Chapter

Convolutional Neural Network Demystified for a Comprehensive Learning with Industrial Application

Anand Raju and Shanthi Thirunavukkarasu

Abstract

In the recent past of time, numerous investigators have driven on and subsidized novelties to image classification methods. In this chapter, an introduction to image classification scheme and their types is offered. Image classification discovers its application in a variety of fields, to name a few, judgment of diseases, finding and identification of faults, classification of nutrition goods based on superiority, valuation of usual capitals and conservation pollution, education of land use and land cover from remote sensing satellite images, character identification and detection in optical character reader, face recognition, object detection, and so on. Automatic image classification schemes found on actual algorithms deliver high accuracy and exactness in recognizing object/features. Convolution neural network is a superior genre of neural network that requires minimal preprocessing. The ability of the convolutional neural network (CNN) to understand the visual content of the input image makes its suitable for recognizing minute variation between the classes. This power of the CNN makes it a good choice to address image classification problems with multi-classes. So, in this chapter, the entire flow of CNN's architecture with different industrial applications will be discussed.

Keywords: convolutional neural network, machine learning, deep learning, python, data prepossessing, pooling, layers, architectures

1. Introduction

The convolutional neural network also termed as ConvNet or CNN is a form of deep learning neural network. Most of the inner layers apply mathematical convolution operation to compute the feature maps for the next layer and hence the name convolutional neural network. The effortless construction of such network makes itself useful for a variety of real-time applications like image classification systems, image recommender systems, optical character recognition (OCR) systems, medical diagnosing systems, fault detection systems, and so on. The CNN architecture is highly inspired by a pattern of neuron connectivity in human visual cortex. The minimum intervention of human in feature selection and nominal preprocessing makes itself preferable for different applications. Like the traditional neural network, CNN consists of input layer, hidden layers, and an output layer [1].

The hidden layers are formed by combinations of convolution layers, ReLU layers, and pooling layers. The CNN differs from regular neural network in several ways. The input is a three-dimensional data that usually includes the color channels of the input image. The parameter sharing and confined connectivity are unique features of CNN. Parameter sharing refers to the sharing of equal weights for all neurons in computation of a feature map. Confined connectivity enables the neurons to get connected to a specific subset of the input layer. This reduces the overfitting problem. These two features of CNN reduce the total number of learnable parameters, thereby decreasing the computation time [2, 3]. The following section explains the functions of each layer in detail.

1.1 Convolutional layer

The convolutional layer computes a simple dot product between the selected region of the image and set of kernel functions called filters. Generally, an image of dimension $M \times N \times C$ is given as input. $M \times N$ represents the length and width of the image, and C represents the number of color channels usually $C = 3$ (red, green, and blue color channels) [4]. The output of the convolutional layer is called feature map and its dimensions are given as $W \times Q \times K$. The values of W and Q can be determined using the equation given below. Several parameters like filter size (F), zero padding (ZP), stride (S), and number of filters (K) are used to compute the dimensions of the feature map.

$$W = \frac{(M - F + (2ZP))}{S} + 1 \tag{1}$$

$$W = \frac{(N - F + (2ZP))}{S} + 1 \tag{2}$$

The filter parameters are explained as below filter size (F) denotes the size of the filter or kernel used for convolution operation. Generally, kernel functions are odd-numbered square matrix, i.e., 3×3, 5×5, etc. [5]; an odd value is preferred so that the center of the kernel matrix can be fixed on the pixel on which it is operated. The figure shows the convolution of 5×5 input image (blue-colored matrix) with 3×3 kernel function (green-colored matrix) and a 3×3 feature map output (**Figure 1**).

1.2 Zero padding

Zero padding refers to the number of zeroes that are added along the brim of the input matrix. The dimensions of output matrix of the convolutional layer will be

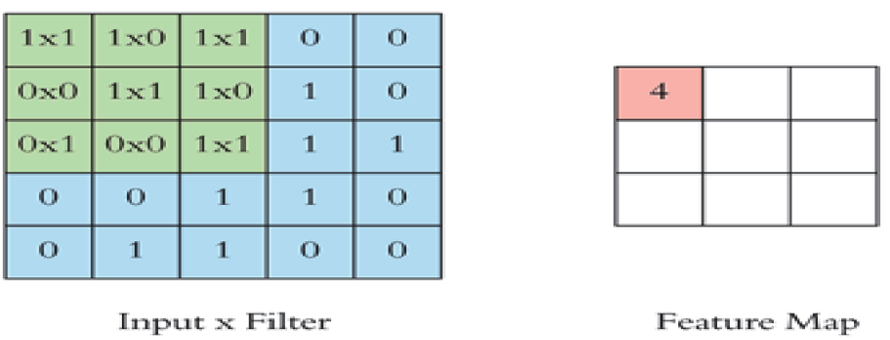

Figure 1.
Convolution of 3 × 3 kernel function over a 5 × 5 input.

