Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2022 SCPE. Volume 23, Issues 4, pp. 263–274, DOI 10.12694/scpe.v23i4.2029

A NEW IMPROVED BINARY CONVOLUTIONAL MODEL FOR CLASSIFICATION OF IMAGES

PUTTA HEMALATHA*, SHANKAR G† AND DEEPAK RAJ D.M‡

Abstract. There are numerous image classification strategies are developed in deep learning. However, due to the complexity of images, conventional image classification strategies have been incapable to meet real application needs. As the amount of pixel information rises, the classification becomes more difficult. However, CNN is widely used method for object identification in picture due to its simple and accurate, but still, it remains hazy which strategies are most supportive for analysing and distinguishing the objects in pictures. In this paper we introduced a CNN network and clustering-based technique called IBCNN to perform classification based on patch extraction. The proposed method can accomplish their goals in the following four different ways: a) Automatic Kernel selection; b) resilient patch size selection; c) CNN layer; and d) pooling layer modification. In addition, it also modifies the pooling layer with average value and calculate the pixel size. The proposed method was applied on ten different image datasets. Finally, the proposed model is compared to three benchmarking models: such as WCNN, MLP, and ELM-CNN to estimate its performance. The obtained results shows that the proposed method gives competitive results compared to the other models.

Key words: classification, machine learning, deep learning, image classification, convolution network

AMS subject classifications. 68T05

1. Introduction.

To gain a complete image interpretation, we should not only focus on classifying different images, but also try to precisely estimate the concepts and locations of objects contained in every image. In general, pictures contain numerous connected data, which causes troubles within the classification errand and permits opening on diverse investigate challenges. In addition pictures may contains commotions and peculiarities, the separation between these substance is regularly troublesome and can lead to a tall rate of misinterpretation. The extraction of objects and characteristics is exceptionally troublesome and vital for an exact classification [1]. Deep learning-based methods have grown quite prominent among many Artificial Intelligence methodologies. It's utilised to solve different challenges in the fields of object detection, image classification and computer vision. Deep learning has become a very promising technology as a result of technological advancements and the availability of large datasets. Convolution Neural Network (CNN) is one of the prominent models widely used in image classification process [2]. CNN is the main deep learning technique, have outperformed most machine learning algorithms in a variety of real-world image classification or object identification applications. It is well known that the architecture of CNNs has a significant impact on their performance [3]. Images are large-scale representations of the object, made up of several pixels and shape data. The goal of convolution is to detect a specific object in an image while ignoring undesirable pixels or image features to handle a variety of problems such as security, cartography, pixel monitoring, shape, edges, and so on [4]. In general, images are large and include a bunch of associated information, which makes classification complex and allows you to open them on multiple research channels. The task of autonomously capturing these large images is as challenging as it is vital. Furthermore, multiple object images may contain noises and irregularities, making it difficult to distinguish between these contents, perhaps leading to misinterpretation [5]. Extraction of patterns, pixels, edges, and textural features is difficult but crucial for accurate classification [6], [7]. In the disciplines of identification and detection, deep learning algorithms have exhibited real-world

*Institute of Aeronautical Engineering, Department of Information Technology, India (p.hemalatha@iare.ac.in)
†R.M.D. Engineering College, Department of Computer Science and Engineering, India (gs.cse@rmd.ac.in)
‡Vel Tech, Rangarajan, Dr. Sagunthala, R&D, Institute of Science and Technology, Department of Computer Science and Engineering, India (drdeepakrajdm@veltech.edu.in)

performance [8]. There are a number of options, including Autoencoders, Stacked Denoising, weighted CNN (WCNN) [9], Multilayer Perceptron (MLP), extreme learning machine CNN (ELM-CNN), and others [10]. The number of neurons, the number of hidden layers, and other characteristics among CNN's, can have a substantial impact on categorization results [11]. The problem statement of object location and detection is to decide where objects are found in each image (image localization) and which category each images depends. So, the channel of conventional object detection methods can be basically distinguished into four stages: a) kernel selection, b)patch selection c) CNN layer, and d) pooling layer estimation.

In this paper, we proposed a combine clustering technique with a modified CNN to propose an improved binary classification approach; hence, the number of kernels is initialised using CKmeans. In addition, we proposed an adjusting max-pooling layers using the average number of patch size value. Finally, we apply a different patch size on each image to select the optimal pixel for better classification.

The article is organized by the following sections: related works described in section-2; section -3 explains the proposed approach, whereas, section-4 is experiment and discussion, section-5 shows the result, and article end up by conclusion in section-6.

2. Related Work. Generally, images are a big scene composed of several pixels. Each of the pixels brings such important information. Unlike other types of images, where each pixel is described by one value, the pixels of the multiple objects contain images are described by a vector of values. This makes the classification of images somewhat cumbersome and complicated. Deep learning has the advantage of extracting data, which allows classification more precisely. To help readers better understand the associated research and the suggested method, this part introduces 12 [13]. CNNs and skip connections, which are regarded the background of the proposed algorithm. The work on discovering the architectures of CNNs is then examined. CNN, a multilayer perceptron, was inspired by the visual neural mechanism. A CNN usually has three layers: an input layer, a hidden layer, and an output layer. The input layer calculates the mean value and uses PCA/whitening to normalise the data. The convolution, excitation, and pool layers are commonly included in the hidden layer, which is made up of numerous neurons and connections between the input and output layers. Numerous activation functions should be employed if the CNN has multiple hidden layers [12].

