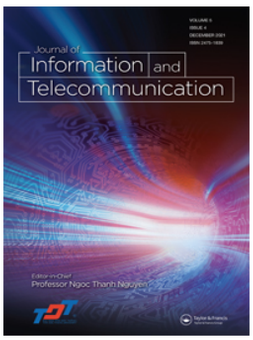

Taylor & Francis
Taylor & Francis Group

Journal of Information and Telecommunication

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/tjit20

New convolutional neural network models for efficient object recognition with humanoid robots

Simge Nur Aslan, Ayşegül Uçar & Cüneyt Güzeliş

To cite this article: Simge Nur Aslan, Ayşegül Uçar & Cüneyt Güzeliş (2021): New convolutional neural network models for efficient object recognition with humanoid robots, Journal of Information and Telecommunication, DOI: 10.1080/24751839.2021.1983331

To link to this article: https://doi.org/10.1080/24751839.2021.1983331

Published online: 06 Oct 2021.

JOURNAL OF INFORMATION AND TELECOMMUNICATION
https://doi.org/10.1080/24751839.2021.1983331

New convolutional neural network models for efficient object recognition with humanoid robots

Simge Nur Aslan [a], Ayşegül Uçar [a] and Cüneyt Güzeliş [b]

[a]Mechatronics Engineering Department, Firat University, Elazig, Turkey; [b]Electrical and Engineering Department, Yasar University, Izmir, Turkey

ABSTRACT

Humanoid robots are expected to manipulate the objects they have not previously seen in real-life environments. Hence, it is important that the robots have the object recognition capability. However, object recognition is still a challenging problem at different locations and different object positions in real time. The current paper presents four novel models with small structure, based on Convolutional Neural Networks (CNNs) for object recognition with humanoid robots. In the proposed models, a few combinations of convolutions are used to recognize the class labels. The MNIST and CIFAR-10 benchmark datasets are first tested on our models. The performance of the proposed models is shown by comparisons to that of the best state-of-the-art models. The models are then applied on the Robotis-Op3 humanoid robot to recognize the objects of different shapes. The results of the models are compared to those of the models, such as VGG-16 and Residual Network-20 (ResNet-20), in terms of training and validation accuracy and loss, parameter number and training time. The experimental results show that the proposed model exhibits high accurate recognition by the lower parameter number and smaller training time than complex models. Consequently, the proposed models can be considered promising powerful models for object recognition with humanoid robots.

ARTICLE HISTORY
Received 24 August 2021
Accepted 4 September 2021

KEYWORDS
Humanoid robots; Convolution neural networks; object recognition

1. Introduction

Nowadays, the humanoid robots are increasingly used in a lot of areas, such as the medicine, the education, the healthcare, the logistics and the house and hotel services, to improve the quality of human life (Andtfolk et al., 2021; Angelopoulos et al., 2021; Dannecker & Hertig, 2021; Garcia-Haro et al., 2021; Nenchev et al., 2018; Oliver et al., 2021). The humanoid robots were used instead of humans or together with humans (Ambrose et al., 2001; Chohra & Madani, 2018; Fitzpatrick & Metta, 2003; Levine et al., 2016; Reforgiato Recupero, 2021; Sakagami et al., 2002). The humanoid robot used in (Fitzpatrick & Metta, 2003) learns the objects around it by using its body so that it can define and interpret its environment. In this case, it is considered the main indicator in the

CONTACT Ayşegül Uçar agulucar@firat.edu.tr

information it can provide vision. Computer vision technologies are generally used in applications as done in this study. For example, computer vision has been used in robotic applications for tasks such as obstacle avoidance and navigation (Chang, 2010; Pandey & Gelin, 2017), human-robot interaction (Le et al., 2018; Yavşan & Uçar, 2016), object detection for assisting robots, recognition (Martinez-Martin & Del Pobil, 2017), and object recognition for capture (Aslan et al., 2020; 2021; Ku et al., 2017; Levine et al., 2016).

Object recognition and detection is essential for humanoid robots, mobile robots, robotic arms, and flying robots etc., to be able to interpret and understand their environment. Many machine learning algorithms have already been proposed and used in object recognition and detection applications (Bhuvaneswari & Subban, 2018; Lee, 2015). Traditional machine learning methods, such as random forest, Support Vector Machines (SVMs), Adaptive boosting, and Feed Forward Neural Networks (FNNs), have some limitations in their ability to process raw data (Alpaydin, 2016; Haykin, 2009). The methods require some attributes for inputs. Hence, they need to extract an attribute from classifiers. They cannot demonstrate end-to-end solutions without using the feature engineering step.

Deep learning methods for object recognition have become a popular field of study in recent years (Lee, 2015; Xie et al., 2017). The deep learning model is a graph of layers without a direct loop. The most common example is that it is a linear set of layers that maps a single input to a single output (Chollet, 2017). Deep Neural Networks (DNNs) can outperform traditional feature-based approaches at the computer vision tasks such as perception, recognition, and segmentation (Girshick, 2015; Girshick et al., 2014; Redmon & Farhadi, 2017; Ren et al., 2015). Convolutional Neural Networks (CNNs) are also a type of DNNs (Srinivas et al., 2016). While the feature vectors are used as the inputs in classical machine learning methods, the image can be used directly in CNNs. Therefore, when CNNs are used for classification, they perform better than conventional machine learning methods (Lee, 2015). But, the DNN models, such as AlexNet (Krizhevsky et al., 2012), GoogleNet (Szegedy et al., 2015), Inception V3 (Szegedy et al., 2016), Residual Network (ResNet) (He et al., 2016) and VGG-16 (Simonyan & Zisserman, 2014), are shown as particularly deep strong structures. However, these models need a large memory to store the weights and a lot of time for performance. DNNs usually suffer from excessive parameterization and often encode highly correlated parameters, resulting in inefficient computing and memory usage (Chen et al., 2015; Han et al., 2015). This paper proposes new models to get rid of this disadvantage.

Recent works on effective deep learning have focused on model compression and reducing the computational operations in DNNs (Chen et al., 2015; Denil et al., 2013; 2014; Han et al., 2015; Idelbayev & Carreira-Perpinán, 2021; Lee et al., 2021; Rastegari et al., 2016). Chen et al. (2015) presented a new network architecture called HashedNets that takes the advantage of inherent redundancy. Han et al. (2015) introduced a new deep compression network with pruning, trained quantization and Huffman coding. Rastegari et al. (2016) proposed simple and accurate binary approximations to apply faster convolutional operations. Denil et al. (2013) presented a parameter prediction technique for reducing the parameter number in DNNs. Jaderberg et al. (2014) used a cross-channel or filter redundancy to generate low-rank filters for speeding up pre-trained CNNs with minimal loss of accuracy. In Lee et al. (2021); Idelbayev and Carreira-Perpinán (2021), a

tensor compose-decompose approximation and an approximation combining low-rank decompositions, using different matrix shapes, were proposed to obtain the compressed DNNs with high performance.

Several works have been carried out to decrease the parameter number of DNNs (Ayinde & Zurada, 2018; Gong et al., 2014; Gowda & Yuan, 2018; Huang et al., 2017; Iandola et al., 2016; Jha et al., 2020; Krizhevsky et al., 2012; Srinivas & Babu, 2015; Yang et al., 2015). Gong et al. (2014); Yang et al. (2015); Srinivas and Babu (2015) rely on the compression of fully connected layers bearing most of the weights. They did not improve the speed of the network. In Iandola et al. (2016), a small DNN architecture is proposed that is completely independent and has 50 times fewer parameters than AlexNet (Krizhevsky et al., 2012), but it is slower. Jha et al. (2020) proposed the LightLayers network to reduce the number of parameters in DNNs. It consists of the classical Conv2D and Dense layers, based on matrix decomposition. In Ayinde and Zurada (2018), a pruning technique was provided for removing redundant features in CNNs. Huang et al. (2017) introduced the Dense Convolutional Network (DenseNet) using direct connections between any two layers with the same feature-map size. In Gowda and Yuan (2018), the images of seven different colour spaces were used together with dense networks for increasing the performance.

This paper proposes four CNN models with small structure without using additional pruning or decomposing steps different from the literature. The performances of the proposed models with the popular models in the literature are comparatively presented on the benchmark the MNIST and CIFAR-10 datasets (Krizhevsky & Hinton, 2009; LeCun et al., 1998). The Robotis-Op3 humanoid robot is used for real object recognition application (Robotis-Op3, 2020). The performances of all models and the VGG-16 and ResNet-20 models generated using transfer learning are evaluated in terms of the training and validation accuracy, the training and validation loss, the parameter number, and the training time.

