The Visual Computer
https://doi.org/10.1007/s00371-023-02784-3

ORIGINAL ARTICLE

# Enhanced image classification using edge CNN (E-CNN)

**Shaima Safa aldin[1] · Noor Baha Aldin[2] · Mahmut Aykaç[3]**

Accepted: 13 January 2023


**Abstract**
Recently, deep learning has become a hot topic in wide fields, especially in the computer vision that proved its efficiency in processing images. However, it tends to overfit or consumes a long learning time in many platforms. The causes behind these issues return to the huge number of learning parameters and lack or incorrect training samples. In this work, two levels of deep convolutional neural network (DCNN) are proposed for classifying the images. The first one is enhancing the training images with removing unnecessary details, and the second one is detecting the edges of the processed images for further reduction of learning time in the DCNN. The proposed work is inspired by the human eye's way in recognizing an object, where a piece of object can be helpful in the recognition and not necessarily the whole object or full colors. The goal is to speed up the learning process of CNN based on the preprocessed training samples that are precise and lighter to work well in real-time applications. The obtained results proved to be more significant for real-time classification as it reduced the learning process by (94%) in Animals10 dataset with a validation accuracy of (99.2%) in accordance with the classical DCNNs.

**Keywords** CNN · Classification · Training time · Edge

## 1 Introduction

The recent years has witnessed a rapid increase in the number of images; due to the development of Internet, social networks and the popularity of mobile devices capturing images with so ease [1]. Therefore, to manage these images, they should be first annotated and described with meaningful words. This step is known as image classification [2]. However, the large number of images/classes is a real challenge for classification, as it will deal with the precise computation of similarity between large-scale images and how to train the supervised classification models for newly emerging classes [3].

✉ Shaima Safa aldin
shaima.safaaldin@nahrainuniv.edu.iq

Noor Baha Aldin
noor.aldin@hku.edu.tr

Mahmut Aykaç
maykac@gantep.edu.tr

[1] Continuing Education Center, Al-Nahrain University, Baghdad, Iraq

[2] Department of Electrical and Electronics Engineering, Hasan Kalyoncu University, Gaziantep, Turkey

[3] Department of Electrical and Electronics Engineering, Gaziantep University, Gaziantep, Turkey

Recently, the researchers achieved outstanding results about the performance of image classification using DCNN based on the extracting step of convolutional features for obtaining the invariance of translations, rotations, and scale [4]. The development of artificial intelligence has been increased using the deep learning (DL) network in many fields of our daily lives such as processing the natural language [5], or medical fields like multimodal medical image fusion that combines two or more input images to produce an informative combination for facilitating the clinical diagnosis and surgical navigation [6], where these sources that formed the combination image may include visible and infrared inputs to define texture details in the visible image and salient target in the infrared image [7]. Also, the DL network has a great influence in the computer-vision field such as image dehazing, which removes haze from a picture to generate a new clear one [8], or supervised applications such as person re-identification that looks for the right walkers' images based on a given query walker's image across non-intersected area cameras administrated by human-annotated labels or unsupervised re-identification system that learns distinctive characteristics without human-annotated labels [9].

Deep learning, as a bottom of line, has verified its highest efficiency in image processing [10]. However, learning the images' features in CNNs with enormous factors could lead

Published online: 04 February 2023

to an overfitting case with so ease, which in turn minimizes the ability of generalization and thus poor performance. Also, the training step often relies on many resources such as memory and computational power that may consume a lot of time [11]. Therefore, seeking a rapid training method and improving the generalization performance became urgent demands. The late researches focus on reducing the complexity of these models like dropout and batch normalization or improving their generalization performance using adequate training samples; which are usually difficult to satisfy [12, 13]. Therefore, different enhancement methods have been taken into consideration for generating more training samples than the original ones, such as rotation, translation, cropping, and flipping. The last two methods (cropping and flipping) were proved in some researches to be good choices for enhancing the performance in deep CNNs like mixup [14], cutMix [15], and RandomErasing [16].

For DNNs, increasing the input images leads to an increase in the model; which in turn requires larger datasets and enormous parameters. Therefore, we have introduced the two-levels DCNN for classifying images that "firstly prepares and defines the sub-sections by extracting the classified object and enhancing it with reducing the unnecessary details such as background or other unrelated objects" and "secondly processes the images to produce Edgy samples by detecting the edges of the extracted object for the training step in the proposed DCNN".

The resulted model becomes a pretrained-model of specified parameters useful for real-time applications based on the preprocessed dataset. This method works well for high-resolution and large-scale image classification that struggles mostly from hardware barriers and computation time. At first, our system works on preprocessing the input images using a new splitting technique that separates the center object from the barrier of a single image to for automatic fine touch-ups and be prepared to the next step (GrapCut) process without any user interference. Moreover, a new suggested rescaling and centering method has been used after cropping; which is helpful in keeping the objects not distorted inside the images.

Training the system model depends primarily on new sketch/edgy images and not raw images like the usual; such a thing helps in further reduction in training time with a high precision for the given classes. The proposed network is based on a modified SqueezeNet, where the original parts are used for feature extraction and the new proposed section is used for a large-scale recognition that increases the recognition's precision. A new method that computes the batch-size was suggested for each give dataset based on structural similarity index (SSIM) and complex wavelet structural similarity (CW-SSIM); in order to reduce the number of trials in specifying the optimum batch-size. Also, defining the quantity ratio of used images (preprocessed color and sketch/edgy augmented images) helps in minimizing the training time and increasing the validation accuracy.