The ReLU and sigmoid functions are two often used activation functions. Because of the problem of strongly correlated pixels, and change in suggested CNN architecture. For instance, Discriminatory abilities, can be increased by using the middle layer and increasing stall rates, as well as adding convolutional layers. During the convolutional operation, the filter slides horizontally (with a specific step size), then vertically (with a different step size) for the next horizontal slide, and so on until the entire image has been scanned. The feature map is a new matrix formed from the filter outputs. The breadth and height of a stride are the horizontal and vertical step sizes. In our research, we have discussed WCNN [9], MLP [10], and ELM [11].

The weight parameter affects input data inside the network's hidden layers, making WCNN a neural network. A neural network is made up of a series of nodes (also known as neurons) that connect together to form a network. Each node is made up of a collection of inputs, a weight, and a bias value. A node receives an input multiple times, and the same it multiplies by a weighting factor to observes or either sends the result to the next hidden layer of the network [14]. The hidden layers of a neural network are typically used to store the network's weights.

A neural network's input layer accepts input signals and forwards them to the next layer. After that, the neural network has a series of hidden nodes that modify the data input. The weights are applied to the buried layer nodes' nodes. For instance, a single node may multiply the incoming data from a well before weighting factor and transferring things to a whole new layer, and then add a bias. The output layer of the neural network is indeed known as the final layer. The concealed layers' input is regularly tweaked by the output layer to create the needed numbers in a specific range represented in Fig. 2.1 (a). A feedforward artificial neural network called a multilayer perceptron (MLP). An MLP is made up of three layers of nodes: an input layer, a hidden layer, and an output layer. Except for the input nodes, each node is a neuron with a nonlinear activation function [15]. MLP employs backpropagation during training as a supervised learning technique. The multiple layers and non-linear activation of MLP separate it from a linear perceptron. It can distinguish between non-linearly separable data. Multilayer Perceptron (MLP) is a sort of computer vision technique that has been replaced by CNN's as seen in Figure 2.1(b). MLP is no longer considered adequate for advanced computer vision tasks.

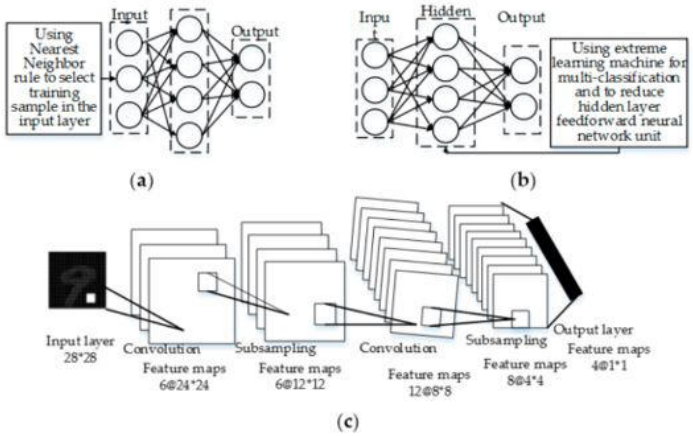

Fig. 2.1: Model of comparison algorithms using a convolution neural network structure a) WCNN, b) MLPCNN and C) ELM-CNN.

Every perceptron gets linked to all of the other perceptron's, resulting in fully connected layers [16].

The extreme learning machine (ELM), a revolutionary paradigm in the field of machine learning, has received a lot of academic attention in recent years. The predictive model is a generalised version of the single-layer feedforward neural network (SLFN), which only contains one hidden layer with randomly generated hidden neurons. SLFNs are networks that do not have any back or side connections between nodes. Fast computational time during training and testing is one of ELM's primary features. Furthermore, unlike typical neural networks, the classifier is capable of achieving strong generalisation without the need to tune network parameters [17]. Figure 2.1(c) shows how ELM has been applied in a range of different learning contexts. ELM, in comparison to earlier tuning-based approaches, necessitates a greater number of hidden nodes in practise. However, in such networks, a small number of hidden nodes may have a minor impact on the network's output, boosting the network's intricacy [18]. Furthermore, if indeed the current hidden node surpasses total training set and the ELM may experience a singularity problem, making the system unstable.

2.1. CNN – General Working Principle. CNN is a type of neural network that has shown to be particularly effective in areas like image recognition and classification. Applications include object detection, vision in robots, autonomous cars etc., and it also used in smart grid applications as well, because, smart grid generates huge amount of data. The general working procedure of CNN is shown in figure 2.2(c). Typically, CNN has three types of layers to build architecture such as convolutional layer with ReLU activation, Pooling layer and fully connected or dense layers. Depict from the figure 2.2, convolutional layer extract information from the input picture, the next layer is pool1 layer reduce extracted information from conv1 which is also called down-sampling layer [19], [20]. Next layer is pattern which is used to convert matrix into vector. Dense layer usually available at the end of convolutional layer, based on fig. 2.2 the process to classify the given image is cat or dog, and the output layer exactly says that the given image is cat or dog with SoftMax activation. Convolution's primary objective is to collect the significant features of an image. By learning input features, convolution retains the spatial link between input. Then the convolutional layer produces output by simply convolving input image with filter. The mathematic representation of CNN is defined as: where y→ output image, x→ input image, which is actually representing the input image with $N * N$ matrix, n represent the size of the input image, 'f' represents size of filter which was chosen for the given example. The size of the filter can change as per the requirement or problem description. Typically, computer interpret image in the form of grey scale (i.e., 0 and 1) and RGB (i.e., 0-255). Depict from figure 2.2, the given image convoluted and distinguished by a layer action in which image's pixels are extracted one by one in a given size patch x→ $n * n$ matrix, with each active pixel enclosed through group of nearby pixels. Second, each patch will traverse the

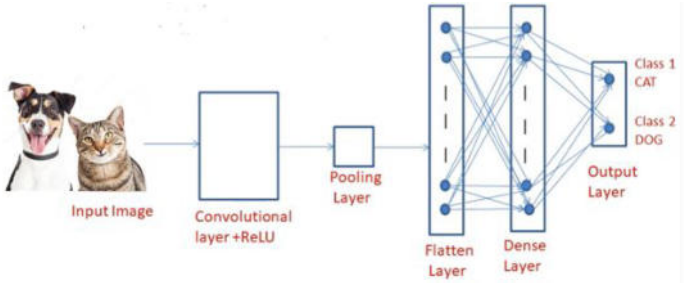

Fig. 2.2: General working procedure of CNN.