The rest of this paper is organized as follows: in Section 2 the basic layers of the CNNs are shortly introduced and are presented the proposed CNN models; The comparative results of the MNIST and CIFAR-10 datasets are presented in Section 3.1. In Section 3.2, the object recognition with humanoid robots is carried out and the simulation results are illustrated to demonstrate the performance of the proposed models. Section 4 concludes this paper.

2. Convolution neural networks

CNNs are similar to conventional FNNs, optimizing their weight and bias by means of self-learning (Girshick et al., 2014; Srinivas et al., 2016). Each input neuron receives an input and employs a mathematical operation such as a product and linear or nonlinear activation function. The obtained layer outputs are sent to the next layer by the weights. The final layer includes a loss activation function to reach to the ground truth values. The main difference between CNNs and FNNs is that CNNs can use the images as input. This property drastically reduces the parameter number with respect to ANN importing a vector to its input.

CNNs are composed of the input layer including the images with usually three dimensions (height, width, and depth), convolutional layers, nonlinear activation functions,

pooling layers, dropout layer, batch normalization layer, one or multiple fully connected layers called dense layers, and loss activation layer. A simple CNN structure is illustrated in Figure 1.

The main functionalities of the five layers are defined as follows:

(1) The input layer receives the image pixel values as the inputs.
(2) Convolution layer includes a set of learnable filters. The filter dimensions are smaller than the input dimensions. The local receptive regions of the input are connected to the neurons in the next layer via a dot product between the weights and the inputs in the region at every spatial dimension. In other words, each filter is moved across the width and height of the input and produces a feature map. Figure 2 depicts the receptive fields and the feature maps. Different from FNNs, CNNs share the same weights for all local regions. The weight sharing reduces the parameter number and provides the learning and expression efficiency, good generalization, and invariant against translation.

Given $\boldsymbol{I_i}$= Input[:, :, $\boldsymbol{i}$] and the kernel K in the form of square weight matrix, the convolution is applied as

$$(I*K) = \sum_m \sum_n \mathrm{K}_{\mathrm{i}-\mathrm{m},\mathrm{j}-\mathrm{n}} \mathrm{I}_{\mathrm{m},\mathrm{n}} \tag{1}$$

where the asterisk shows the convolution. 0-th feature map, F_0, is calculated as

$$F_0 = g\left(\sum_{i=0} \mathrm{I}_\mathrm{i} * \mathrm{K}_{\mathrm{i},0} + \mathrm{b}_0 1\right) \tag{2}$$

where g is the activation function and b is the bias value. After the convolution process by using each filter, the image features, such as edge and corners, are extracted.

(1) Pooling layer applies down-sampling along the spatial dimension of the convolution layers. Thanks to the pooling layer, the overfitting problem and vanishing gradient problem are slighted. Min, max, or average pooling are some of the pooling methods.

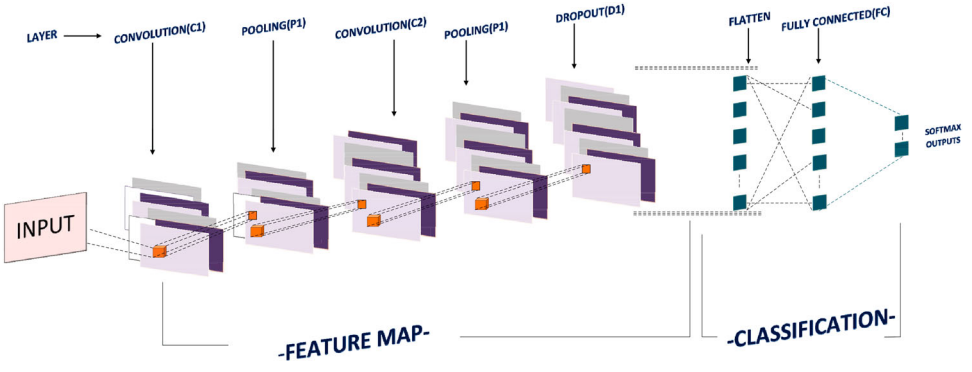

Figure 1. An CNN structure and the layer representation.

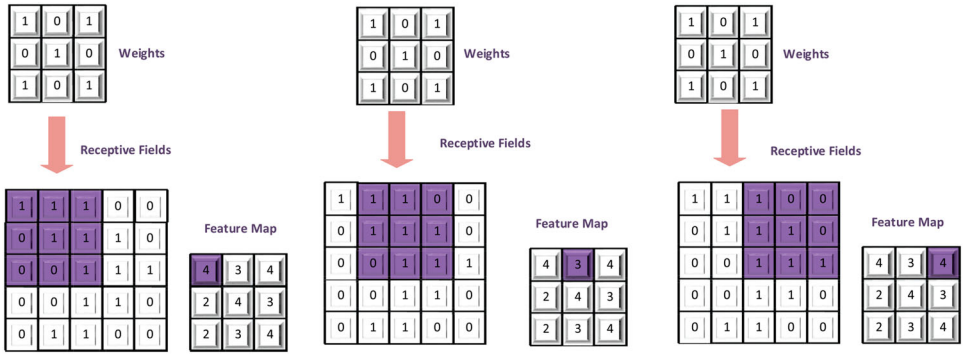

Figure 2. The results of the convolution processes for one depth channel of the image.

(2) Dropout layer is a regularization method. During the training stage, some nodes of the layer are randomly and temporarily removed to prevent the overfitting. After the dropout layer, the nodes are usually flat by being transformed to a vector.

(3) Batch normalization layer normalizes the outputs of the layer as follows:

$$c = \frac{1}{o}\sum_{i=1}^{o} z_i \tag{3}$$

$$s^2 = \frac{1}{o}\sum_{i=1}^{o} (z_i - c)^2 \tag{4}$$

$$\tilde{z}_i = \frac{z_i - c}{\sqrt{s^2 + \in}} \tag{5}$$

where c, s, and o are the mean, standard deviation, and mini-batch size, respectively. $\tilde{z}$ is the normalized value and $\in$ is a small number. The batch normalization makes the

Table 1. The structure of Model 1.

Type	Output	Feature maps
Input	Nxn	1 or 3
Convolution 1	Nxn	32
Activation 1	Nxn	32
Max Pooling 1	(nxn)/2	32
Dropout 1	(nxn)/2	32
Convolution 2	(nxn)/2	64
Activation 2	(nxn)/2	64
Max Pooling 2	(nxn)/4	64
Dropout 2	(nxn)/4	64
Convolution 3	(nxn)/4	128
Activation 3	(nxn)/4	128
Max Pooling 3	((nxn)/7)-1	128
Dropout 3	((nxn)/7)-1	128
Convolution 4	((nxn)/7)-1	256
Activation 4	((nxn)/7)-1	256
Max Pooling 4	(nxn)/n	256
Dropout 4	(nxn)/n	256
Flatten 1	256	1
Dense 1	80	1
Activation 5	80	1
Dropout 4	80	1
Dense 2	C	1
Activation 6	C	1

Table 2. The structure of Model 2.

Type	Output	Feature maps
Input	nxn	1 or 3
Convolution 1	nxn	16
Activation 1	nxn	16
Batch Normalization 1	nxn	16
Convolution 2	nxn	16
Activation 2	nxn	16
Batch Normalization 2	nxn	16
Max Pooling 1	(nxn)/2	16
Dropout 1	(nxn)/2	16
Convolution 3	(nxn)/2	32
Activation 3	(nxn)/2	32
Batch Normalization 3	(nxn)/2	32
Convolution 4	(nxn)/2	32
Activation 4	(nxn)/2	32
Batch Normalization 4	(nxn)/2	32
Max Pooling 2	(nxn)/4	32
Dropout 2	(nxn)/4	32
Convolution 5	(nxn)/4	64
Activation 5	(nxn)/4	64
Batch Normalization 5	(nxn)/4	64
Convolution 6	(nxn)/4	64
Activation 6	(nxn)/4	64
Batch Normalization 6	(nxn)/4	64
Max Pooling 3	((nxn)/7)-1	64
Dropout 3	((nxn)/7)-1	64
Flatten 1	1152	1
Dense 1	C	1

Table 3. The structure of Model 3.