## 2 Related works

The object behind the usage of data enhancement is obtaining more distinct features or enlarging the training set in DCNNs [10]. Therefore, it can be regarded as a form of regularization, that is not only achieved by reforming the model of DCNN but by generating more efficient input set [11]. The rotation, translation, flipping, and random cropping are forms of data enhancements [12, 13]. Blending two randomly chosen images with their labels is what the Mixup method did for regularizing the DCNNs [14]. The Cutout method [15] masks out the square sections of input set during training, while randomly choosing a rectangular region is what the RandomErasing method did for replacing the corresponding pixels with their mean or random values [16].

In [9], a novel hybrid contrastive model (HCM) was introduced to perform an image-level contrastive learning for unsupervised person Re-ID that sufficiently explores features' similarities within hard sample pairs. The authors in [6] introduced an unsupervised multiscale adaptive transformer to fuse multimodal medical images for best utilization of global information in both superficial and thorough layers. The YDTR model [7] is designed to acquire the distinct context information as well as the local features by combining the thermal targets from infrared pictures and visible pictures for enhancing the target saliency with keeping the significant details to produce a clear visible picture. A fully end-to-end dehazing method was presented in [8], which restores a coarse clean image using a dilated CNN (not a standard one) of high-level feature maps to acquire more context information using a dilation convolutional layer that increases the receptive size for reconstructing a final dehazed picture.

However, the mentioned methods deal or clarify/enhance all the abundant texture scene details, while our proposed model is based on a supervised standard CNN that focuses on local features to acquire only the significant target details of a sketch/edgy image that part of it (region of interest) is enhanced and the other part is blurred to get-rid of any unwanted information for time and overhead reduction.

Usually, the semantics conveyed of various labels is correlated in a multilabel case; which is totally different from a single-label case. For example, modeling the topics and backgrounds of an animal image is correlated specifically, when this animal only lives in a specific environment. Therefore, the mutual structures of various labels tend repeatedly to involve the correspondence characteristics and coherence of object class [17]; which helps in separating the different objects of multiclass images, where the shared structures may

include multiple objects of different scale, poses and that represents a challenging case but common in visual computers [18].

Predicting the scene location represents an eye-fixation prediction in a saliency detection, where a human observer may fixate and detect visually the remarkable objects/regions. The initial research of this topic has focused on detecting a salient object within an image [19], and then it dealt with the detection of RGB-D saliency that takes into consideration the information-depth. Over the few past years, many researchers have dealt with this topic; but majority of them are based on hand-crafted characteristics [20].

The mentioned methods did not get rid of the unwanted details but enhanced the training images; which in turn produces massive data compared to the original for training parameters in deep-neural nets. Rother et al. have introduced a segmentation method based on a selected bounding box that is computed using an iteration between the defined foreground/background color models [21], while an instance-level of segmentation method has been introduced by Liang et al. that takes a proposal initial object and several CNN features maps to apply them iteratively for refining the segmentation [22]. These methods each compute their segmentation using a manual definition of bounding box and not an auto-definition.

Also, an Inception v3 model has been proposed by Macoadha to fix the issues of the previous method iNaturalist that reached a 0.4 validation error when trained over 75 epochs. Such a low error was achieved by utilizing a pretrained InceptionNet and finetunes the hyper-parameters carefully [23]. The authors in [24] have proved that preprocessing the dataset before classification was helpful in improving the accuracy using zero component analysis (ZCA). However, their technique has improved the accuracy from (43–50)% to (67–69)%. Also, in [25], the authors have worked on improving image quality for identifying its class, but for a small image of $32 \times 32$ size. Therefore, the above-mentioned researches have focused on improving the training accuracy and did not focus on how to lower the training time of large-scale images, the number of epochs and recommended batch-size to be well-suited for real-time tasks.

## 3 Proposed method

This section presents the proposed E-CNN in detail:

### 3.1 Preprocessing the dataset

#### 3.1.1 Sub-sections construction

In this section, we will demonstrate the key-steps of preparing a dataset as an input to our system model. The preprocessing stage includes three primary steps (splitting, GrabCut, and cropping). The reason behind these steps is to get rid of the unwanted/unnecessary spaces without distorting the image, while focusing on the goal object. Before enhancing the section containing the classified object, the image will be first segmented into 2 parts. The first one defines the barrier or the sure background section (outer window), and the other one contains the classified object (inner window). The reason behind this segmentation is to enhance the details of the inner window with blurring the other one (outer window). The resulted output is helpful for the second step that removes the background using the GrabCut method automatically without the interference of the human. Where it finds the first contour after applying the auto-specified window mask, in order to remove the other unwanted spaces closer to the required object where the removed data from the images are the unwanted objects distracting the model from its goal object that needs to be classified specifically.

The barrier section (outer window) is defined based on the dimension of each input image. Everything inside this section will be regarded as a background. The center of each input image is calculated for defining the center coordinates of the ellipse window that represents the mask, where its major and minor axes are computed as shown in Eqs. (1 and 2):

$$|M_{\mathrm{j}}| = \begin{cases} (3 \times h)/4, & h > w \\ (3 \times w)/4, & w > h \end{cases} \tag{1}$$

$$|M_{\mathrm{n}}| = \begin{cases} (3 \times h)/5, & h > w \\ (3 \times w)/5, & w > h \end{cases} \tag{2}$$

where $M_{\mathrm{j}}$ represents the major axis and $M_{\mathrm{n}}$ represents the minor axis. The height and width are represented by $h$ and $w$. Anything outside this ellipse is blurred, and any object inside it will be applied to edge and details enhancement filters. We call this step as an auto-specified bounding window for the image split using the GrabCut technique.