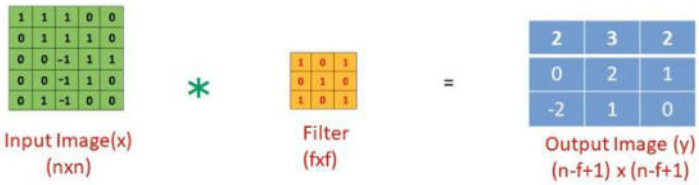

Fig. 2.3: Convoluting the input images 5*5 with 3*3 filter and produce output image 3*3 dimensions.

convolution and max - pooling several times. Third, it will be twisted using the various specified kernels at the convolution layer. According to the filter selected, it would be treated only at max pooling. Finally, the patch gets smaller and smaller until it's just a simplified data vector.

$$y = \sum_{n=1}^{n} x * f \tag{2.1}$$

The content of the pixel will be determined by comparing these values to the database [21], [22]. From the figure 2.3 and 2.4 the input image sample chosen as $5*5$ dimension with filter $3*3$ dimension. When the input image is convolved with $3*3$ filter produce output image $3*3$ dimension.

In order to find (1,1) element of this output with $3*3$, lets super impose this filter into input image such that (1,1) of this filter is super imposed on (1,1) of input image; (1,2) of this filter is super imposed on (1,2) of an input image and so on till (3,3) of this filter is super imposed on $3*3$ element as shown on in figure 3(a). Therefore, element wise multiplication $1*1$, $1*0$, $1*1$, $0*0$. $1*1$, $1*0$, $0*1$, $0*0$, $-1*1$ produce result 2. Similarly, if we want to find $1*2$ element of this output image matrix we have to shift this filter ones on the right side, so that (1,1) filter super imposed on (1,2) of an input image and (1,2) of this filter is super imposed on (1,3) of an input image and so on till (3,4) of this input image. And, once again the element wise multiplication and add all the values will be finding 3 and so on till finding 0 in the out image. To understand little more about the pixel calculation is broadly described in figure 2.4.

3. Proposed Method. We start by separating the cat image (named H) into multiple scales because the input image is 2D shaped. Each scale is dealt with independently. Kernels are extracted from each scale. However, the problem is figure ring out how to identify the number, position, and length of kernels, as well as the length of both the patches that encircled each pixel. To solve this problem, we proposed an algorithm called IBCNN (Improved Binary CNN) using the CK-means method with a few improvements. (1) Kernel selection (Number, Position, and Size), (2) Patch Extraction, (3) Hidden layer processing, and (4) Classification are the four primary procedures in our system as shown in algorithm 1.

3.1. Kernel Selection. The proposed algorithm begins making a feature map, which contains locations of all the important pixels (kernel centre). In this paper, the CK-means algorithm has been adopted to determine

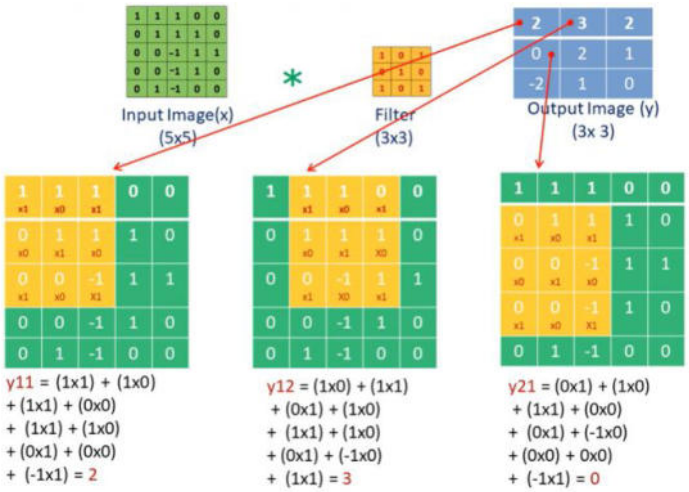

Fig. 2.4: Comprehensive convolution layer operation.

kernels. CK-means intends to stake the X= $x_1, x_2, x_3.x_n$ in vague pixel 'p' with 'μ_{ij}' represent cluster along with $X_i \rightarrow$ degree of membership in respective of ith clusters. The membership degree on the matrix represents the clustering outcome μ.

First and foremost, set the parameters: p $\rightarrow$ number of clusters n $\rightarrow$ number of training data m $\rightarrow$ fuzzy parameter used to define the width of p parameter

For the number of cluster p initialization, for the two relative integers n and p the Euclidean distance division was used to find the mean difference between patch size along with zero value in distinct p. Euclidean division connects two relative integers i.e., quotient $\leftarrow$ n, remainder $\leftarrow$ m, with respect to training sample n: n = pm + w, where 0$\leq$ w $\leq$ $\mid p \mid$. This scenario could help to cover as much area as possible, whereas, 'w' should be decreased with respect to the predefined parameter m, n and p; and m ϵ (1.25,2). Procedure involved in the proposed algorithm as follows: To begin, split each pixel and construct x vectors of data one by one. For data ranging from 1 to n: 1) In order to generate number of clusters p, the fixed predefined fuzzy parameter m have to select random patch size in cluster. 2) Generate novel matrix Φ; after the membership degree φ has been determined.