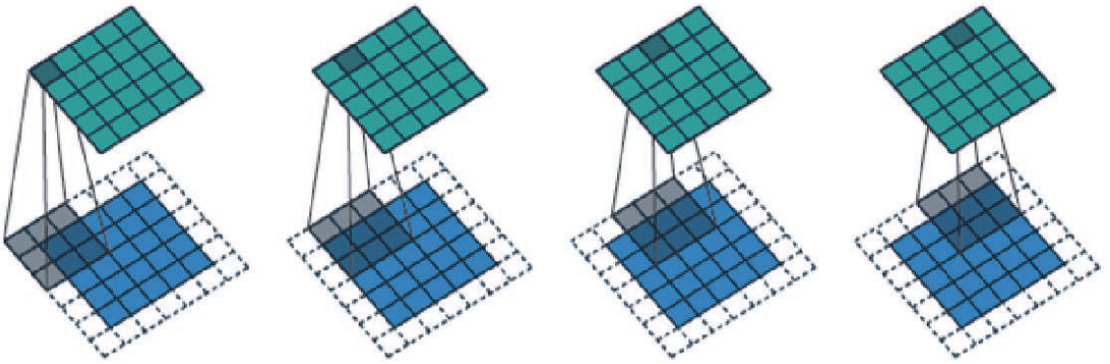

Figure 2.
Demonstration of zero padding to maintain the output size same as that of the input.

reduced because of mathematical operation performed in that layer. Appending zeroes to the input matrix will prevent the shrinkage of size of feature maps produced at the output. Maintaining the size of feature maps can be achieved by assigning zero padding as depicted in the figure, and hence, it becomes essential in networks with more number of layers (**Figure 2**).

1.3 Stride (S)

A filter or the kernel matrix has to be translated throughout the input matrix vertically from top to bottom and also horizontally from left to right, covering all the elements in the input matrix. Stride controls the movement of the translation. It represents the number of steps taken by the kernel matrix during its translational movement. The figure illustrates the movement of the filter when a 3×3 kernel function is applied over a 7×7 input with stride S = 2 output and results in a size of 3×3 (**Figure 3**).

A demonstration of the function of convolutional layer is displayed in the figure. The three-dimensional input with the volume $5 \times 5 \times 3$ (M = N = 5, C = 3) when padded with zero outer border turns into a volume of $7 \times 7 \times 3$ and is shown in blue color. Two filters with the volume $3 \times 3 \times 3$ (F = 3, K = 2) are shown in red color. The three layers are shown one below the other. The filter is convolved with the input matrix with zeros in the outer border (zero padding ZP = 1) and stride S = 2. With these parameters, the dimension of the feature map is computed using the equation:

$$W = Q = \frac{(M - F + (2ZP))}{S} + 1 = \frac{(5 - 3 + (21))}{2} + 1 = 3 \qquad (3)$$

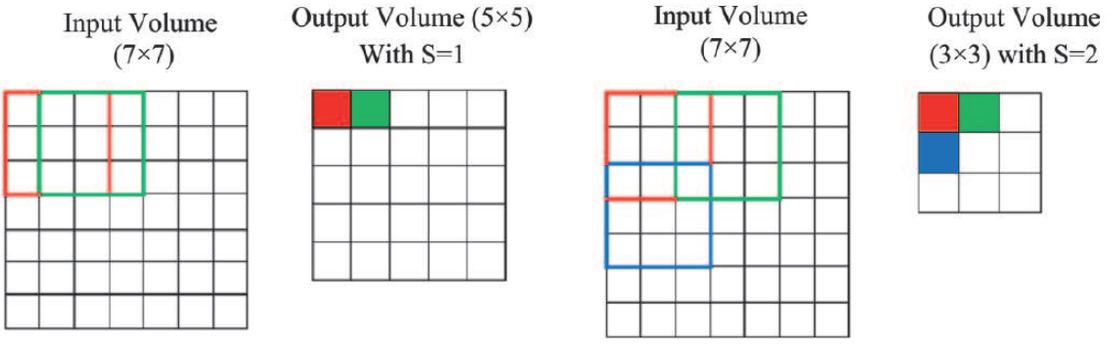

Figure 3.
Application of 3 × 3 kernel function over a 7 × 7 input with stride S = 1 and S = 2.

The dimension of the feature map is given as W × Q × K = 3 × 3 × 2, and it is shown in green color. The kernel matrix contains different weights in order to differentiate features of higher importance with other features. Filter weights are the values of the kernel matrix. Apart from assigning weights, bias values are also provided to mention the probability of occurrence of a particular pattern. The weights and biases refer to the parameters of the network. The parameters are learned by the network during the training phase, and all other parameters are termed as hyper parameters (**Figure 4**). The values in the output matrix are calculated using the equation:

$$\text{Output} = (\text{Inputxfilter}) + \text{bias} \tag{4}$$

A unique feature of CNNs is that same filter is shared with many neurons. Using the same bias and weight vectors along the region reduces the memory requirement to a great extent. The convolution explained in this section is basically a 2D convolution. It is called 2D convolution because the kernel is moved in only two directions, i.e., along the height and width of the input layer. 3D convolution is also possible if a 3D kernel is applied and moved along all three dimensions, i.e., along the height, width, and depth of the input volume. Apart from the regular convolution, there are few other types of convolutions as discussed below.

1.4 Types of convolution

1.4.1 Transposed convolution

This type of convolution is preferred for deconvolution operation and generally introduces checkerboard effects in the output image. This type of convolution generally increases the size of output image as it does up sampling process [4].

1.4.2 Dilated convolution

In this type of convolution (d-1), spaces are inserted in between the kernels before applying convolution. If d = 1, it resembles regular convolution.

