lunacek, M. Green AI: Insights Into Deep Learning's Looming Energy Efficiency Crisis. In Proceedings of the Conference: Presented at the AI and Electric Power Summit, Rome, Italy, 4–6 October 2022; U.S. Department of Energy: Washington, DC, USA, 2022.
17. Salehi, S.; Schmeink, A. Data-Centric Green Artificial Intelligence: A Survey. *IEEE Trans. Artif. Intell.* **2024**, *5*, 1973–1989. [CrossRef]
18. Desislavov, R.; Martínez-Plumed, F.; Hernández-Orallo, J. Trends in AI inference energy consumption: Beyond the performance-vs-parameter laws of deep learning. *Sustain. Comput. Inform. Syst.* **2023**, *38*, 100857. [CrossRef]
19. Luccioni, S.; Jernite, Y.; Strubell, E. Power Hungry Processing: Watts Driving the Cost of AI Deployment? In Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency, Rio de Janeiro, Brazil, 3–6 June 2024; pp. 85–99.
20. 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]
21. 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]
22. 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]
23. Cheng, Y.; Wang, A.; Wu, L. A Classification Method for Electronic Components Based on Siamese Network. *Sensors* **2022**, *22*, 6478. [CrossRef]
24. Hożyń, S. Convolutional Neural Networks for Classifying Electronic Components in Industrial Applications. *Energies* **2023**, *16*, 887. [CrossRef]
25. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
26. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In Proceedings of the Advances in Neural Information Processing Systems 28 (NIPS 2015), Montreal, QC, Canada, 7–12 December 2015; pp. 1–9.
27. Chand, P. A Low-Resolution Used Electronic Parts Image Dataset for Sorting Application. *Data* **2023**, *8*, 20. [CrossRef]
28. Crawford, K. Generative AI is guzzling water and energy. *Nature* **2024**, *626*, 693. [CrossRef]
29. Chand, P. Developing a Matlab Controller for Niryo Ned Robot. In Proceedings of the 2022 International Conference on Technology Innovation and Its Applications (ICTIIA), Tangerang, Indonesia, 23–25 September 2022; pp. 1–5.
30. Niryo. NED User Manual. Available online: https://docs.niryo.com/robots/ned/ (accessed on 30 June 2022).
31. Ting, K.M. Confusion Matrix. In *Encyclopedia of Machine Learning*; Sammut, C., Webb, G.I., Eds.; Springer: Boston, MA, USA, 2010; p. 209.
32. Mathworks. Create Simple Deep Learning Network for Classification. Available online: https://au.mathworks.com/help/deeplearning/ug/create-simple-deep-learning-network-for-classification.html (accessed on 5 October 2023).

33. Hosna, A.; Merry, E.; Gyalmo, J.; Alom, Z.; Aung, Z.; Azim, M.A. Transfer learning: A friendly introduction. *J. Big Data* **2022**, *9*, 102. [CrossRef]
34. Mathworks. Train Deep Learning Network to Classify New Images. Available online: https://au.mathworks.com/help/deeplearning/ug/train-deep-learning-network-to-classify-new-images.html#TransferLearningUsingGoogLeNetExample-2 (accessed on 26 June 2024).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.