The GrabCut estimates the color distribution of both background and target object by a Gaussian-mixture model (GM) for extracting that object with minimum user interaction using a constructed graph that distinguish the foreground and background pixels based on the cuts of minimal costs as shown in Fig. 1. This process is done through iterated grab cuts on the graph of the image to fully define all pixels with a label since some of them will initially receive an unknown label. The previous steps are repeated for further reduction of the background/unwanted objects and refining the segmentation until the classification converges, as shown in Fig. 2:

The last step (cropping) applies a binary threshold to the pixels of resulted image. The pixel's value will be 255, if its original value is greater than the threshold; otherwise, it will

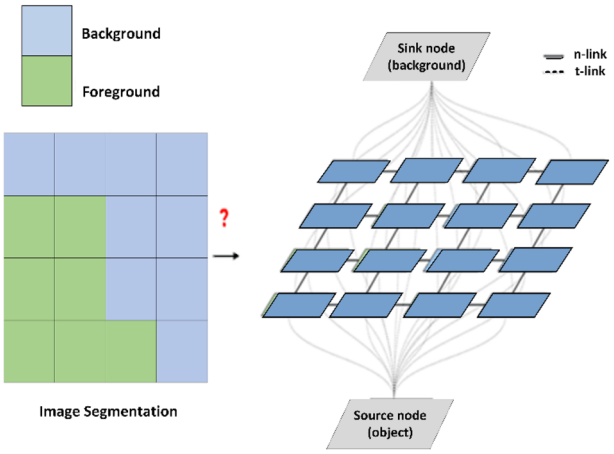



**Fig. 1** Graph model of image pixels [26]

**Fig. 2** The refined image segmentation

be 0, as shown in Eq. (3):

$$\text{Binary dst}(a,\, b) = \begin{cases} \text{max - value if src}(a,\, b) > T \\ 0 \text{ otherwise} \end{cases} \tag{3}$$

where $\boldsymbol{T}$ represents a threshold calculated individually for each pixel. Thresholding is a widely known segmentation method for separating an object with regards to its background. The next step is finding the contours in the image after a grayscale conversion and binarization. Contours are typically used to find a white object from a black background for detecting the boundary edges of the wanted blocks using the binary threshold. If the object of interest is considered white, then the binarization formula is determined by Eq. (4), and if it is considered black, then the binarization formula is determined by Eq. (5):

$$I_{\text{bin}}(a,\, b) = \begin{cases} 0 \text{ if} I_{\text{src}}(a,\, b) < T \\ 1 \quad \text{otherwise} \end{cases} \tag{4}$$

$$I_{\text{bin}}(a,\, b) = \begin{cases} 1 \; if \, I_{\text{src}}(a,\, b) < T \\ 0 \quad \text{otherwise} \end{cases} \tag{5}$$

The dimension of the first contours is extracted to define the cropped image precisely. This process helps in removing the unwanted objects/spaces for the reduction of unnecessary details, assuming that the classified object is centered or near the center point (away from the edges). Then, the resulted image is augmented randomly (flipping or rotation) with keeping its color-space. Random flipping generates a reflection of image corresponding to the one chosen axis (or more), while the rotation rotates a picture around the center by an angle $\theta$ that followed by a suitable interpolation to fit with the size of original image. It is denoted by $R$ and often paired with zero-padding putted on the missing pixels, as shown in the following equation:

$$R = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & -\cos\theta \end{pmatrix} \tag{6}$$

### 3.1.2 Sketch/edge conversion

In this section, the image is converted into a sketch type for defining its edges. The reason behind choosing a sketch process rather than other edge detection methods (like Canny, Sobel…etc.) is that it gives clearer view and higher precision. The images' dataset is divided into 2 divisions.

One-third of the dataset are applied only to the steps illustrated in the former section, while the remaining images continue with the processing steps described in this section. The reason behind this division is that the colored part helped in increasing the validation accuracy, while the remaining sketchy/edgy images helped in reducing the training time. The color space of the two-thirds dataset is converted to the Gray scale and become a negative one using the not bitwise operation for extracting the distinct parts of image. The negative operation is done by inverting the grayscale image's pixels (0–255) as follows:

$$N(i,\, j) = 255 - P(i,\, j) \tag{7}$$

The inverted image is then applied to a Gaussian-blur filter for smoothing the image and reducing its noise which uses a Gaussian distribution (GD) function for computing the transformation to be applied for each pixel. In 2D systems, it is the product of two GD functions, one for each dimension as shown in Eq. (8).

$$G(i,\, j) = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}} \tag{8}$$

where $x$ and $y$ define the length from the origin in the horizontal and vertical axis, respectively. $\sigma$ represents the standard deviation of GD. This formula, in 2D systems, generates a surface that consists of contours that are concentric circles with a GD from the origin point. The blurred image is then

normalized using a color dodging that uses a bitwise division in regard to the original gray scale image, as shown in Eq. (9). Dealing with the image's pixels as they are (0–255) leads to a high computation and more complexity in DCNN. Thus, normalizing them ranging from 0 to 1 simplifies the complexity and speeding-up the computation.

$$\text{divide}(X,\ Y) = \begin{bmatrix} x_{00}/y_{00} & x_{01}/y_{01} \\ x_{10}/y_{10} & x_{11}/y_{11} \end{bmatrix} \tag{9}$$

where $X$ and $Y$ are matrices computed as below:

$$X = \begin{bmatrix} x_{00} & x_{01} \\ x_{10} & x_{11} \end{bmatrix},\ Y = \begin{bmatrix} y_{00} & y_{01} \\ y_{10} & y_{11} \end{bmatrix} \tag{10}$$

The color dodge combines the foreground and background by normalizing around 255 and subtracting the background pixels, where the bottom layer is divided by the inverted top layer. The value of top layer controls the lightening of bottom layer, which results in a blurred image highlighting the boldest edges. Implementing the Dodging of a colorful image '$A$' with a mask '$B$' is as shown in Eq. (11):

$$C[i] = \begin{cases} 255, & B[i] > 255 \\ \min\left(255,\ \frac{A[i]<<8}{255-B[i]}\right), & B[i] \leq 255 \end{cases} \tag{11}$$

where $C[i]$ is the dodging result, $A[i]$ is the pixel after decolorization, and $B[i]$ is the pixel after Gaussian blur filter.