$$\Phi_{ij} = max\Big(\frac{\Phi_{ij}}{\sum_{i=1}^{n}\Phi_{ij}} \mid \frac{\Phi_{ij}}{\sum_{i=1}^{n}\Phi_{ij}}\Big) \tag{3.1}$$

from eqn (3.1) follow the procedures to calculate the Φ_{ij} matrix: i) To begin, fill a random number between 0 and 1, to ensuring that the sum of each row or column is 1. ii) second, the largest integer in a place can be given the value 1 in a new matrix, while the others are given the value 0. iii) at last, create a vector that consisting total of the values in each column. Give a value in column as 1 if it includes or else 0, thus ultimately assist to find the largest value in vector. After that, calculate the centroid cluster '$Cent_j$' with respect 'j'.

1) Use eqn(3.2) to calculate the centroid value of the cluster can be defined as:

$$Cent_{ij} = \frac{\sum_{i=1}^{n} x_{ij}\Phi_{ij}}{\sum_{i=1}^{n}\Phi_{ij}} \tag{3.2}$$

2) Calculate initial value j_A using eqn(3.3):

$$J_A = \sum_{i=1}^{n}\sum_{j=1}^{n}\Phi_{ij}, d_{ij}(x_i, c_j)^2 \tag{3.3}$$

Data: set of input image datasets D for classification, Maximum pixel number generation
Result: classification of given images y=Cn (n=1,2,3…
Initialization $I_0 \leftarrow$ Using the proposed variable-kernel selection technique, initialise a pixel size with the provided image size.
step 1 Kernel selection
while $t \leftarrow$*0 the maximal generation* **do** $\Phi_{ij} = max\left(\frac{\Phi_{ij}}{\sum_{i=1}^{n}\Phi_{ij}} \mid \frac{\Phi_{ij}}{\sum_{i=1}^{n}\Phi_{ij}}\right)$
if Φ_{ij} *is even* **then**
estimate number, position, and size of the kernel with ckmeans, **then**, calculate the centroid value of the cluster
$Cent_{ij} = \frac{\sum_{i=1}^{n} x_{ij}\Phi_{ij}}{\sum_{i=1}^{n}\Phi_{ij}}$
elseif *threshold* $\theta > 0$
step 2 patch size estimation *calculate the average of the current pixel and its neighbours pixels using* $W_{i,j}$
then$W_{ij} = (\frac{x_{vi}+x_k}{n}, \frac{y_{vi}+y_k}{2})$
$p_i x_1(x_i, y_i) = (x_{ci} \colon y_{lu})$
step 3 CNN layer
for each cnn $\rightarrow C_i : Ci \leftarrow I_0 * W_1$ select active pixel size $C_i \rightarrow$ choose 'n' α and β to map convolution layer l calculate the average of the current pixel:
$y_n^{l-1}(i,j)\sum_{x=1}^{a=1}(y_n^{l-1}, (i-x, j-1), ...y_n^{l-1}(1-x, j), ...y_n^{l-1}(i, j-x).., y_n^{l-1}(i,j))$
step 4 Average_pooling layer
for each the filter size is a*a and
if a ≥ 1 **then** $y_n = f_1(z_n^{l-1} * w_n^l + \beta_n^1)$
end if
end if
end while

Algorithm 1: Proposed IBCNN Algorithm

3) Calculate fuzzy parameter $Cent_{ij}$ using eqn(3.4):

$$Cent_{ij} = \frac{(\frac{1}{d_{ij}(x-i,c_j)})^{\frac{2}{m-1}}}{\sum_{i=1}^{n}(\frac{1}{d_{ik}(x_i,c_k)})^{\frac{2}{m-1}}} \tag{3.4}$$

4) By using Euclidean distance find dij between x_i and c_j.

5) Check the finishing condition, and then set the positive parameter $\in > 0$ from the beginning.for better result we chosen $\in = 0.001$.

$$d_{ij}(J_{a-1}, J_A) \leq \in \tag{3.5}$$

If $J_A$is the previous iteration's objective function and $J_{(A-1)}$ is the current iteration's objective function, then move to eqn. (3.3) to (3.5) otherwise return to (2). We build a binary matrix from the clustering matrix to represent the kernel placements. To accomplish this, we've established a threshold parameter θ. The number of neighbouring pixels[3, 2], that are similar to centre pixel or active pixels $\rightarrow$ k. Active pixel calculating is illustrated in greater detail in the following diagram figure 3.1. Where, I $\in$ [2, 0], in the centre pixel or active pixel 'x', if $p_i x_i = =$ X, then the nearest pixel value could $N++$. However, if $N \geq \theta$ the centroid cluster X must be marked as 1 otherwise 0.

3.2. Patch size estimation. In patch calculation, initially we have to set a threshold θ to consider. Based on the size of the image the pixel value may assigned. For instance, let assume v be the nearest kernel centre neighbors i.e., v = v1, v2, v3, v4. The smaller value has been chosen by calculating the distance between each pixel and each kernel's centre. Then, calculate the average of the current pixel and its neighbours' pixels using

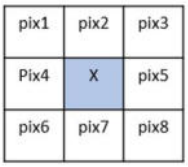

Fig. 3.1: Estimation of centre pixel 'x'