Type	Output	Feature maps
Input	nxn	1 or 3
Convolution 1	nxn	16
Max Pooling 1	(nxn)/2	16
Convolution 2	(nxn)/2	32
Max Pooling 2	(nxn)/4	32
Flatten 1	1600	1
Dense 1	C	1

training faster and the performance of the network higher. The batch normalization is usually used together with Rectified Linear Unit Activation Function (ReLU). The ReLU function is defined as max(0, input). The ReLU and its derivative calculation are easy and fast. The derivative of ReLU has 0 value for the negative input, otherwise it has 1 value. Hence, there is no vanishing gradient problem when applying ReLU (Girshick et al., 2014; Srinivas et al., 2016).

(4) Fully connected layers perform the same duties in FNNs by connecting every neuron in the current layer to every neuron in the next layer. Before the layer is used, the data are transformed into a vector. The operation is called flatten. These layers include a nonlinear activation function or a softmax activation to produce the class scores (Girshick et al., 2014; Srinivas et al., 2016). The softmax function is used in the classification applications. The function assigns with the probability each class by making the total

Table 4. The structure of Model 4.

Type	Output	Feature Maps
Input	nxn	1 or 3
Convolution 1	nxn	32
Activation 1	nxn	32
Max Pooling 1	(nxn)/2	16
Dropout 1	(nxn)/2	32
Convolution 2	(nxn)/2	64
Activation 2	(nxn)/2	64
Max Pooling 2	(nxn)/4	64
Dropout 2	(nxn)/4	64
Convolution 3	(nxn)/4	128
Activation 3	(nxn)/4	128
Max Pooling 3	((nxn)/7)-1	128
Dropout 3	((nxn)/7)-1	128
Flatten 1	1152	1
Dense 1	80	1
Activation 4	80	1
Dropout 4	80	1
Dense 2	C	1
Activation 5	C	1

probability 1. The function is defined as follows:

$$softmax(z_j) = \frac{e^{z_k}}{\sum_{i=1}^{C} e^{z_i}} \quad for\ j = 1, \ldots, C \tag{6}$$

where C is the class number.

2.1. Proposed convolutional neural network models

In this paper, we propose new four CNN models with small structure instead of large ones using multiple convolutions for the object recognition with humanoid robots. Our models are listed in Tables 1–4.

The structure of Model 1 consists of four sequential blocks: convolution, activation, max pooling, and dropout. The layers of the model are followed with the flatten, dense, activation, dropout, dense, and activation layers. In Model 1, the dimensions of the feature maps are 16, 32, 64, 128, and 256, respectively. Model 2 is generated from

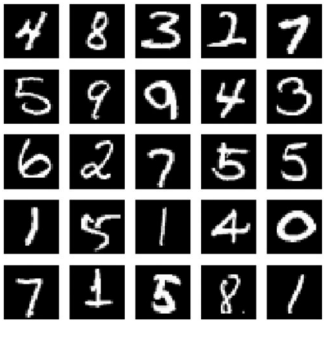

Figure 3. Some examples from the MNIST dataset (LeCun et al., 1998).

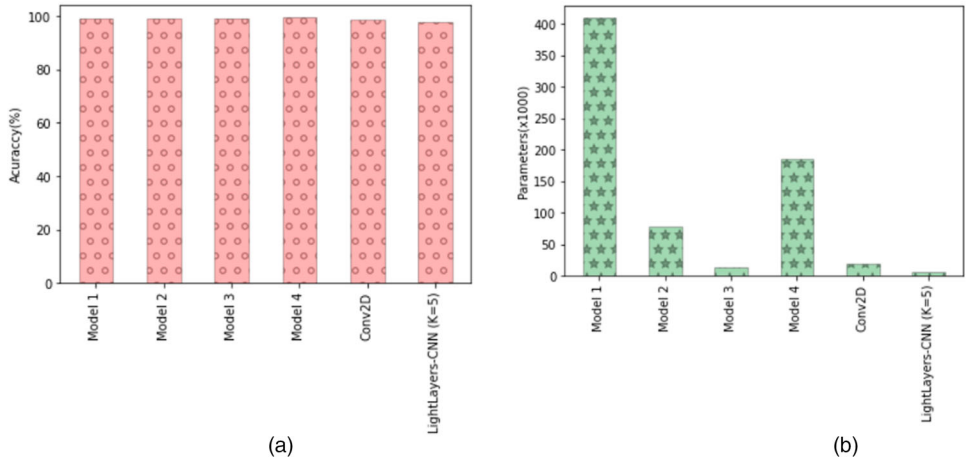

Figure 4. The results on the MNIST dataset (a) test accuracy and (b) parameter number.

three sequential blocks, including convolution, activation, batch normalization, convolution, activation, batch normalization, max pooling, and dropout. The layers of the model are followed with the flatten and dense layers. In Model 2, the dimensions of the feature maps are 16, 32, and 64, respectively. Model 3 includes Convolution, Max Pooling, Convolution, Max Pooling, Flatten, and Dense layers. In Model 3, the dimensions of the feature maps are 16 and 32, respectively. Model 4 is constructed by three sequential blocks, including convolution, activation, max pooling, and dropout. The layers of the model are followed with flatten, dense, activation, dropout, dense, and activation layers. In Model 4, the dimensions of the feature maps are 32, 64, and 128, respectively. In all models, we used a 3×3 filter, the ReLU activation function, and same convolution and we adaptively generated our model structure with respect to the input image size, nxn, and class number, C.

3. Experiments

To evaluate the performance of the proposed CNN models, we conducted two different experiments. Firstly, we used the MNIST and CIFAR-10 datasets and evaluated the performance of our models on the datasets (Krizhevsky & Hinton, 2009; LeCun et al., 1998) to show the performance of our models over the models in the literature. Secondly, we carried out the object recognition application with a real humanoid robot.

Table 5. Test accuracy and parameter number on the MNIST dataset.

Models	Test Accuracy (%)	Parameter number
Model 1	99.37	409,210
Model 2	99.25	78,458
Model 3	98.99	12,810
Model 4	99.40	185,722
ResNet-20	98.70	279,206
Conv2D (Jha et al., 2020)	98.87	18,818
LightLayers ($K = 3$) (Jha et al., 2020)	97.75	6,135

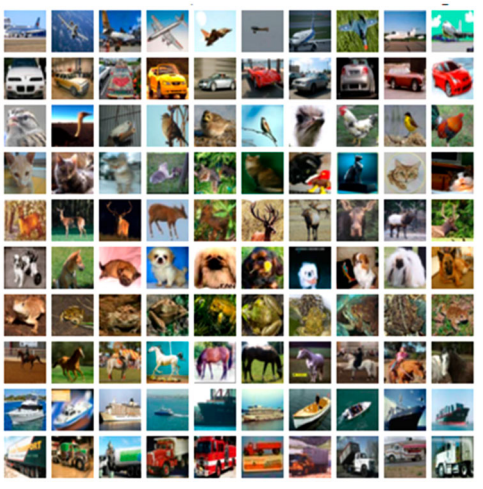

Figure 5. Some examples from the CIFAR-10 dataset (Krizhevsky & Hinton, 2009).

3.1. Experimental results on the MNIST dataset

We used firstly the benchmark MNIST dataset (LeCun et al., 1998). It consists of handwritten digits ranging from 0 to 9. This dataset contains a total of 70,000 data with 60,000 for training and 10,000 for testing. All images in the dataset have a size of 28 × 28. Figure 3 shows some examples of the MNIST digits.

In the experiments, we applied Adam optimization with a minimum batch size of 32 and the learning rates as 0.001, and cross entropy loss for 20 epochs. We trained all models on the MNIST dataset. Moreover, we transferred the pre-trained ResNet-20 model and then trained it also.