It divides the grayscale value of an image pixel $A[i]$ by the inverse of the mask-pixel $B[i]$ with assuring that the resultant pixel is in the range [0,255] and not divided by zero. A darkening step is then applied by creating new pixels retaining the smallest values of the foreground and background pixels as shown in Eq. (12):

$$D = \min(p1,\ p2) \tag{12}$$

where D defines resultant pixel after darkening, $p1$ represents the foreground pixel, and $p2$ represents the background pixel. Each output image is then resized to the desired size with keeping the scale by centering the image for preserving the objects' shape without deformation as shown in Eq. (13):

$$|S| = \begin{cases} \frac{x}{h},\ h > w \\ \frac{x}{w},\ w \geq h \end{cases} \tag{13}$$

where $S$ defines the scale of resizing, $x$ is the size of original image, and $h$ and $w$ are the image's height and width, respectively.

$$|C| = \begin{cases} B[0 : h_2 \times s,\ T : T + w_2] + A_2, & w > h \\ B[T : T + h_2,\ 0 : w_2] + A_2, & w < h \\ B[0 : h_2,\ 0 : w_2] + A_2, & w = h \end{cases} \tag{14}$$

where $C$ defines the centered image, $A_2$ represents the resized image, $B$ represents the black image, $h2 = h \times s$, which defines the new height after scaling by the factor s, $w2 = w \times s$, which defines the new width after scaling, and $T$ represents the starting corner that is calculated as follows:

$$|T| = \begin{cases} (x - w_2)/2,\ h > w \\ (x - h_2)/2,\ w > h \end{cases} \tag{15}$$

The resulted output is a cropped edgy image, as shown in Fig. 3.

### 3.2 Edge CNN (E-CNN)

Figure 4 illustrates the E-CNN architecture. An E-CNN is built by a linear stacking of the hidden and output layers for classifying the image. The network takes as input the processed datasets that has two-third of sketchy/edgy images and one-third of colored-augmented images.

The suggested model includes two sections (feature extraction based on a SqueezeNet and precise large-scale recognition), as shown in Fig. 4. The DCNN consists of a SqueezeNet that reduces the number of parameters using fire modules of 1 × 1 convolutions. Choosing the initial feature extraction layers is crucial for compromising between speed and precision. Therefore, our system model is based on a SqueezeNet model due to its impressive accuracy and lightweight as the number of parameters is reduced using the fire modules.

#### 3.2.1 SqueezeNet feature extraction section

As the SqueezeNet network compresses the parameters to about 1/50 of AlexNet [26] with ensuring a precise recognition. It has been chosen for shortening the extraction and speeding up the detection, as it replaces the standard 3 × 3 convolution layers by fire modules that each consist of a squeeze (1 × 1) and expand (1 × 1 and 3 × 3) layers, followed by an activation layer (rectified linear unit/ReLU) for enhancing the depth of network.

The precision is improved using a 3 × 3 convolution kernel, while the 1 × 1 convolution kernel reduces the weight parameters. The original structure of SqueezeNet has 1 convolutional layer, 3 MaxPooling with a stride of 2, 9 fire modules, ends with a conv layer (1 × 1), activation (ReLU) and a global average pooling. The input-image size is 227