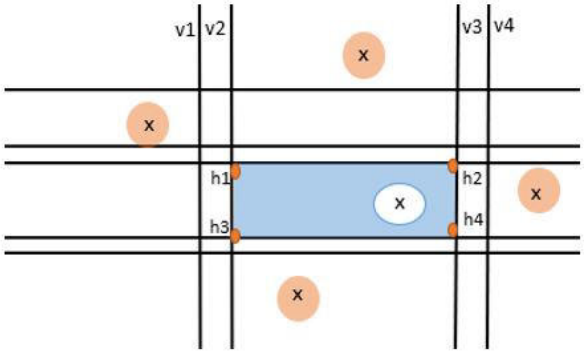

Fig. 3.2: Selection of kernel size in coordinates intersection

eqn 3.6.

$$W_{ij} = \left(\frac{x_{vi} + x_k}{n}, \frac{y_{vi} + y_k}{2} \right) \tag{3.6}$$

The ordinate of the graph is represented by x and y point respectively with denoting the active pixel k. We choose the equivalent outlines with respect to x and y axes, that cross on all the generated points after selecting the new points. Constant affine functions fc(x) is used for to estimate the neighbors kernel function i.e., fcx = ax+b, where, constant a=0, and co-ordinates of the neighbor kernels b. Four horizontal lines h1, h2, h3, h4 which represent rows and four vertical lines v1, v2, v3, and v4 represents columns are used to find the co-ordinates intersection values. The rectangle's coordinates to be retrieved or represented by intersection values, as shown in Fig 3.2. The active pixel's space '1' is then divided and shifted into four sections: up, down, left, and right. The new parameter position p can estimate as: $p_i x1 = m_n \wedge x_l$, $p_i x2 = m_n \wedge x_r$, $p_i x3 = m_d \wedge x_l$, $p_i x4 = m_d \wedge x_r$ with resection lines ln and column cn. The position of the active pixel can be evaluated with the build objective function p_i(x, y) = (xL n: d + xC r: l + yL u: d + yC r: l). Therefore, when applying pi(x, y) to calculate the neighbors pixel size the intersection coordinates y_C and x_l are seeming to be equal because they are parallel to the axes. So, some position of the parameter might change with respect to the initialization values and the appropriate steps taken to adjust as follows:

select_start_position_1

$$p_i x_1(x_i, y_i) = (x_{ci} : y_{lu}) \tag{3.7}$$

select_start_position_2

$$p_i x_2(x_i, y_i) = (x_{cr} : y_{lu}) \tag{3.8}$$

select_start_position_3

$$p_i x_3(x_i, y_i) = (x_{ci} : y_{ld}) \tag{3.9}$$

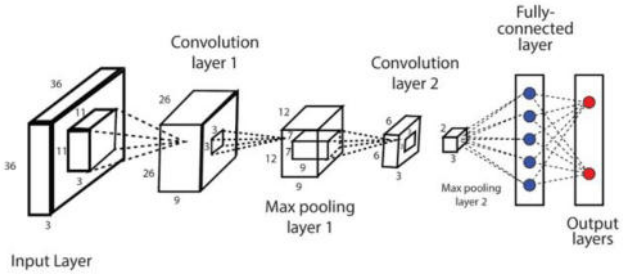

Fig. 3.3: Overview of CNN work cycle.

select_start_position_4

$$p_i x_4(x_i, y_i) = (x_{cr} : y_l) \tag{3.10}$$

Training, testing, and validation are all performed on the extracted sample from the input image. Therefore, the training samples will be subjected to the following treatment. The pixels in the centre of the kernel X are used to find the co-ordinates of pixel h1-h2:h3-h4 and vertex v1-v2:v3-v4 as shown in fig 3.2.

3.3. CNN Layer. The convolution layer weights the kernels chosen and adds local neighbours to every image pixel. The centre element of each kernel is positioned upon that active pixel as shown in fig 3.3. A weighted sum of the neighbours' pixels and itself will replace active pixel. Depict from eqn. 3.11 let y_n be given input image determines the image patch size Bi and number of layer 'n' α and θ are the two positive integers used to map the convolution layer 'l'.

$$y_n^l = f_i(\sum_{m \epsilon n} y_m^{l-1} + \beta_n^1) \tag{3.11}$$

However, a convolutional operator represented as ϕ and f_i is an activation function of the layer, in the average_pooling layer S_1, the bias for attributes map β_n^1 The W_n^l column displays a pixel group on each layer 'l-1' which related to the n-dimensional characteristic map. At last, the convolution kernel W_n^l, come into the picture to calculate and assign the different filters.

3.4. Average pooling layer. The average_pooling layer permits each average_pooling pixel's input size to be halved or greater (i.e., depending on the filter size). Estimate the average pixel size in term of a*a for each patch, which corresponds to the filter's length; before being added to the bias, such outcome would be increased from a tuneable weight. In every average_pooling layer if the filter size is a*a and if a ≥ 1 as well then, the output N*N matrix $\rightarrow z_n^{l-1}$(i,j) can be calculated as:

$$y_n^{l-1}(i,j) = \sum_{x=1}^{a=1}(y_n^{l-1}, (i-x, j-1), ...y_n^{l-1}(1-x, j), ...y_n^{l-1}(i, j-x).., y_n^{l-1}(i,j)) \tag{3.12}$$

Using eqn. 3.12, let $y_n^{l-1}(i,j)$, as a result of the average of such images in every patch and the features map 'n', the average_pooling layer 'l' is now computed with respect to y_n input shown in eqn 3.13.

$$y_n = f_1(z_n^{l-1} * w_n^l + \beta_n^1) \tag{3.13}$$

The output image map size $Z^l W^l$, where $H \rightarrow$ horizontal position of the pixel and $W \rightarrow$ represent vertical position of the pixels. In the final convolution layer, each pixel is linked to exactly 1 preceding features map. The input characteristic maps are convolution kernels from the same size. Sigmoidal neurons make up the output layer referred as 'y', whereas, 'n' specifies the amount of sigmoidal output neurons. Therefore, eqn. 3.12 is the equivalent equation for computing the output of a sigmoidal neuron. Here β seems to be the biased linked with both the 'L' layers and 'n' neuron, and 'w' is the weight of the last convolutional layer's characteristic map 'P' at the output layer's neurons 'N'. Finally, compute all of the sigmoidal neurons' outputs that lead to a network's results 'y':