The results of accuracy and parameter number, relating to the proposed CNN models, are presented by means of the pink and green blocks in Figure 4 and the values in Table 5

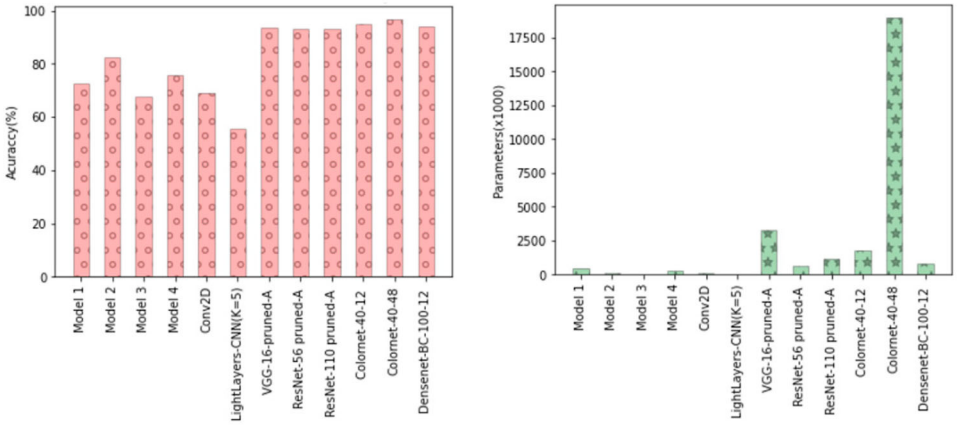

Figure 6. The results on the CIFAR-10 dataset (a) test accuracy and (b) parameter number.

Table 6. Test accuracy and parameter number on the CIFAR-10 dataset.

Models	Test accuracy (%)	Parameter number
Model 1	72.34	471,226
Model 2	**82.51**	**110,458**
Model 3	67.55	16,618
Model 4	75.55	257,978
Conv2D (Jha et al., 2020)	68.82	76,794
LightLayers-CNN ($K = 5$) (Jha et al., 2020)	55.76	20,557
VGG-16-pruned-A (Ayinde & Zurada, 2018)	93.67	3,230,000
ResNet-56 pruned-A (Ayinde & Zurada, 2018)	93.12	650,000
ResNet-110 pruned-A (Ayinde & Zurada, 2018)	93.27	1,130,000
Colornet-40-12 (Gowda & Yuan, 2018**)**	**95.02**	**1,750,000**
Colornet-40-48 (Gowda & Yuan, 2018)	96.86	19,000,000
Densenet-BC-100-12 (Huang et al., 2017)	94.08	800,000

list the test results relating to our four models and three CNN models: ResNet-20 (He et al., 2016), Conv2D (Jha et al., 2020), and LightLayers-CNN (Jha et al., 2020) from the literature. The accuracy values in percent of our models 1-4, ResNet-20, Conv2D, and LightLayers-CNN are 99.37, 99.25, 98.99, 99.40, 98.70, 98.87, and 97.75, respectively. The highest values of accuracy are 99.40 belonging to our model 4 and 99.25 belonging to our model 2, as shown in Figure 4.

The parameter number values of our models 1–4, ResNet-20, Conv2D and LightLayers-CNN are 409,210, 78,458, 12,810, 185,722, 279,206,1 8,818, and 6,135, respectively. The lowest value of parameter number is 6,135 belonging to LightLayers-CNN and 12,810 belonging to our model 3. The accuracy of our model 3 is 98.99, while that of LightLayers-CNN is 97.75. We can see that our model 3 has an acceptable trade-off between the accuracy and parameter number.

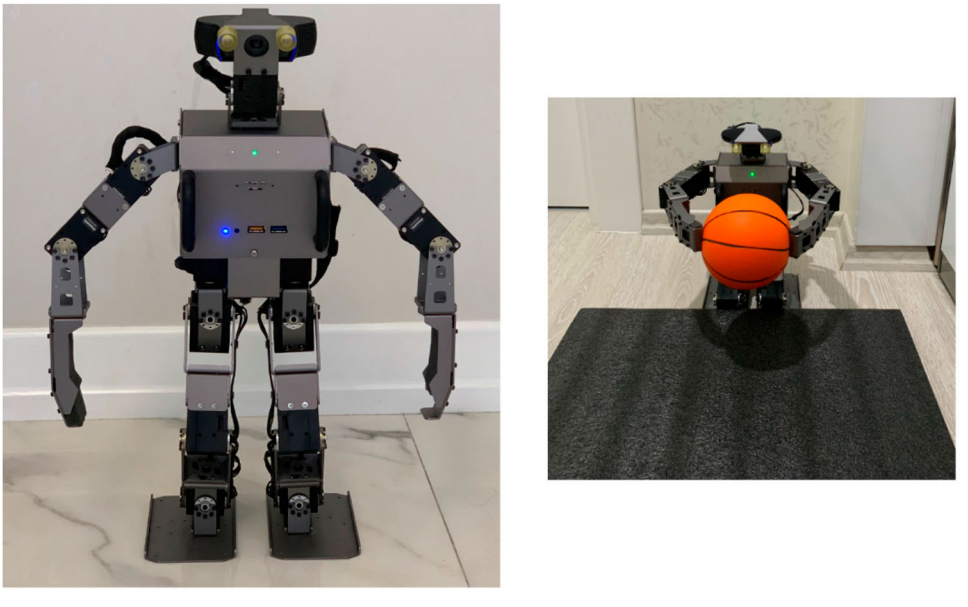

Figure 7. The Robotis-Op3 humanoid robot and its object manipulation photo.

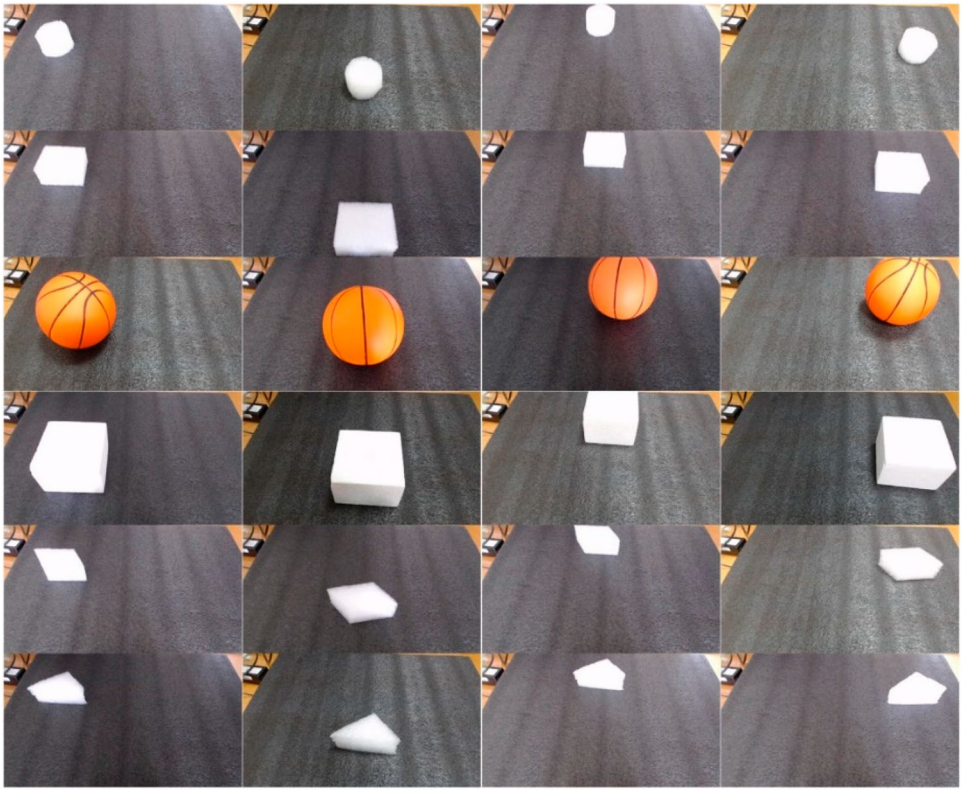

Figure 8. Some of the images used in the training dataset.

3.2. Experimental results on the CIFAR-10 datasets

We used firstly the benchmark the CIFAR-10 dataset (Krizhevsky & Hinton, 2009). It has 10 classes: aeroplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. It includes the colour images that are 50,000 in training and 10,000 in testing. The image size is 32 × 32. Figure 5 depicts a random sample of images belonging to the dataset.