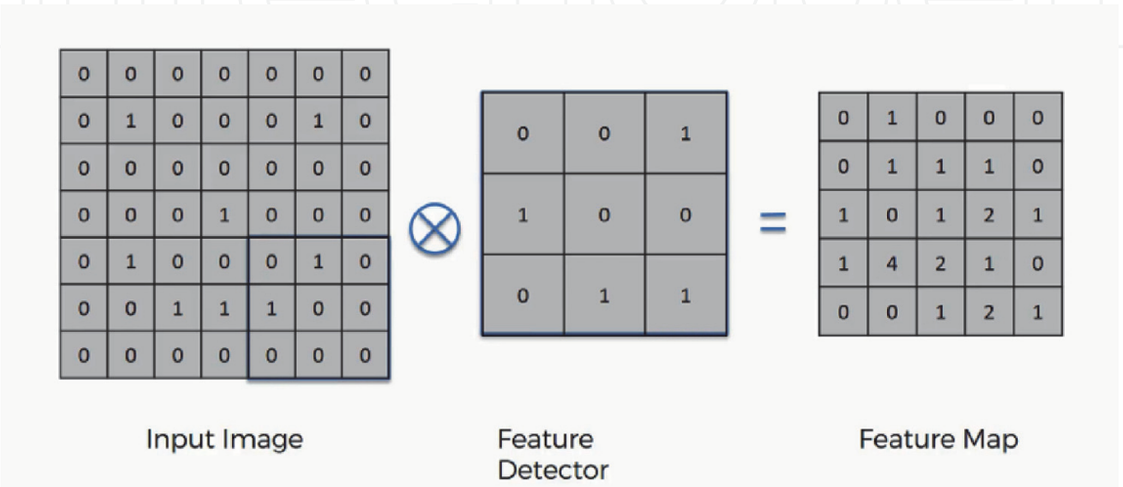

Figure 4.
Display of convolution function of convolutional layer.

1.4.3 Separable convolution

In this type of convolution, the kernels are separated into two smaller kernels. A 2D kernel will be broken into row matrix and column matrix. This reduces the number of computations. Despite reduction in the computational cost, this type of convolution is not preferred generally in many deep learning algorithms as it provides only suboptimal results.

1.4.4 Flattened convolution

In this type of convolution, a 1D kernel is used to traverse over the 3D input volume to produce the output feature maps. This greatly reduces the number of learnable parameters and results in less computational cost.

1.5 Pooling layer

Pooling layer is an important layer in CNN. Pooling is used for down sampling. It is mainly involved in reducing the spatial dimension of the feature maps. This considerably reduces the number of parameters required for training the network. By reducing the trainable parameters, it minimizes the computations required. Pooling layer becomes essential in networks with a greater number of layers. Applying a 2×2 pooling filter reduces the size of the feature map to half of its original size. The figure demonstrates the effect of 2×2 pooling filter applied on $224 \times 224 \times 64$ feature map that reduces the size of the o/p feature map to $112 \times 112 \times 64$. However the depth of the feature map remains unaltered (**Figure 5**) [6].

There are two types of pooling filters: maximum pooling (max pooling) and average pooling (avg pooling). A 2×2 max pooling filter selects a 2×2 region from the input image and passes the maximum value to the next stage. Average pooling filter passes the average value of the selected region to the next stage (**Figure 6**).

1.6 Activation functions

The input for the any CNN network is an image, and the output is the class score. This establishes a nonlinear relation between the input and output data. Activation functions are the nonlinear transformation inserted in between the layers. Firing of neurons depends on the results of the activation functions. The important characteristic of an activation function is that its derivative should exist (**Figure 7**).

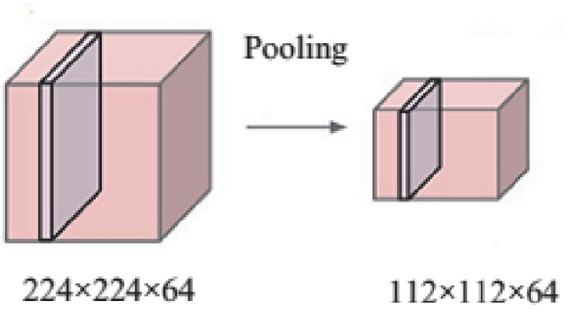

Figure 5.
Effect of pooling layer with 2 × 2 filter.

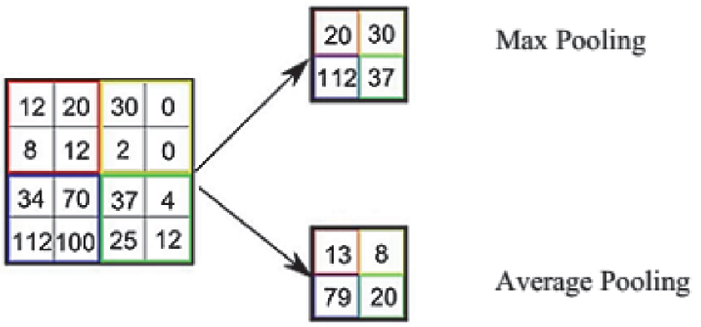

Figure 6.
Application of 2 × 2 max pooling filter and average pooling filter on a 4 × 4 feature map.