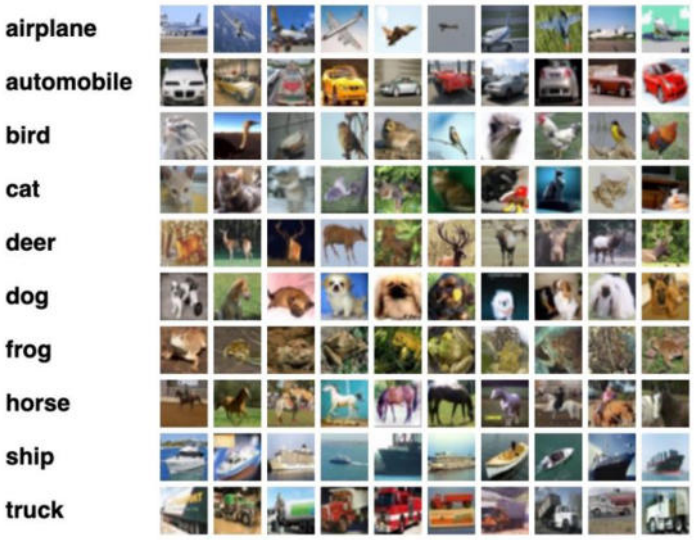

Fig. 5.1: Visual description of the dataset. However, column represent the dataset label and row represents the types of the dataset.

Table 5.1: Characteristics of the dataset

Name of the Dataset	Total number of images	pixel dimension
airplane	10028	227 * 227
automobile	45001	1080 * 1080
bird	35484	800 * 600
truck	14750	720 * 720
frog	17584	227 *227
deer	14856	1280* 1227

4. Experiment and Discussion. The quadratic CNN learning method is implemented on a 64-bit Ubuntu 16.04.4 using the Python programming language. TensorFlow 1.5.0 is used to evaluate the accuracy of the proposed method. The reliability of said dropout-based CNN algorithm and the SGD optimizer are assessed [23], examined, and associated to the other three models at diverse learning rates such as training loss or training accuracy and validation loss or validation.

5. Dataset Description. To demonstrate the efficiency of our suggested method, we evaluated the model on ten different image datasets. The characteristics of the datasets is shown in table 1 and visually in Fig 5.1. The method was compared with different models such as weighted CNN (WCNN), convolution with fully connected multilayer perceptron (MLP-CNN), external learning with support vector machine (ELM-SVM)[24]. However, the datasets are publicly available at(https://paperswithco
de.com/dataset/cifar-10).

6. Result. We retrieve the pixel size from each input image for experimental purposes. In tradition CNN various nonlinear functions such as Tanh, Sigmoid, ReLU and leaky ReLU are used as activation functions. However, in this work, we have used ReLU [25], as a common activation function for the entire process which help easily to maps a real number in to [1,0]. For each pixel, we estimate a cluster matrix, then create a new feature map (binary matrix) that specifies the locations and sum of kernels from each matrix which could help to retrieve the kernel size from the feature map. Typically, the proposed method conducted two different tests:

Table 6.1: setting patch size, pooling size, convolutional kernel size

Dataset	airplane	automobile	bird	apple	fish	baby	cats	dogs
size of the patch	36*36	36*36	24*24	21*21	36 * 36	21 * 21	9 * 9	36 * 36
Convolutional kernel size	12*12	12*12	6*6	5*5	5*5	9*9	5*5	5*5
Pooling filter size	9*9	9*9	5*5	3*3	3*3	6*6	3*3	3*3

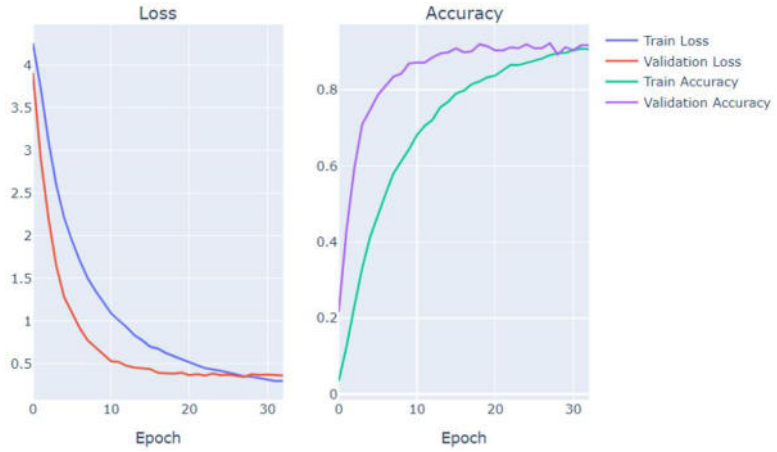

Fig. 7.1: Performance analysis in term of loss and accuracy

first, it involved applying a constant patch size (see Table 5.1). Secondly, we returned patches of varying sizes, the size of which is governed by the dimensions of the input image and the number of kernels produced.