In the experiments, we applied Adam optimization with a minimum batch size of 32 and the learning rates as 0.001, and cross entropy loss for 20 epochs. We trained all models on the CIFAR-10 dataset. The results of test accuracy, relating to the proposed CNN models, are presented by means of the pink and green blocks in Figure 6 and the values in Table 6. Table 5 lists the test results relating to our four models and eight CNN models called LightLayers-CNN (K = 5) (Jha et al., 2020), VGG-16-pruned-A (Ayinde & Zurada, 2018), ResNet-56 pruned-A (Ayinde & Zurada, 2018), ResNet-110 pruned-A (Ayinde & Zurada, 2018), Colornet-40-12 (Gowda & Yuan, 2018), Colornet-40-48 (Gowda & Yuan, 2018), and Densenet-BC-100-12 (Huang et al., 2017) from the literature. The accuracy values in percent of our models 1–4, LightLayers-CNN (K = 5), VGG-16-pruned-A, ResNet-56 pruned-A, ResNet-110 pruned-A, Colornet-40-12, Colornet-40-48, and Densenet-BC-100-12 are 72.34, 82.51, 67.55, 75.55, 68.82, 55.76, 93.67, 93.12, 93.27, 95.02, 96.86 and 94.08, respectively. The highest value of accuracy is 95.02 belonging to Colornet-40-12 (Gowda & Yuan, 2018). On the other hand, Model 2 has the highest accuracy (82.51) within our models, as in Figure 6.

Table 7. The results of accuracy, loss, parameter number, and training time on our dataset.

Models	Train loss	Train acc (%)	Val. loss	Val. Acc (%)	Parameter number	Training time (s)
Model 1	**0.039**	**99.11**	**0.0331**	**100.0**	**5,631,862**	**25.2065**
Model 1(28 × 28)	0.4368	81.38	0.3392	86.52	409,462	3.5011
Model 1(32 × 32)	0.2140	92.55	0.0606	100.0	470,902	3.4965
Model 2	**1.6122**	53.68	1.4546	50.35	466,198	82.7385
Model 2(28 × 28)	1.6442	43.59	1.3239	46.10	76,438	5.1617
Model 2 32 × 32)	1.7006	40.56	1.0583	52.48	79,126	5.3596
Model 3	**0.0013**	**100.0**	**0.0017**	**100.0**	**743,142**	**12.5431**
Model 3(28 × 28)	0.1562	97.34	0.1513	100.0	9,894	2.4821
Model 3(32 × 32)	0.0687	100.0	0.0776	100.0	12,006	2.4311
Model 4	**0.0298**	**99.29**	**3.2e-04**	**100.0**	**10,579,574**	**23.9928**
Model 4 (28 × 28)	0.2631	90.25	0.1464	97.87	185,974	3.1184
Model 4 (32 × 32)	0.2354	92.20	0.0954	100.0	257,654	3.1887
ResNet-20	**0.7451**	**77.32**	**0.6114**	**82.98**	**298,374**	**120.0167**
ResNet-20 (28 × 28)	0.6135	81.28	0.8342	73.05	292,614	17.1682
ResNet-20 (32 × 32)	0.5140	87.33	0.6437	79.43	298,374	14.3619
VGG-16	0.0281	99.08	0.0019	100.0	23,105,094	112.8447
VGG-16 (32 × 32)	0.5856	74.22	0.5816	75.89	14,847,558	15.86730

The parameter number values of all models are shown in Table 6. The lowest value of parameter number is 16,618 belonging to our model 3. However, its accuracy is low and its value is 67.55. The parameter number is 110,458. We can see that model 2 has an acceptable trade-off between the accuracy and parameter number.

3.3. Experimental results for object recognition with humanoid robots

In the object recognition application with the humanoid robots, we used the humanoid robot, Robotis-Op3 humanoid robot in Figure 7 (Robotis-Op3, 2020). Robotis-Op3 consists of 20 axes and includes Intel NUC i3 (Dual core, 2133 Mhz) operating system, 3 axis gyro, 3 axis magnetometer sensors, 3 axis accelerometer, Linux operating system, Logitech C920 HD-Pro camera, C, Robot Operating System (ROS), and Dynamixel SDK. The experiments show results on workstation using Nvidia Titan XP.

In the experiments, we used the real objects with 7 different shapes: a small square, a big square, a cylinder, a ball, a rhombus, a triangle in addition to a black sponge. Figure 8

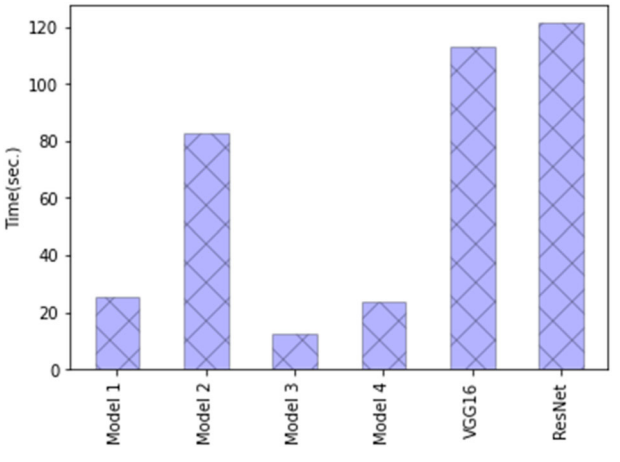

Figure 9. The training times of the models on our dataset.

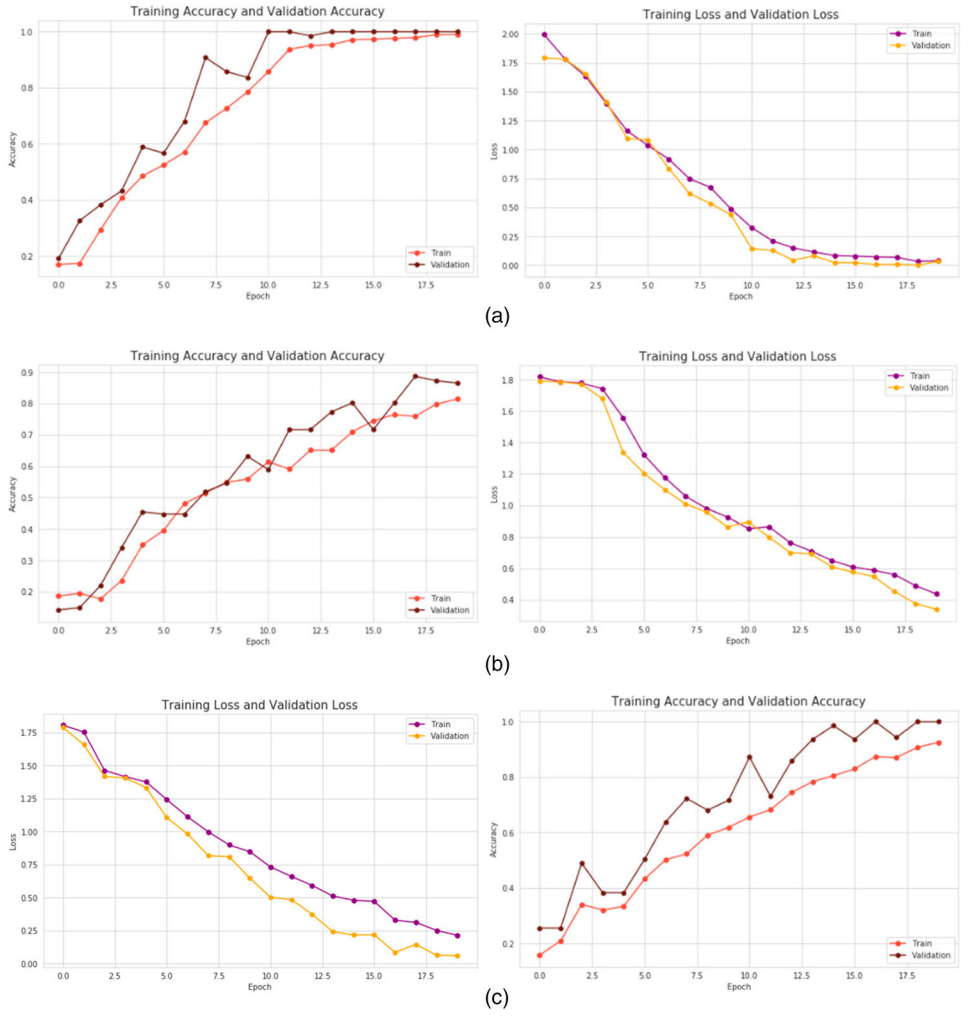

Figure 10. The training and validation accuracy and the loss values of Model 1 for the inputs of (a) 256×256, (b) 28×28, and (c) 32×32.

depicts some example objects. We set the head and neck angles of the humanoid robot to 0, -0.95 radian. We take the images with the resolution of 640×360 pixels from the camera of the robot using the ROS environment in the robot. We localized the objects at 20 different locations and views and collected 720 images. We removed the clutter regions not within the grasping area of the robot. Training set and validation set were prepared by using 576 images and 144 images, respectively. We re-sized the images to 256×256. We trained our models 1–4 for 20 iterations by using the Adam method and cross entropy loss function. Moreover, we reconstructed our models for 28×28 and 32×32 and trained the new models also similarly.