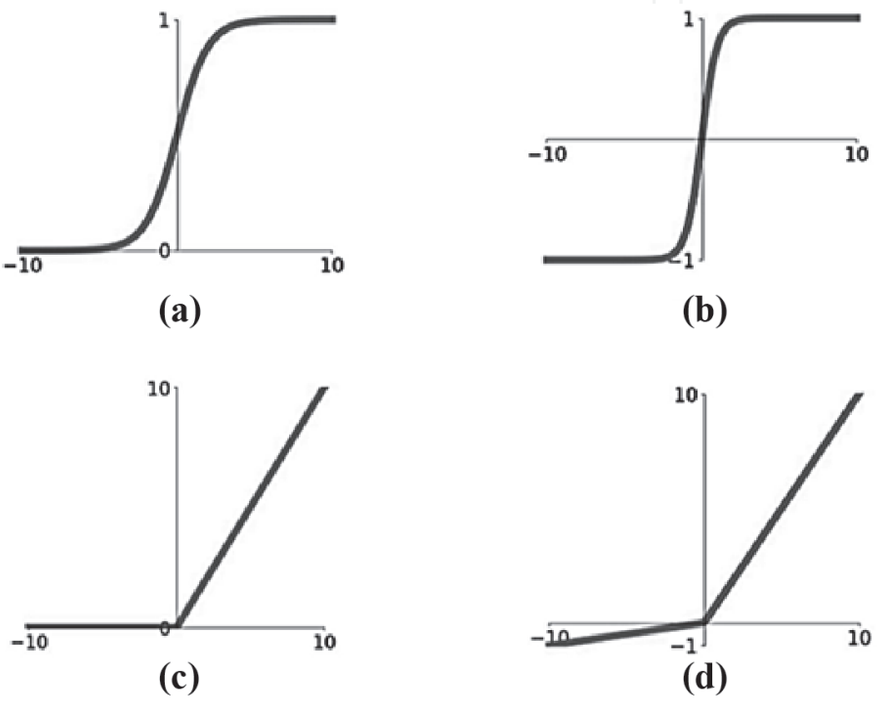

Figure 7.
Activation functions: (a) sigmoid function, (b) tanh function, (c) ReLU function, and (d) leaky ReLU function.

The activation functions are shown in the figure in which the y-axis represents the function f(a) and x-axis denotes the value a. The following functions are the most prevalent activation functions used in neural network.

1.7 Sigmoid function

Sigmoid or logistic function is a function that ranges between 0 and 1. It is computed as given in the equation. Slow convergence of the function makes it less popular.

$$f(a) = \frac{1}{1 + e^{-a}} \tag{5}$$

1.8 Hyperbolic tangent function

The hyperbolic tangent function ranges between −1 and 1. It overcomes the disadvantage of sigmoid function of having the zero centered function. It makes the optimization easier and it is preferred over sigmoid function.

$$f(a) = \tanh(a) = (e^{a} - e^{-a})/(e^{a} + e^{-a}) \tag{6}$$

1.9 ReLU: rectified linear units

The function results in values in the range [0, ∞). It is the most commonly used activation function. The negative values of the input matrix are removed completely. A simple thresholding can be used for implementing this activation. This function avoids the simultaneous activation of all neurons, thereby reducing the number of computations. Convergence of this function is also faster than the sigmoid and tangent functions. The disadvantages of this function are that it cannot update all the weights during back propagation when the gradient is zero and it results in more number of dead neurons when used with high learning rate.

$$f(a) = \begin{cases} 0 \text{ for } a < 0 \\ a \text{ for } a \geq 0 \end{cases} \tag{7}$$

1.10 Leaky ReLU: rectified linear units

The function results in values in the range (−∞,∞). Instead of removing negative values, it provides a negative slope for $a < 0$. The values of slope can be very small in the range $\delta = 0.01$, 0.1, etc. This eliminates the problem of dying neurons as in the case of ReLU.

$$f(a) = \begin{cases} \delta a \text{ for } a < 0 \\ a \text{ for } a \geq 0 \end{cases} \tag{8}$$

1.11 Normalization layer

The stability of the network can be improved by introducing batch normalization layers in between the regular layers. In this layer the mean and standard deviation of the values in previous layer are computed. Then the mean value is subtracted from each of the input value and divided by the standard deviation. The batch normalization gives a choice to network from getting rid of dropouts and enables the network to have a higher learning rate. It introduces two more learnable parameters p1 and p2 to the network. The output y after the batch normalization of a small batch of the input data $x_i = x_1, x_2, x_3, .x_m$ is given by the equation:

$$y = BN_{p_1 p_2}(x_i) = p_1 \hat{x}_i + p_2 \tag{9}$$

where batch mean $\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$ and variance is given as $\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)$.

1.12 Fully connected layer

The name fully connected layer is coined since each neuron in the layer is connected to every neuron in the previous layer as displayed in **Figure 8**. Generally, two to three fully connected layers are appended as the final layers in every CNN. Activations of fully connected layer follow affine transformation, which involves matrix multiplication and addition of bias values. This fully connected layer is equivalent to the classifiers, whereas the previous layers are responsible for feature extraction [7].

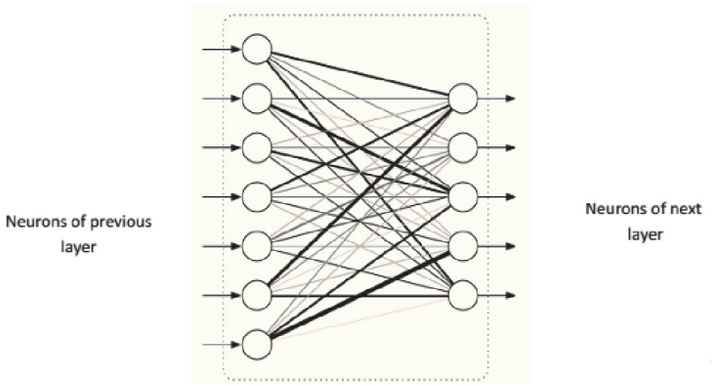

Figure 8.
Fully connected layer.