To test if our proposed technique is effective, we first compare the performance to an arbitrarily pre - defined number of kernels. Each of the ten datasets is subjected to the tests. The proposed algorithm calculates the size of the kernel in the second testing, usually mean number of hidden layers completed on all datasets is between 6 and 12 depending on the number of kernels generated as well as the magnitude of such original pixel size. The suggested technique finds the number of kernels to be 50, 55, 60, 65, 70, 75, 80, 85, 90, and 95 for each of the ten datasets. Indeed, in the other variants, we manually chose the number of kernels: 35, 40, 45, 50, 55, 60, 65, 70, 75, and 80. The precision or number of kernels on every approach was set according to the values defined in columns, are shown in Table 6.1. This suggested methodology offers the highest categorization accuracy. As a result, we can conclude that the suggested method can locate a large number of kernels for various images.

7. Performance Analysis. The performance analysis of the proposed model evaluated in term of loss and accuracy of validation and train metrics. However, the validation accuracy range is increased from 0.8 to 0.95. The recognition rate of the model increases as the learning rate increases, and the learning and recognition rates have a positive relationship, as shown in Figure 7.1. When the SGD optimizer's training an accuracy learning rate reached 0.94, the model's recognition rate was relatively constant. The validation accuracy and train accuracy of the proposed model are currently growing faster than the previous model, The accuracy rate rises considerably whenever the learning rate goes up to more than 0.06. The proposed algorithm is executed ten times on each dataset. Table 6.1 describe the kernel size as well the filter size, and Figure 7.2 illustrate the accuracy of the proposed algorithm comparing with other algorithms.

According to [26], for the airplane dataset test, the proposed algorithm achieved 2% higher than MLP-CNN, however the parameter we selected here to test is 10.2M. Similarly, IBCNN achieved greater result on bird, apple, fish, baby, and dogs. However, MLP-CNN achieved 1% higher on cat's dataset than other models. WCNN and IBCNN performed 2% greater on apple dataset. ELM-SVM attained reasonable accuracy of 91% with 2.2M

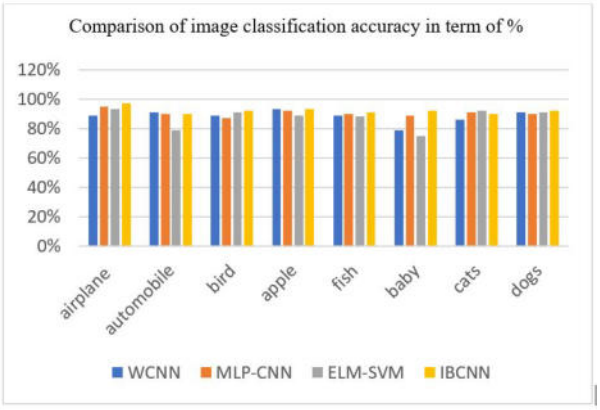

Fig. 7.2: Accuracy comparison with other image classification models

parameter, with 9*9 patch size, however, these accuracies are estimated by selecting the patch size and manual parameter [27], [28]. Another parameter: pooling filter size 5*5, 9*9, 3*3 are used to test WCNN, IBCNN, MLP-CNN. Similarly, the validation accuracy is obtained 0.95M which is 2% higher accuracy than another model. Depict from figure 7.2, the accuracy of classification is evaluated in %, based on training accuracy and validation accuracy IBCNN gained 97% on airplane dataset, 93% on apple, 91% on fish, 92% on baby, 92% on bird, and 92% on dogs which is greater than WCNN, MLP=CNN, ELM-SVM. As a result, we can conclude that the suggested method can locate a large number of kernels for various images and obtained highest accuracy in classification of object in image. The proposed technique has the highest accuracy in classification. As a result, we can conclude that the suggested method can locate a large number of kernels for various images. The classification accuracy of the proposed approach is marginally greater than that of existing CNN models for peer competitors in the second category.

8. Conclusion. Object detection in image is a cutting-edge technique that can be used in a variety of security, autonomous, and surveillance applications. Classifying and object identification in large picture or image in deep learning is achieved benchmarking result with the help of convolutional network. The use of CNN for image classification is a solution that involves data and parameter initialization. We developed a classification approach in this paper that combines clustering and convolutional neural networks. There are four significant contributions of the work as: (1) using a clustering algorithm to determine the number of kernels; (2) picking the kernel placements; (3) computing an adaptive patches size; and (4) softening the pooling layer. On ten different image datasets, the proposed approach was tested. The proposed model's performance was assessed in terms of validation and train metrics loss and accuracy. The validation accuracy range is extended from 0.8 to 0.95, the model's recognition rate increases as the learning rate increases, and the learning and recognition rates have a positive relationship. The results show that the proposed strategies are competitive with alternative approaches. The further work suggested on spatial images or hyperspectral image classifications.

REFERENCES

[1] Milan Tripathi, *Analysis of Convolutional Neural Network based Image Classification Techniques*, Journal of Innovative Image Processing (JIIP), Vol. 3 No.2, July 2021, pp 100-117.

[2] Y. Sun, B. Xue, M. Zhang, G. G. Yen and J. Lv , *The LaTeX Automatically Designing CNN Architectures Using the Genetic Algorithm for Image Classification*, IEEE Transactions on Cybernetics, Vol. 50 No. 9, September 2020, pp. 3840-3854.

[3] G. Huang, Z. Liu, L. Van Der Maaten and K. Q. Weinberger , *Densely Connected Convolutional Networks* , IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2261-2269.

[4] L. Xie and A. Yuille , *Genetic CNN*, In Proceedings of 2017 IEEE International Conference on Computer Vision, 2017, pp. 1388–1397.