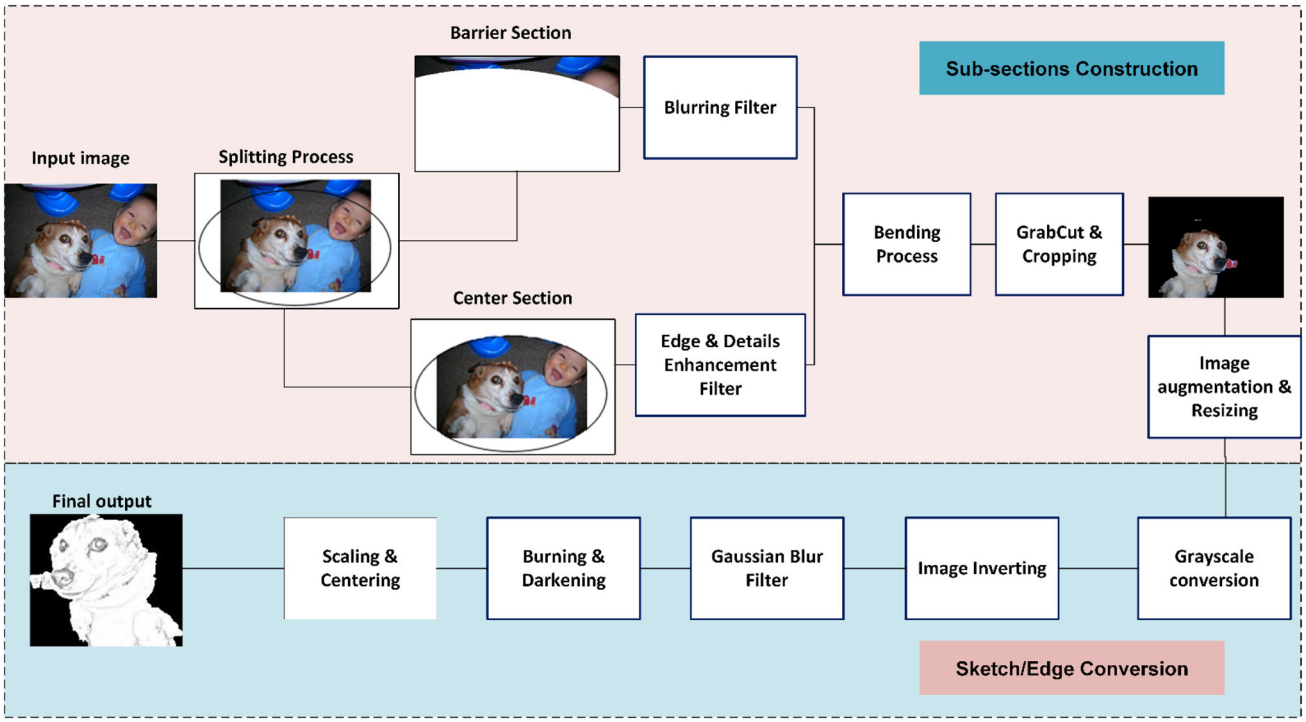



**Fig. 3** Sketch/edge conversion process

$\times$ 227 $\times$ 3, where the usage of 3 $\times$ 3 MaxPooling and 1 $\times$ 1 convolution has reduced the feature-maps' size to the half with less loss in information. As a result, this section can retain the information while improving the speed of feature extraction.

### 3.2.2 Large-scale recognition section

The overfit issue is what many neural nets suffer when training on small datasets and thus produces inaccurate precision for the new data. Therefore, the precise large-scale recognition network was designed for minimizing the training time with pertaining a high precision and avoiding the overfitting. The proposed section has replaced the last conv layer (1 $\times$ 1) and activation from the original SqueezeNet, by a MaxPool2D, batch normalization, dropout (0.5), followed by a 3 $\times$ 3 convolutional layer and then dropout (0.2), global average pooling and softmax activation layer.

The MaxPool2D is used for reducing the dimensions of output feature map from the previous 5 layers, which in turn helps in preserving the important data of input image and in turn reducing the computation time. Batch normalization normalizes the output of the previous layers and allows each layer to work more independently, which in turn becomes more efficient against the overfitting. Although trying the average of many various parameters for generalization could be theoretically a best solution for training a model, it consumes a lot of resources and time.

Therefore, the dropout was introduced to get around this, where some nodes either from hidden or input layers are dropped. A value of 0.5 was chosen as it gave the best results for our experimental datasets. Then, a 3 $\times$ 3 conv kernel was used for ensuring the network precision and another dropout layer was used of value (0.2). The global average pooling layer is used for down sampling and keeping only the remarkable details with minimizing the amount of computations as it reduces each channel in the feature map to a single value. Thus, $na \times nb \times nc$ feature map is reduced to $1 \times 1 \times nd$ feature map.

The softmax layer is utilized right before the output layer for multiclass classification. It gives the probability of an input image belongs to a specific category. An Adam optimizer was used due to its efficiency and less-memory requirement. For a multiclass classification system, we have used a categorical cross-entropy for loss function.

### 3.2.3 Network parameters

*Batch size*: Choosing the best batch size is sometimes a time wasting. Therefore, to find out the optimum batch size for our network, a new method has been proposed based on the SSIM and CW-SSIM in accordance with the first image. The first image is chosen based on its shape and characteristics to be the ideal one representing its class. The reason behind this is to find the relation between the batch-size and similarity indexes as represented by the flowchart shown in Fig. 5,