We evaluated our 12 models on the datasets we obtained by using the humanoid robot. Moreover, we loaded the ResNet-20 model and the VGG-16 model by using transfer learning and we later trained the models. Table 7 shows the training and

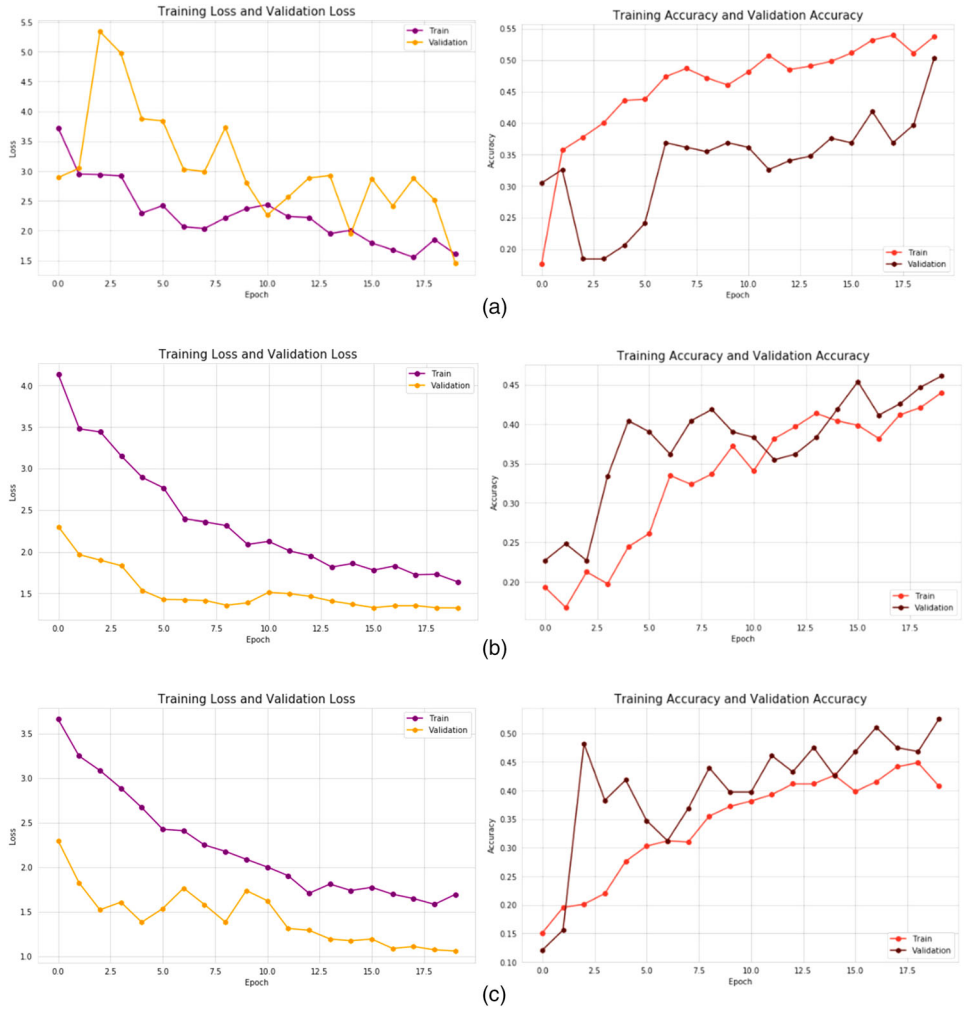

Figure 11. The training and validation accuracy and the loss values of Model 2 for the inputs of (a) 256×256, (b) 28×28, and (c) 32×32.

validation accuracies, the training and validation losses, the parameter numbers, and the training times of all models. On the other hand, Figures 10–13 show the accuracy and loss values with respect to epochs. As can be seen from the figures, the accuracy increases with respect to the epoch. The validation accuracy is usually smaller than training accuracy as the epoch increases. The training loss is smaller than validation loss as the epoch increases. There is no overfitting. Table 7 and Figures 10–13 show that Model 3 is the best one and it has the lowest training loss, training accuracy, validation loss, and validation accuracy of 0.0013, 1.0000, 0.0017, and 1.0000, respectively. Moreover, the parameter numbers in Table 7 present that the proposed models have much less complexity than those of VGG-16 and ResNet-20. As can be seen from Table 7, small sizes of the input showed a lower parameter number and less training time than using one of the original versions. For example, the parameter number of

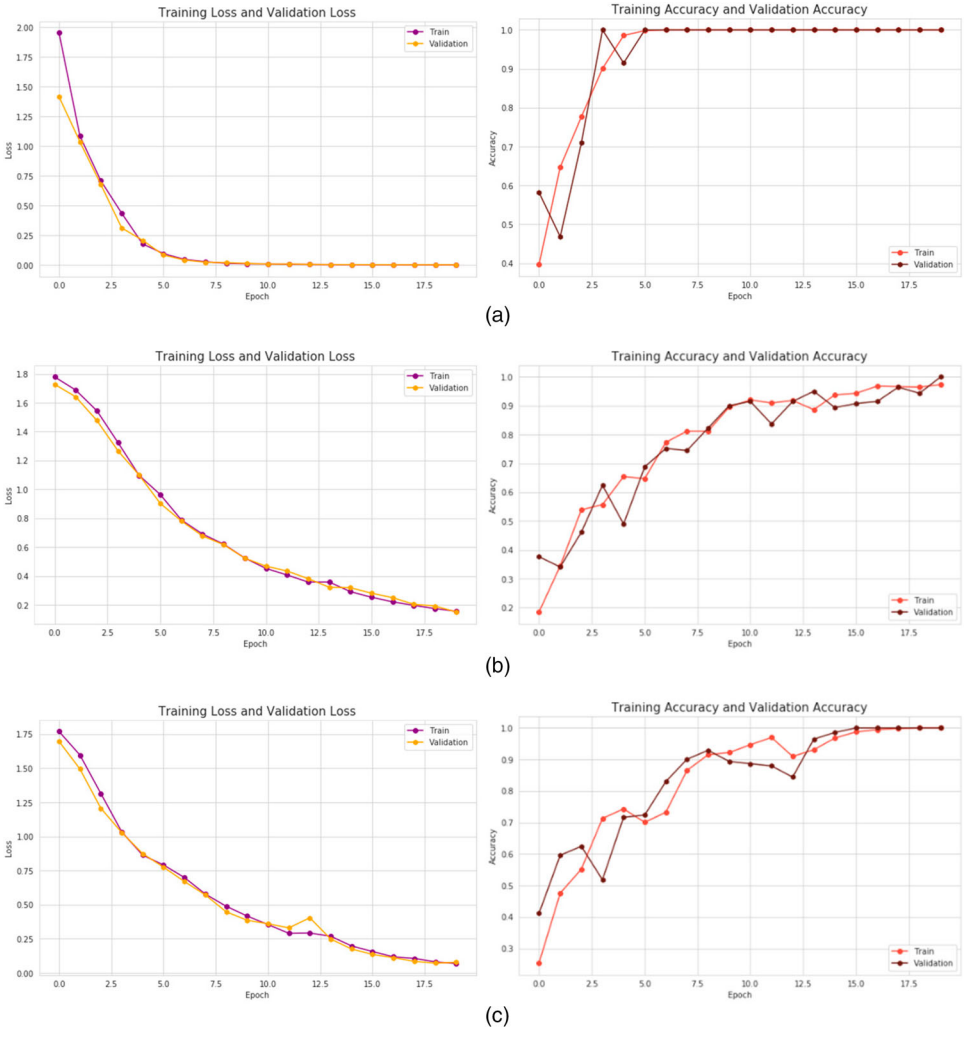

Figure 12. The training and validation accuracy and the loss values of Model 3 for the inputs of (a) 256×256, (b) 28×28, and (c) 32×32.