1.13 Optimization methods

A CNN classifier is involved in calculating the output class score (Y) from the input image with a set of predictors (X). Optimization algorithms aid the network to curtail the error function by updating the parameters in the right direction. Effective training of the network is greatly influenced by the choice of correct network parameters. Optimum values of such parameters can be achieved by adopting suitable optimization algorithms. Few commonly used optimization algorithms are described below.

1.13.1 Stochastic gradient descent (SGD)

A matrix formed with first-order derivatives is called Jacobian matrix, and it is used to represent a gradient function. SGD is an iterative method that is used with batch processing. The network weights are updated with the average value of the gradients of a batch. If L(w) is the loss function or error function, the objective of the SGD is to identify the parameter Ï‰ that minimizes L(w). The SGD algorithm performs the iterations using equation to compute the parameter value.

$$w^{k+1} = w^k - \eta\delta L(w) = w^k - \eta \sum_{i=1}^{n} L(w) \tag{10}$$

where η is the learning rate

1.13.2 SGD with momentum

In this algorithm the update for the parameter Ï‰ is computed as a linear combination of the previous update and the gradient. This algorithm prevents oscillation by traveling in the same direction.

$$\delta w = \alpha\delta w - \eta\delta L_i(w) \tag{11}$$

$$w^{k+1} = w^k + \delta w \tag{12}$$

$$w^{k+1} = w^k + \delta w = \alpha\delta w - \eta\delta L_i(w) \tag{13}$$

where η is the learning rate and α is the momentum term usually 0.9 or a similar value.

1.13.3 Nesterov's accelerated gradient (NAG)

NAG utilizes the momentum to predict the future values of the parameter. Thus, the momentum $w - \alpha v_{t-1}$ is used to estimate the next possible position of the parameter. This enables the algorithm to converge more rapidly.

1.13.4 AdaGrad

AdaGrad algorithm is an adaptive gradient algorithm in which a flexible learning rate is adopted. Larger learning rate is used for features that occur very rarely, and smaller learning rate is used for frequently occurring features. The learning of network stops at a point when the learning rate shrinks to zero. This is a drawback for this method.

1.13.5 RMSProp

RMSProp also adopts the idea of using a variable learning rate. The learning rate is decided based on the signs of the previous two gradients. If the previous two gradients are of the same sign, it increases the step size in the order of 1.2 times the previous learning rate. Otherwise the step size is decreased in the order of 0.5 times the previous learning rate. The learning rate is always limited between one millionth and 50.

1.13.6 Adam optimizer

Adam optimizer is the name derived from adaptive momentum estimation. Adam optimizer attracts with its advantages like simple implementation procedure, minimum memory requirements, and efficient computation. The algorithm is best suited for very big data size with less number of tuning and problems with sparse gradients. Adam optimizer retains the benefits of both RMSProp and AdaGrad optimizers. Apart from the first moments of gradients, the mean of the second moments of gradients were also used in Adam optimizer.

1.14 Softmax layer

The final layer of CNN architecture is mostly the softmax layer. Normalized exponential function is called the softmax function. This layer converts an N number of input vectors into N probabilities. This layer is essential as it normalizes the input vector of any value (negative values, positive values greater than one, etc.). The number of nodes in the softmax layer is equal to the number of output classes. The softmax function for the input vector $z_i = z_1, z_2, z_3, z_N$ is converted in to probabilities $P(z)_i$ using the given equation:

$$P(z_i) = \frac{e^{zi}}{\sum_{j=1}^{N} e^{zi}} \tag{14}$$

2. LeNET architecture and its applications

Optical character recognition is one of the major research problems in real-time applications, and it is used to recognize all the characters in an image. As English is a universal language, character recognition in English is a challenging task. Deep

Layer	Feature map	Size	Kernel size	Stride	Activation
Input layer	1	32 × 32	—	—	—
Convolution layer 1	6	28 × 28	5 × 5	1	tanh
Maximum pooling layer	6	14 × 14	2 × 2	2	tanh
Convolution layer 2	16	10 × 10	5 × 5	1	tanh
Maximum pooling layer 2	16	5 × 5	2 × 2	1	tanh
Convolution layer 3	120	1 × 1	5 × 5	1	tanh
Fully connected layer	—	84	—	—	tanh
Output layer (FC 2)	—	10	—	—	Softmax

Table 1.
Summary of LeNET architecture.

learning approach is one of the solutions for the recognition of optical characters. The aim of this research work is to perform character recognition using convolutional neural network with LeNET architecture. Optical character images have been binned in each class (10 classes) to form 2495 samples of training images and 1069 test images. Each of the images fed for training/testing has a size of 32 × 32 such that the convolution layer is enabled with a filter size of 5 × 5. Thus the output of the convolution layer is 28 × 28 along with six feature maps [24, 25]. The second (maximum pooling) layer gets a maximum of 2 × 2 value from the 28 × 28 image. The maximum pooling layer [21], [22] thus gives an output size of 14 × 14 with six feature maps. The convolution layer forms the third layer which performs the convolution operation of input with filter. The output of this layer is 10 × 10 with 16 feature maps for a filter size of 5 × 5. The last layer in the network is a convolution layer with filter size of 5 × 5. The output of the last layer is fed to fully connected layers 1 and 2. The fully connected layer 2 forms the output layer with a total of 10 outputs (**Table 1**).

3. CNN applications

3.1 AlexNET architecture and its applications

Image classification is one of the major research problems in real-time applications, and it is used to recognize all the objects in an image. Deep learning approach is one of the solutions for the recognition and identification of different images. The aim of this research work is to perform character recognition using convolutional neural network with AlexNET architecture. For analysis of AlexNET architecture with the created database, an input image is fixed with a size of [227 × 227]. The first layer in AlexNET is the convolution layer 1 with an input size of [227 × 227]. This layer convolves the input with a filter size of [11 × 11] and provides an output with 96 feature maps with a size of [55 × 55] (stride 4) [8].