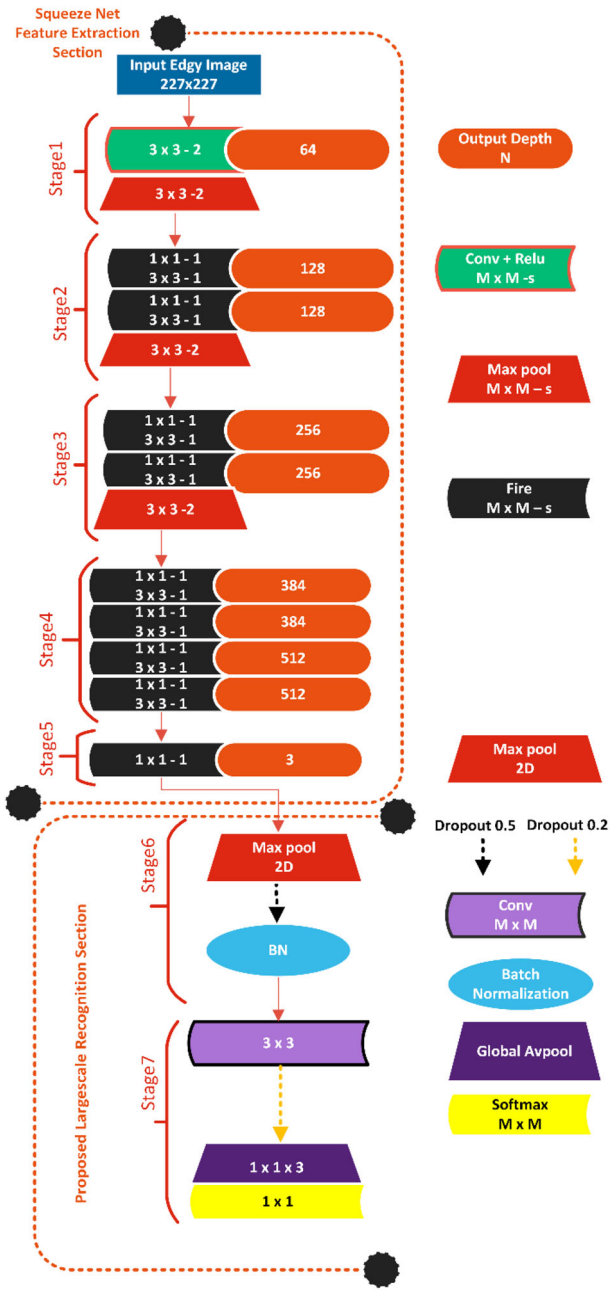



**Fig. 4** The network system model

where the SSIM index involves the image regression as a perceived alteration in structural information, regarding the key perceptual phenomena as contrast and luminance masking, while the CW-SSIM deals with image scaling, translation, and rotation. Thus, this process reduces the number of trials for finding the optimum batch size.

$$\mathrm{SSIM}(a,\ b) = \frac{(2\mu_a\mu_b + \mathrm{con}_1)(2\sigma_{ab} + \mathrm{con}_2)}{\left(\mu_a^2 + \mu_b^2 + \mathrm{con}_1\right)\left(\sigma_a^2 + \sigma_b^2 + \mathrm{con}_2\right)} \tag{16}$$

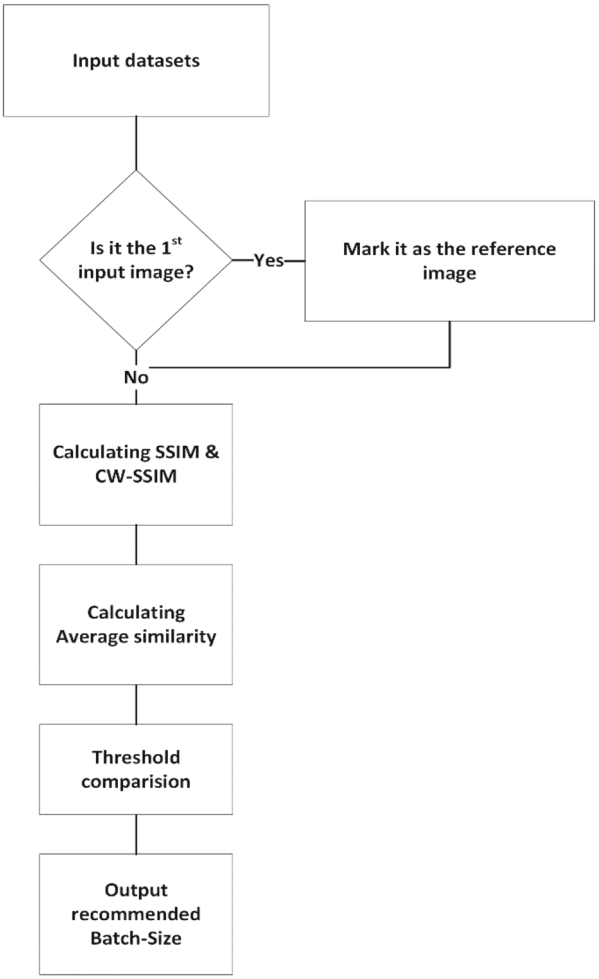



**Fig. 5** Computation flowchart of batch-size

where $\sigma_{ab}$, $\sigma_a^2$, $\sigma_b^2$, $\mu_a$, $\mu_b$, $\mathrm{con}_1$, $\mathrm{con}_2$, define the covariance of $a$ and $b$, the variance of $a$ and variance of $b$, the average of $a$ and average of $b$, respectively. The constants con1 and con2 are calculated using the following equations:

$$\mathrm{con}_1 = (m_1 R)^2,\ \mathrm{con}_2 = (m_2 R)^2 \tag{17}$$

where $m_1 = 0.01$, $m_2 = 0.03$, and $R$ defines the dynamic range of pixels ($2^{\#\mathrm{bits/pixel}} - 1$).

$$\mathrm{CW\text{-}SSIM}(a_i, a_j) = \left(\frac{2\sum_{x=1}^{K}|a_{i,x}||a_{j,x}| + N}{\sum_{x=1}^{K}|a_{i,x}|^2 + \sum_{x=1}^{K}|a_{j,x}|^2 + N}\right) \left(\frac{2\left|\sum_{x=1}^{K} a_{i,x}a_{j,x}^*\right| + N}{2\sum_{x=1}^{K}\left|a_{i,x}a_{j,x}^*\right| + N}\right) \tag{18}$$

where $a_i$ is the complex-wavelet transformation of signal i and $a_j$ is the complex-wavelet transformation of signal $j$. In addition, $N$ is a small positive number that ideally equal to

**Table 1** Training datasets

| Dataset | Training images | Test images | Class labels |
|---|---|---|---|
| Flowers | 3,178 | 1,362 | 5 |
| Natural image | 4,828 | 2,071 | 8 |
| Imagenette2 | 8,820 | 3,783 | 10 |
| Imagenette160 | 9,374 | 4,020 | 10 |
| Cats-v-Dogs | 19,080 | 8,177 | 2 |
| Animals10 | 20,730 | 8,884 | 10 |
| Pascal-2007 | 6,974 | 2,989 | 20 |
| MIRFlickr25K | 17,500 | 7,500 | 10 |

zero, given for stabilizing the function. The proposed similarity index is calculated as follows:

$$\mathrm{SIM}_{\mathrm{index}} = \mathrm{SSIM} \times 0.3 + \mathrm{CW} - \mathrm{SSIM} \times 0.7 \tag{19}$$

where the $\mathrm{SIM}_{\mathrm{index}}$ is used for determining the recommended batch-size, when compared with the following thresholds (2, 4, 8, 16, 32, 64). For example, if $16 < \mathrm{SIM}_{\mathrm{index}} < 32$, then the recommended batch size is 16. These thresholds were chosen according to the used experimental datasets.