Model 3 was reduced from 743,142 to –9,894, which is about 75 times less than that of the original one. Hence, the small input size is important for low training time and fast calculation due to low computational load in the testing stage. Figure 9 shows in blocks the training time of our models, VGG-16 model, and ResNet-20 model for the inputs of 256×256 size. As can be seen from the figure, Model 3 takes the smallest training time and the best one for real-time applications. Embedding our models into the robot in the ROS environment, the robots recognize all objects and later go ahead to the manipulation step. In addition, it should be given an attention to overfitting as in Figures 10–13 and the training process should be stopped when the validation loss is smaller than the training loss. We did the experiment with different numbers of epochs, such as 20, 40, and 60. We observed higher accuracy for MNIST in all models as the epoch was increased, but for CIFAR-10 and our dataset, we

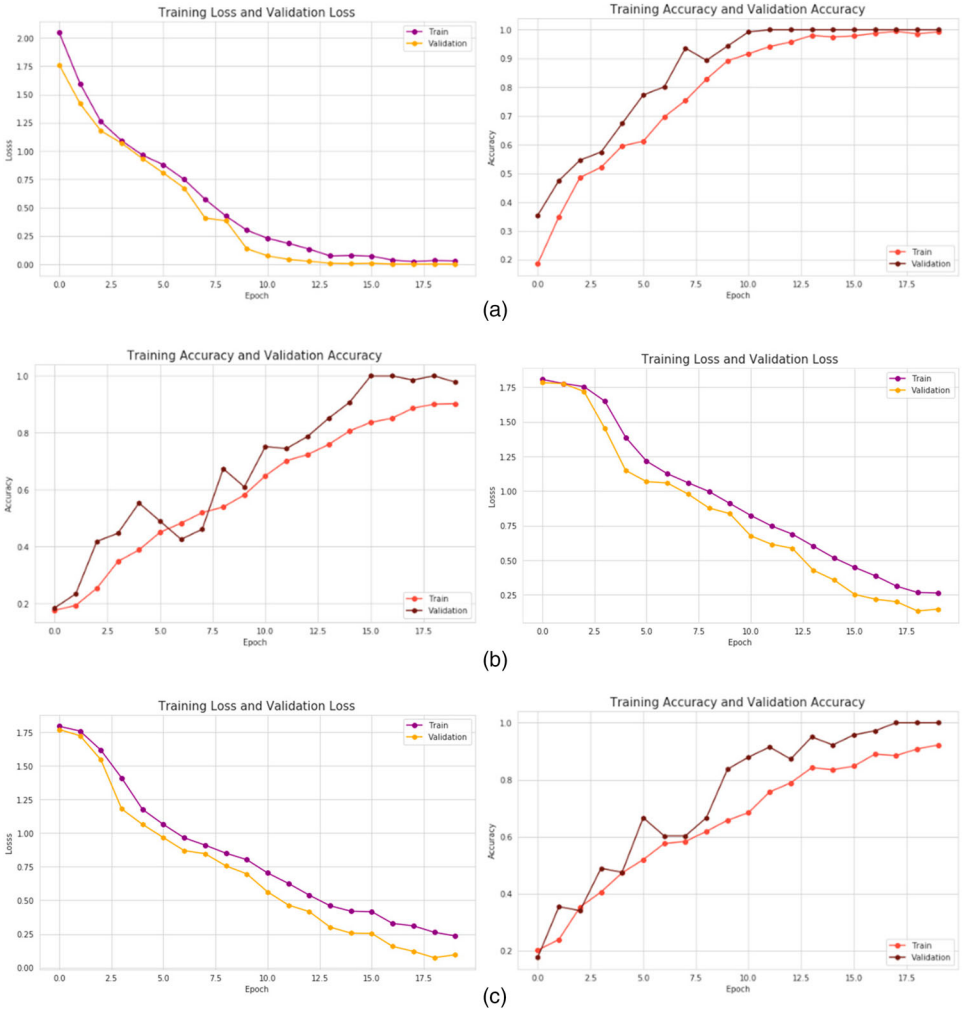

Figure 13. The training and validation accuracy and the loss values of Model 4 for the inputs of (a) 256×256, (b) 28×28, and (c) 32×32.

observed lower accuracy in some models. Hence, we used 20 epochs for all datasets. On the other hand, the performance of our models can be improved by using convolutions with different sizes as in the Inception network (Szegedy et al., 2016). Moreover, the performance can be increased with different optimization methods and learning rates.

4. Conclusion

In this paper, four efficient and fast CNN models were developed for recognizing the objects with humanoid robots. The main aim of the proposed models was to take on the powerful classification capability of CNN models and to provide fast and correct decisions in real time with humanoid robots.

The MNIST and CIFAR-10 datasets are first used for object recognition on the proposed four CNN models. The results on the MNIST dataset show that the proposed models provide the best accuracy of 99.40% even for small iteration numbers. Moreover, the models have smaller structures over those of the literature. The results on the CIFAR-10 dataset show that the proposed models provided high accuracy about 82.5% although they are the smallest parameter numbers and trained for only 20 epochs.

A comparative study was also conducted on the proposed four models and the derivations of different sizes. The experimental results showed that our model 3 has significantly surpassed the performance of the VGG-16 and Resnet20 models in terms of shorter training time and smaller parameter number; all versions of the model 3 also provided both training and validation recognition accuracy of 100%. In addition, our models can also be applied to real-time recognition applications. In future studies, the performance of proposed models will be improved using smaller and explainable structures at object recognition.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported by the Scientific and Technological Research Council of Turkey (Türkiye Bilimsel ve Teknolojik Araştirma Kurumu, TUBITAK) [grant number 117E589]. In addition, GTX Titan X Pascal GPU in this research was donated by the NVIDIA Corporation.

Notes on contributors

Simge Nur Aslan (Student Member, IEEE) received his BS degree from the Mechatronics Engineering Department at the University of Firat of Turkey in 2019. He is currently pursuing a master degree in the same department. His research interests include humanoid robots and deep learning.

Ayşegül Uçar (Senior Member, IEEE) received her BS degree, MS degree, and PhD degree from the Electrical and Electronics Engineering Department at the University of Firat of Turkey in 1998, 2000, and 2006, respectively. In 2013, she was a visiting professor at Louisiana State University in the USA. She has been a professor in the Department of Mechatronics Engineering since 2020. She has more than 21 years of experience in autonomous technologies and artificial intelligence, its engineering applications, robotic vision, teaching and research. Ucar is active in several professional bodies; she is an associate editor of IEEE Access and Turkish Journal Electrical Engineering and Computer Sciences and a member of European Artificial Intelligence Alliance Committee.

Cüneyt Güzeliş received the BSc, MSc, and PhD degrees in electrical engineering from Istanbul Technical University, Istanbul, Turkey, in 1981, 1984, and 1988, respectively. He was with Istanbul Technical University from 1982 to 2000 where he became a full professor. He worked between 1989 and 1991 in the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, California, as a visiting researcher and lecturer. He was with the Department of Electrical and Electronics Engineering from 2000 to 2011 at Dokuz Eylül University, Izmir, Turkey. He was with Izmir University of Economics, Faculty of Engineering and Computer Sciences, Department of Electrical and Electronics Engineering from 2011 to 2015. He is currently working in Yasar University, Faculty of Engineering, Department of Electrical and Electronics Engineering. His research interests include artificial neural networks, biomedical signal and image processing, nonlinear circuits-systems, and control, and educational systems.

ORCID

Simge Nur Aslan http://orcid.org/0000-0002-2738-7722
Ayşegül Uçar http://orcid.org/0000-0002-5253-3779
Cüneyt Güzeliş http://orcid.org/0000-0001-5416-368X

References

Alpaydin, E. (2016). *Machine learning: The new AI*. MIT press.

Ambrose, R., Askew, S., Bluethmann, W., & Diftler, M. (2001). Humanoids designed to do work. *Proceedings of the IEEE 2001 International Conference on Humanoid Robots, Robotics and Automation Society*. Tokyo, Japan.

Andtfolk, M., Nyholm, L., Eide, H., & Fagerström, L. (2021). Humanoid robots in the care of older persons: A scoping review. *Assistive Technology*, 1–9. https://doi.org/10.1080/10400435.2021.1880493

Angelopoulos, G., Baras, N., & Dasygenis, M. (2021). Secure autonomous cloud brained humanoid robot assisting rescuers in hazardous environments. *Electronics*, *10*(2), 124. https://doi.org/10.3390/electronics10020124

Aslan, S. N., Özalp, R., Uçar, A., & Güzeliş, C. (2021). New CNN and hybrid CNN-LSTM models for learning object manipulation of humanoid robots from demonstration. *Cluster Computing*, 1–16. https://doi.org/10.1007/s10586-021-03348-7

Aslan, S. N., Uçar, A., & Güzeliş, C. (2020). Fast object recognition for humanoid robots by using deep learning models with small structure. In *2020 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)* (pp. 1–7). IEEE. https://doi.org/10.1109/INISTA49547.2020.9194644

Ayinde, B. O., & Zurada, J. M. (2018). Building efficient convnets using redundant feature pruning. ArXiv Preprint ArXiv:1802.07653.