Maximum pooling layer forms the second layer, which subsamples the output of the first layer with a pooling size of [3 × 3] and gives an output size of [55 × 55] (96 feature maps). The third layer convolves the output of the second layer with a filter size of [5 × 5]. The output of the third layer is once again pooled with a filter (size 27 × 27) to get a feature map vector of size 256. The maximum size of filter utilized in the next layer is [3 × 3] for an image input of [27 × 27], thereby providing an output of 256 feature maps. Another convolution layer forms the sixth layer which convolves with the fifth layer to produce a feature vector size of 256.

Layer	Feature map	Size	Kernel size	Stride	Activation
Input layer	1	227 × 227	—	—	—
Convolution layer 1	96	55 × 55	11 × 11	4	ReLU
Maximum pooling layer	96	27 × 27	3 × 3	2	ReLU
Convolution layer 2	256	27 × 27	5 × 5	1	ReLU
Maximum pooling layer 2	256	13 × 13	3 × 3	2	ReLU
Convolution layer 3	384	13 × 13	3 × 3	1	ReLU
Convolution layer 4	384	13 × 13	3 × 3	1	ReLU
Convolution layer 5	256	13 × 13	3 × 3	1	ReLU
Max pooling	256	6 × 6	3 × 3	2	ReLU
Fully connected layer 1	—	9216	—	—	ReLU
Fully connected layer 2	—	4096	—	—	ReLU
Fully connected layer 3	—	4096	—	—	ReLU
Output layer (FC -4)	—	10	—	—	Softmax

Table 2.
AlexNET architecture summary for palm leaf characters.

Table 2 gives a summary of the AlexNET architecture that has been created during this experiment.

3.2 FaceNet architecture and its applications

Face detection is the process of detecting a face in an image or a video. Face recognition is the process of detecting face in an image and then using algorithms to identify who the face belongs to. Face recognition is thus a form of person identification. Initially, features are extracted from the image for training the machine learning classifier to identify faces in the image. Not only are these systems not subjective, but also they are also automatic—no hand labeling of facial features is required. Since for face recognition, we need to detect a face from the image or video, we can think of face recognition as a two-phase stage: Stage 1, detecting the presence of faces in the image or video stream using methods such as Haar cascades, HOG+Linear SVM, deep learning, or any other algorithm that can localize faces, and Stage 2, taking each of these faces detected during the localization phase and

Type	Patch size/stride	Output size
Convolution	34 × 34/2	112 × 112 × 64
Max pool	3 × 3/2	56 × 56 × 64
Convolution	3 × 3/1	56 × 56 × 192
Max pool	3 × 3/2	28 × 28 × 192
Inception(3a)	—	28 × 28 × 256
Inception(3b)	—	28 × 28 × 480
Max pool	3 × 3/2	14 × 14 × 480
Inception(4a)	—	14 × 14 × 512
Inception(4b)	—	14 × 14 × 512
Inception(4c)	—	14 × 14 × 512

Type	Patch size/stride	Output size
Inception(4d)	—	14 × 14 × 528
Inception(4e)	—	14 × 14 × 832
Max pool	3 × 3/2	7 × 7 × 832
Inception(5a)	—	7 × 7 × 832
Inception(5b)	—	7 × 7 × 1024
Avg pool	7 × 7/1	1 × 1 × 1024
Dropout (40%)	—	1 × 1 × 1024
Linear	—	1 × 1 × 7
Softmax	—	1 × 1 × 7

Table 3.
FaceNet architecture.

learning whom the face belongs to; this is where you assign a name to a face. Face detection is the techniques to finding all the faces in an image or videos [9]. Face recognition is the next step after face detection. In face recognition you identify which face belongs to which person using an existing (Pre-Trained Image) image in the repository. Face analysis is a process of extracting facial features like age, complexion, etc., from the image after recognizing a face in it. Generally, OpenCV provides three different methods for face recognitions like eigenfaces [8], local binary pattern histograms, and fisherfaces. But, nowadays, deep learning using FaceNet is a very popular algorithm used in many applications, which is shown in **Table 3**.

3.3 Object detection and its applications

The environmental problems and its treatment go back to the fifteenth century. A wide variety of road types such as intersections and highways pose a real challenge to the computer vision algorithms. Hence, there is a need of efficient algorithm to detect the accident on road and also evaluate the severity of the incident. In this paper, an accident detection approach using ResNet architecture has been presented with specific focus on road accident. The convolutional neural network used in this paper has utilized around 50 layers, viz., convolution layer, pooling layer, activation layer, fully connected layer, and softmax classifier and inspection layer. The paper has also created a database of accident video set by utilizing the video images of accident. The recognition of ResNet 50 classifier has been found to be 98.1%. The prediction rate is found to be higher due to the large quantum of features extracted in each of the CNN layers. There is no space for waste. Our landfill sites are filling up fast; by 2010, almost all landfills in the UK will be full. Financial expenditure in the economy is reduced. Creating products from raw materials costs much more than if they were made from recycled products. Natural resources should be preserved for future generations. Recycling reduces the need for raw materials; it also uses less energy, therefore preserving natural resources for the future garbage sorter robot, a device which uses image processing combined with robotics to collect and sort the available garbage using highly sensitive sensors and a smartphone camera. This would be very useful in cleaning the environment and for effective recycling without wasting man power. Garbage is an important cause of many diseases. Recycling must be made more effective. Usage of man power must be considerably reduced which is shown in **Figure 9**. This being our