*Number of epochs*: According to our training edge images and batch size, we found that it is enough to train our network for no more than 5–6 epochs. It converges fast in accordance with the normal subset of training images, which in turn helps in reducing the computation time. Also, we have defined the early stop and learning rate of reduction to stop the learning process at the best epoch.

## 4 Experimental results

This section presents the experimental results of the proposed model for showing its robustness in multilabel classification with reduced training time. We have tested the proposed model on two widely known large-scale datasets: PASCAL-VOC2007 [28] and MIRFlickr25k [29], with other datasets downloaded from the Kaggle website, as shown in Table 1.

Pascal-voc2007 has a total of 9963 images of 20 labels (Aeroplan, bird, bicycle, bottle, boat, bus, cat, car, chair, cow, dog, dining-table, horse, motorbike, potted-plant, person, sheep sofa, train, tv-monitor).

The MIRFlickr25k dataset includes 25,000 pictures in the official-site FLickr for recognizing objects, interiors, and scene categories. However, these tags frequently lack reliability for multilabel classification, since users make annotation to those images several times. Therefore, we have made 10 unique labels based on the center for classifying this dataset (Animals, Food, Indoor, People, Plant-life, Portrait, Sky, Structures, Transport, Water). Animals10 [30] has 10 classes (butterfly, cat, cow, dog, elephant, hen, horse, sheep, spider, squirrel). Flowers [31] has 5 classes (daisy, dandelion, roses, sunflowers, tulips). Natural Images [32] has 8 classes (airplane, dog, car, at, flower, motorbike, fruit, person). Imagenette2 [33] and Imagenette160 [34] have 10 classes (church, dog, fish, fuel_meter, garbage_truck, golf_ball, parachute, radio, saw, trumpet). Cats-v-Dogs [35] has 2 classes (Cats and Dogs).

The large-scale images of decent resolution ($227 \times 227 \times 3$) represent a challenge for us as its processing requires a massive computational power. Such a thing is too much for a normal laptop, which is a downside. However, converting these images using the proposed technique helped in passing these struggles and made it functional even for a normal laptop; as it achieved the best performance compared with different state-of-art models.

### 4.1 Implementation details

Our approach was implemented using a Jupyter Notebook with Keras, TensorFlow and OpenCV libraries. The platform has been implemented inside a Windows 10–64 bit of Processor Intel(R) Core (TM) i7-8565U CPU @ 1.80 GHz with Installed RAM16.0 GB.

### 4.2 Evaluation metrics

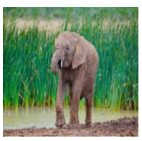
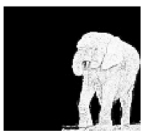
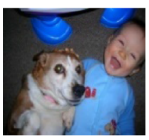
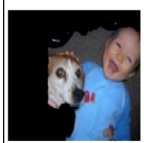
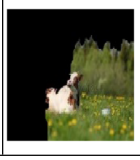
The label predictions are generated and compared with their ground-truth labels for each image within a dataset using our proposed system model. For each category, the value of average precision (AP), mean-average precision (mAP), precision, and recall are determined with showing the performance of the classification model using the receiver operating characteristic (ROC) curve at all thresholds. The division of #correctly annotated categories by #predicted categories defines the precision score, while the recall is determined by the division of #correctly annotated categories by the #ground-truth categories.

Also, the reduction in time has also been considered in our evaluations to prove the robustness of our model for real-time platforms. The training accuracy and loss has been computed for widely known datasets PASCAL-VOC207 and MirFlicker. The other implemented datasets as well as the PASCAL and MirFlicker are used to check the reduction in time per epoch and computational complexity. The preprocessed input images corresponding with the original GrabCut algorithm are shown in Table 2.

Figure 6 shows the real-time captured images (toys), their predicted categories and confidence score using our proposed model (under different light and pose conditions).


**Fig. 6** Predicted image's type and confidence score (under different poses and light conditions)

**Table 2** Preprocessing the input dataset

| Input | Original GrabCut | (Proposed) construct | Final (Proposed) |
|---|---|---|---|
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

The system model has also been tested under different conditions of (noise, pixelized, blurred and sketch) images, as shown in Fig. 7.

## 4.3 Image classification results

### 4.3.1 Mean average precision (mAP) and receiver operating characteristic (ROC)

The performance of localization and object detection method is evaluated by AP and mAP metrics that score the comparison between the ground-truth bounding box with the detected box. The model is more precise in detecting objects when its score is higher. We have compared our proposed model with the following high-performance models: CNN–SVM [36], RLSD [37], CNN–RNN [38], DenseNet [39], ReseNet101 [40] in which they achieve a mAP of (73.9, 84.0, 88.5, 89.9, 89.6)% consequentially, while our proposed model has 97.2% as shown in Fig. 8.