Bhuvaneswari, R., & Subban, R. (2018). Novel object detection and recognition system based on points of interest selection and SVM classification. *Cognitive Systems Research*, *52*, 985–994. https://doi.org/10.1016/j.cogsys.2018.09.022

Chang, O. (2010). Evolving cooperative neural agents for controlling vision guided mobile robots . In *2010 IEEE 9th International Conference on Cyberntic Intelligent Systems* (pp. 1–6). IEEE.

Chen, W., Wilson, J., Tyree, S., Weinberger, K., & Chen, Y. (2015). Compressing neural networks with the hashing trick. In *International Conference on Machine Learning* (pp. 2285–2294). PMLR.

Chohra, A., & Madani, K. (2018). Biological regulation and psychological mechanisms models of adaptive decision-making behaviors: Drives, emotions, and personality. In *Transactions on computational collective intelligence XXIX* (pp. 69–83). Springer. https://doi.org/10.1007/978-3-319-90287-6_4

Chollet, F. (2017). *Deep learning with python*. Simon and Schuster.

Dannecker, A., & Hertig, D. (2021). Facial recognition and pathfinding on the humanoid robot pepper as a starting point for social interaction. In *New trends in business information systems and technology* (pp. 147–160). Springer. https://doi.org/10.1007/978-3-030-48332-6_10

Denil, M., Shakibi, B., Dinh, L., Ranzato, M., & De Freitas, N. (2013). Predicting parameters in deep learning. ArXiv Preprint ArXiv:1306.0543.

Fitzpatrick, P., & Metta, G. (2003). Grounding vision through experimental manipulation. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, *361*(1811), 2165–2185. https://doi.org/10.1098/rsta.2003.1251

Garcia-Haro, J. M., Oña, E. D., Hernandez-Vicen, J., Martinez, S., & Balaguer, C. (2021). Service robots in catering applications: A review and future challenges. *Electronics*, *10*(1), 47. https://doi.org/10.3390/electronics10010047

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1440–1448). IEEE. https://doi.org/10.1109/ICCV.2015.169

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 580–587). IEEE. https://doi.org/10.1109/CVPR.2014.81

Gong, Y., Liu, L., Yang, M., & Bourdev, L. (2014). Compressing deep convolutional networks using vector quantization. ArXiv Preprint ArXiv:1412.6115.

Gowda, S. N., & Yuan, C. (2018). Colornet: Investigating the importance of color spaces for image classification. In *Asian Conference on Computer Vision* (pp. 581–596). Springer. https://doi.org/10.1007/978-3-030-20870-7_36

Han, S., Mao, H., & Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. ArXiv Preprint ArXiv:1510.00149.

Haykin, S. S. (2009). *Neural networks and learning machines*. Prentice Hall.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770–778). IEEE. https://doi.org/10.1109/CVPR.2016.90

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4700–4708). IEEE. https://doi.org/10.1109/CVPR.2017.243

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). Squeezenet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. ArXiv Preprint ArXiv:1602.07360.

Idelbayev, Y., & Carreira-Perpinán, M. A. (2021). Neural network compression via additive combination of reshaped, low-rank matrices. In *2021 Data Compression Conference (DCC'21)* (pp. 243–252). IEEE. https://doi.org/10.1109/DCC50243.2021.00032

Jaderberg, M., Vedaldi, A., & Zisserman, A. (2014). Speeding up convolutional neural networks with low rank expansions. ArXiv Preprint ArXiv:1405.3866.

Jha, D., Yazidi, A., Riegler, M. A., Johansen, D., Johansen, H. D., & Halvorsen, P. (2020). Lightlayers: Parameter efficient dense and convolutional layers for image classification. In *International Conference on Parallel and Distributed Computing: Applications and Technologies* (pp. 285–296). Springer.

Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images (Technical Report). University of Toronto.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, *25*, 1097–1105. https://doi.org/10.1145/3065386

Ku, L. Y., Learned-Miller, E., & Grupen, R. (2017). An aspect representation for object manipulation based on convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 794–800). IEEE.

Le, T. D., Huynh, D. T., & Pham, H. V. (2018). Efficient human-robot interaction using deep learning with mask R-CNN: Detection, recognition, tracking and segmentation. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)* (pp. 162–167). IEEE.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. https://doi.org/10.1109/5.726791

Lee, A. (2015). *Comparing deep neural networks and traditional vision algorithms in mobile robotics*. Swarthmore University.

Lee, S., Kim, H., Jeong, B., & Yoon, J. (2021). A training method for low rank convolutional neural networks based on alternating tensor compose-decompose method. *Applied Sciences*, *11*(2), 643. https://doi.org/10.3390/app11020643

Levine, S., Finn, C., Darrell, T., & Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, *17*(1), 1334–1373. arXiv preprint arXiv:1504.00702, 2015.

Martinez-Martin, E., & Del Pobil, A. P. (2017). Object detection and recognition for assistive robots: Experimentation and implementation. *IEEE Robotics & Automation Magazine*, *24*(3), 123–138. https://doi.org/10.1109/MRA.2016.2615329

Nenchev, D. N., Konno, A., & Tsujita, T. (2018). *Humanoid robots: Modeling and control*. Butterworth-Heinemann.

Oliver, G., Lanillos, P., & Cheng, G. (2021). An empirical study of active inference on a humanoid robot. *IEEE Transactions on Cognitive and Developmental Systems*. IEEE. https://doi.org/10.1109/TCDS.2021.3049907

Pandey, A. K., & Gelin, R. (2017). Humanoid robots in education: A short review. In A. Goswami & P. Vadakkepat (Eds.). In *Humanoid robotics: A reference* (pp. 1–16). Springer. https://doi.org/10.1007/978-94-007-7194-9_113-1

Rastegari, M., Ordonez, V., Redmon, J., & Farhadi, A. (2016). Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision* (pp. 525–542). Springer.

Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, faster , stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 7263–7271). IEEE. https://doi.org/10.1109/CVPR.2017.690

Reforgiato Recupero, D. (2021). Technology enhanced learning using humanoid robots. *Future Internet*, *13*(2), 32. https://doi.org/10.3390/fi13020032

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems*, *28*, 91–99. IEEE. https://doi.org/10.1109/TPAMI.2016.2577031

Robotis-Op3. (2020, May 5). http://emanual.robotis.com/docs/en/platform/op3/introduction/

Sakagami, Y., Watanabe, R., Aoyama, C., Matsunaga, S., Higaki, N., & Fujimura, K. (2002). The intelligent ASIMO: System overview and integration. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, *3*, 2478–2483. https://doi.org/10.1109/IRDS.2002.1041641

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. ArXiv Preprint ArXiv:1409.1556.

Srinivas, S., & Babu, R. V. (2015). Data-free parameter pruning for deep neural networks. ArXiv Preprint ArXiv:1507.06149.

Srinivas, S., Sarvadevabhatla, R. K., Mopuri, K. R., Prabhu, N., Kruthiventi, S. S., & Babu, R. V. (2016). A taxonomy of deep convolutional neural nets for computer vision. *Frontiers in Robotics and AI*, *2*, 36. https://doi.org/10.3389/frobt.2015.00036

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions . In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1–9). IEEE. https://doi.org/10.1109/CVPR.2015.7298594

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2818–2826). IEEE. https://doi.org/10.1109/CVPR.2016.308

Xie, L., Wang, S., Markham, A., & Trigoni, N. (2017). Towards monocular vision based obstacle avoidance through deep reinforcement learning. ArXiv Preprint ArXiv:1706.09829.

Yang, Z., Moczulski, M., Denil, M., De Freitas, N., Smola, A., Song, L., & Wang, Z. (2015). Deep fried convnets. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1476–1483). IEEE. https://doi.org/10.1109/ICCV.2015.173

Yavşan, E., & Uçar, A. (2016). Gesture imitation and recognition using kinect sensor and extreme learning machines. *Measurement*, *94*, 852–861. https://doi.org/10.1016/j.measurement.2016.09.026