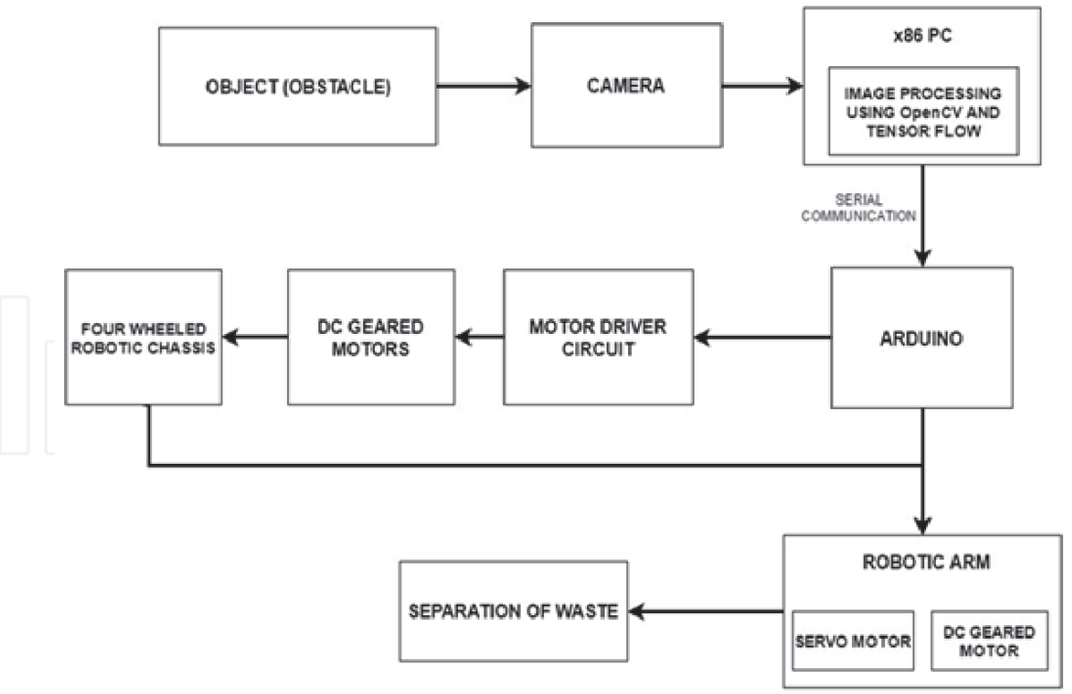

Figure 9.
Object detection system.

motivation, we could take some naturally available parameters that come out of the garbage and make a device that could read them and detects the changes in them. Thus by doing so, the negative effects due to waste could be reduced. The above gives us a clear picture of the hardware and software used accordingly. The components used along with their usage are explained below:

3.3.1 Camera

A webcam which is around 5 MP is used for capturing the images, and the captured images are fed to the ×86 PC for processing them.

3.3.2 ×86 PC

The required image processing software, i.e., OpenCV and TensorFlow, is installed and used for processing the captured images. Here the language used is Python. Similarly the Arduino IDE is used to program the Arduino board. Arduino board is used for the movement of wheels and robotic arm which is shown in **Figure 10**.

3.3.3 Robotic base

The robotic base consists of an acrylic sheet which has two pair of wheels whose movement is done via a DC geared motor. The base also has a robotic arm.

3.3.4 Robotic arm

The arm consists of a gripper which is connected to an actuator that moves up and down. There is another actuator connected to the previous one which moves back and forth; this movement is done using DC geared motors. The below given is the flow diagram which shows the work flow of the system.

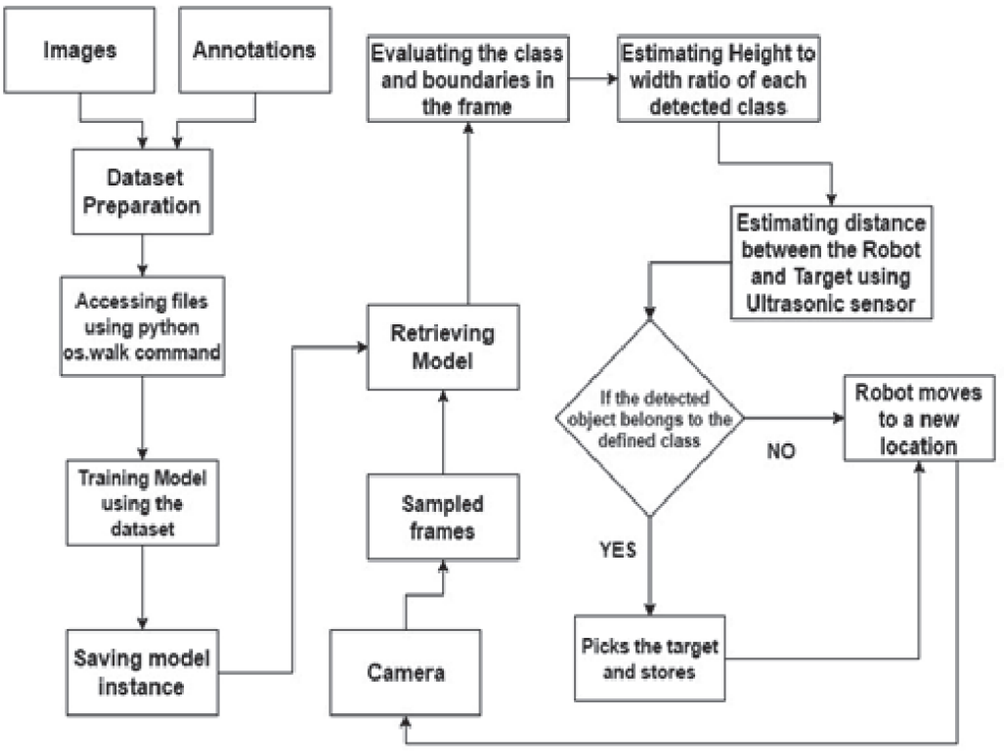

Figure 10.
Flow diagram of the system.