[5] Y. Sun, B. Xue, M. Zhang and G. G. Yen , *Evolving Deep Convolutional Neural Networks for Image Classification*, in IEEE Transactions on Evolutionary Computation, Vol. 24 No. 2, April 2020, pp. 394-407.
[6] M. Liu, D. Cheng, and W. Yan , *Classification of Alzheimer's disease by combination of convolutional and recurrent neural networks using FDG-PET images*, Frontiers in Neuroinformatic, Vol. 12 No 2, March 2018, pp. 35-43.
[7] Feng-Ping , *A Medical Image Classification Algorithm Based on Weight Initialization-Sliding Window Fusion Convolutional Neural Network* , Hindawi, Complexity, Vol. 2019 No.12, Article ID 9151670, 2019, pp. 1-15.
[8] A. Singh , *Detection of brain tumor in MRI images, using combination of fuzzy c-means and SVM* , In Proceedings of the 2015 2nd International Conference on Signal Processing and Integrated Networks (SPIN), 2015, pp. 98–102.
[9] Yang, Jing, and Guanci Yang , *Modified Convolutional Neural Network Based on Dropout and the Stochastic Gradient Descent Optimizer* , MDPI Algorithms, Vol. 3 No. 28, March 2018, pp.1-15.
[10] Singh P., Verma A., Chaudhari N.S , *Deep Convolutional Neural Network Classifier for Handwritten Devanagari Character Recognition* , Information Systems Design and Intelligent Applications. Advances in Intelligent Systems and Computing, Vol 434. Springer, 2016, pp. 551-561.
[11] Mahmood, Saif F., Marhaban, Mohammad H., Rokhani, Fakhrul Z., Samsudin, Khairulmizam and Arigbabu, Olasimbo Ayodeji , *SVM–ELM: Pruning of Extreme Learning Machine with Support Vector Machines for Regression* , Journal of Intelligent Systems, Vol. 25 No. 4, 2016, pp. 555-566.
[12] R.R. Vargas, B. Rene, C. Bedregal, E.S. Palmeira , *A Comparison between KMeans, FCM and CKmeans Algorithms* , Theoretical Computer Science, Vol 7 No.3, 2011, pp. 15-28.
[13] Zhou, F.Y., Jin, L.P. and Dong , *A Review of Convolutional Neural Networks. Journal of Computers* , World Journal of Engineering and Technology, Vol.9 No.4, 2021, pp. 1229-1251.
[14] ZhiFei Lai, HuiFang Deng , *Medical Image Classification Based on Deep Features Extracted by Deep Model and Statistic Feature Fusion with Multilayer Perceptron* , Computational Intelligence and Neuroscience, Vol. 2018 No.13, 2018, pp. 1-13.
[15] Zinkevich, M, Weimer, M., Li, L. Smola , *Parallelized stochastic gradient descent. In Advances in Neural Information Processing Systems* , Neural Information Processing Systems Foundation, 2010, Vol. 23, No.23, pp. 2595–2603.
[16] Luo, P, Li, H.F , *Research on Quantum Neural Network and its Applications Based on Tanh Activation Function* , Computer and Digital Engineering, Vol. 16, No.13, March 2012, pp.33-39.
[17] Günnemann, N, Pfeffer, J , *Predicting Defective Engines using Convolutional Neural Networks on Temporal Vibration Signals* , In Proceedings of the First International Workshop on Learning with Imbalanced Domains: Theory and Applications, Vol. 74, 2017, pp. 92–102..
[18] Shi, X.B.; Fang, X.J.; Zhang, D.Y.; Guo, Z.Q , *Image Classification Based on Mixed Deep Learning Model Transfer Learning*, Journal of System Simulation, 2016, 28, pp.167–173.
[19] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio , *Maxout networks* , In Proceedings of the 30th International Conference on Machine Learning, 2013, Vol. 41, pp. 1319–1327. .
[20] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L.Fei , *Image-net large scale visual recognition challenge* , International Journal of Computer Vision, Vol. 115, No. 3, pp. 211–252, 2015.
[21] K. Nag and N. R. Pal , *A multi objective genetic programming-based ensemble for simultaneous feature selection and classification*, IEEE Transactions on Cybernetics, 2015, Vol. 46 No. 2, pp. 499–510.
[22] G. Xuan, *Apple Detection in Natural Environment Using Deep Learning Algorithms* , 2020, Vol. 8, pp. 216772-216780.
[23] Behera, Santi Kumari; Rath, Amiya Kumar; and Sethy, Prabira Kumar, *Fruit Recognition using Support Vector Machine based on Deep Features*, Karbala International Journal of Modern Science, 2020, Vol. 6 No. 2, pp. 235-245.
[24] J. Samuel Manoharan, , *Capsule Network Algorithm for Performance Optimization of Text Classification* , Journal of Soft Computing Paradigm (JSCP), 2020, Vol.03 No.01, p.p1-9.
[25] M. Pan, Y. Liu, J. Cao, Y. Li, C. Li and C. -H. Chen , *Visual Recognition Based on Deep Learning for Navigation Mark Classification*, in IEEE Access, 2020, vol. 8, pp. 32767-32775.
[26] Díaz-Pernas FJ, Martínez-Zarzuela M, Antón-Rodríguez M, González-Ortega D , *Deep Learning Approach for Brain Tumor Classification and Segmentation Using a Multiscale Convolutional Neural*, Network. Healthcare (Basel), 2021, vol.9 No.153, pp.1-14.
[27] Krizhevsky, A.; Sutskever, I.; Hinton, G.E., *ImageNet classification with deep convolutional neural networks. Comms*, ACM, 2017, vol.60, p.p 84–90.
[28] Chen, L.; Wu, C.; Fan, W.; Sun, J.; Naoi, S , *Adaptive Local Receptive Field Convolutional Neural Networks for Handwritten Chinese Character Recognition*, In Chinese Conference on Pattern Recognition, 2014, pp. 455–463.

Edited by: Vinoth Kumar
Received: Jun 30, 2022
Accepted: Dec 5, 2022