Classification of object is done using image processing and TensorFlow. Camera is used to capture the image of the object present on the surface which is then matched with the pre-trained dataset. Here the object detection is achieved using YOLO object detection, and the detected object is directed to its respective class. There are several modes that are made available such as plastic, glass, metal, degradable, and nondegradable. Based on the information of the selected mode, either the object is picked up or ignored. The object is picked up using a robotic arm which is controlled using the pre-programmed Arduino board. Once the desired object is picked up in accordance with the selected mode, object classification is completed successfully. Advances are done in proposed method in comparison with the existing method:

- Some areas in the design of the robot in existing method have limitations; these limitations may be overcome by improving upon the design.

- Thus we have made some improvements in the design of the robot by making it a bit smaller, adding a webcam which is weightless and making the wheel base strong and stable.

- We have also removed the bins available on top of the robot as the previous one only classifies between two classes but our robot classifies almost between more than three classes.

- The above robot mentioned in the existing method uses manual pickup of the classified waste, but we have added a robotic arm gripper which automatically picks up the classified object.

- Also the Recyclebot is controlled manually over a Zigbee powered joystick application, but our robot is fully automated and uses Arduino to achieve it.

- On the software side, we have made some major improvements like we have used OpenCV and TensorFlow for processing the obtained image.

- Also we have used an algorithm called YOLO for object detection and classification which provides better accuracy when compared with the previous method.

- Since the robot moves by itself, the number of hardware and software used is reduced which means there is no need for Zigbee and the joystick application anymore.

- When it comes to recognizing objects, the Recyclebot only recognizes a small set of images which is around 1000 to 2000.

- But our robot could almost recognize more than 5000 images which prove to be a larger image set than the previous one.

- Also the speed of processing is improved. The accuracy of our robot is far better for an automated robot.

- Thus this shows that our robot is much better than its predecessor.

3.3.5 Skin cancer classification using convolutional neural network

Skin diseases are becoming the most common health issues worldwide. In this paper we propose a method that detects four types of skin disease using computer vision [10, 11]. The proposed approach involves convolutional neural networks with specific focus on skin disease. The convolutional neural network used in this paper has utilized around 11 layers, namely, convolution layer, pooling layer, activation layer, fully connected layer, and softmax classifier. Images from the DermNet database are used for validating the architecture. The database comprises all types of skin diseases out of which we have considered four different types of skin diseases like acne, keratosis, eczema herpeticum, and urticaria with each class containing around 30–60 different samples. The challenges in automating the process includes the variation of skin tones, location of the disease, specifications of the image acquisition system, etc.

Author details

Anand Raju*† and Shanthi Thirunavukkarasu†
Department of Electronics and Communication Engineering, Sona College of Technology, Salem, Tamil Nadu, India

*Address all correspondence to: anandvimal1@gmail.com

† These authors contributed equally.

References

[1] Sainath TN, Mohamed AR, Kingsbury B, Ramabhadran B. Deep convolutional neural networks for LVCSR. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing. IEEE; 2013. pp. 8614-8618. Available from: http://www.cs.toronto.edu/~asamir/papers/icassp13_cnn.pdf

[2] Simard PY, Steinkraus D, Platt JC. Best practices for convolutional neural networks. In: International Conference on Document Analysis and Recognition (ICDAR). 2003. p. 958

[3] Lawrence S, Giles CL, Tsoi AC, Back AD. Face recognition: A convolutional neural-network approach. IEEE Transactions on Neural Networks. 1997; **8**(1):98-113

[4] Liu L, Shen C, van den Hengel A. The treasure beneath convolutional layers: Cross-convolutional-layer pooling for image classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015. pp. 4749-4757

[5] Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems. 2012. pp. 1097-1105

[6] Graham B. Fractional max-pooling. 2014. arXiv preprint arXiv:1412.607

[7] Anand R, Shanthi T, Sabeenian RS, Veni S. Real time noisy dataset implementation of optical character identification using CNN. International Journal of Intelligent Enterprise. 2020;**7**(1-3):67-80

[8] Anand R, Kalkeseetharaman PK, Naveen Kumar S. Automatic facial expressions and identification of different face reactions using convolutional neural network

[9] Anand R, Shanthi T, Nithish MS, Lakshman S. Face recognition and classification using GoogleNET architecture. In: Soft Computing for Problem Solving. Singapore: Springer; 2020. pp. 261-269

[10] Sabeenian RS, Paramasivam ME, Anand R, Dinesh PM. Palm-leaf manuscript character recognition and classification using convolutional neural networks. In: Computing and Network Sustainability. Singapore: Springer; 2019. pp. 397-404

[11] Shanthi T, Sabeenian RS, Anand R. Automatic diagnosis of skin diseases using convolution neural network. Microprocessors and Microsystems. 2020;**76**:103074