Figure 9 shows average precision-recall/receiver operating characteristic (ROC) curves of PASCAL-VOC2007 using

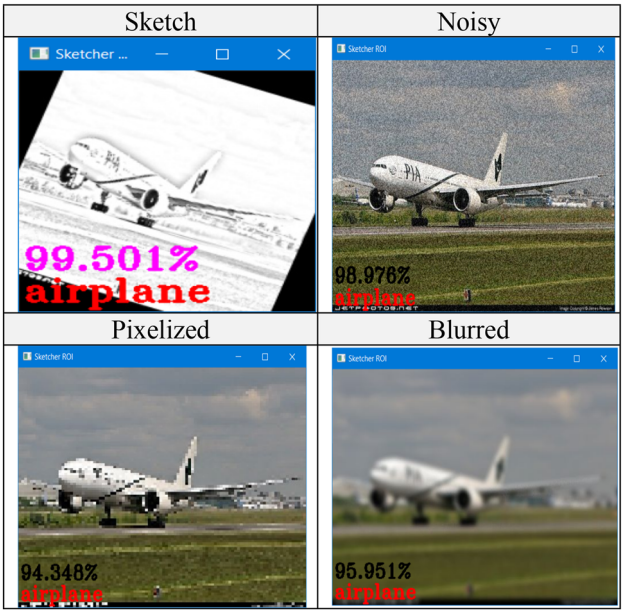



**Fig. 7** Predicted image and confidence score (under noise, pixelized, blur and sketch conditions)

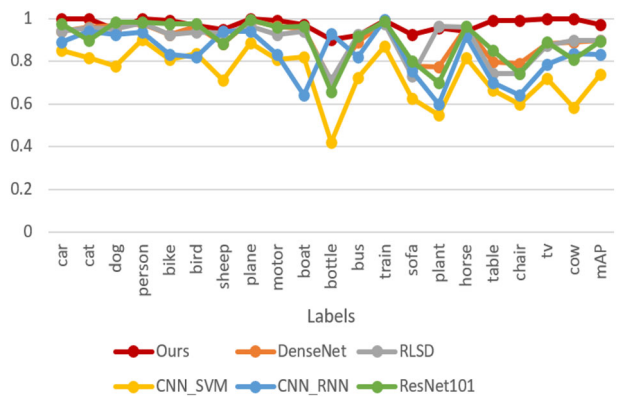



**Fig. 8** AP and mAP scores on the PASCAL VOC-2007

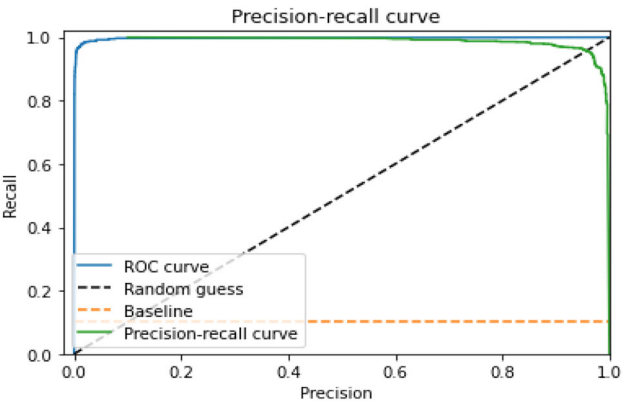



**Fig. 9** Precision vs. recall, and ROC curve in PASCAL-VOC2007

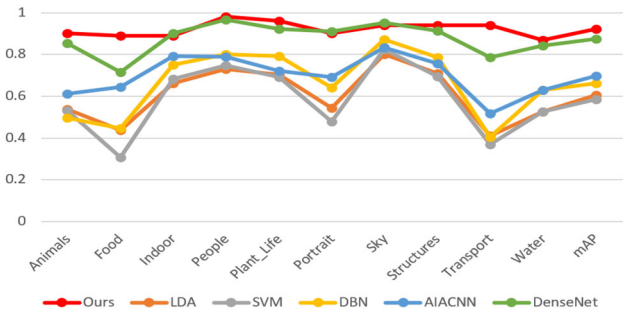



**Fig. 10** AP and mAP scores on the MIRFlickr25k

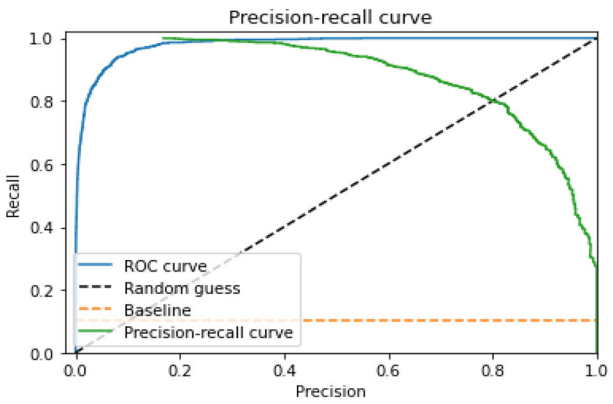



**Fig. 11** Precision vs. recall, and ROC curve in MirFlicker

the proposed model; where it a big under-curve area. Generally, the higher the area under the precision–recall curve (AUC-PR) curve, the better the classifier performs for the given task, where it reaches to (98%) in our model. Moreover, the precision is correspondingly high at thresholds of low recall, while at very high recall, the precision begins to drop. As a result, the proposed model has achieved higher predictions than the "baseline".

The obtained results shown in Fig. 10 are for the MirFlicker dataset, where the state-of-art methods (LDA, CNN_SVM, DBN [37], AIACNN [38], DenseNet [35] have achieved a mAP of (60.61, 58.55, 66.23, 69.86, 87.58) % in consequence; while our proposed model has reached to 92.12% higher than them. Figure 11 shows the average precision–recall/receiver operating characteristic (ROC) curves of MirFlicker dataset, where its AUC-PR score has reached 95%.

### 4.3.2 Time reduction and computational complexity

The computational reduction in training time has been calculated using the following formula

$$\text{Time}_{\text{reduction}} = \frac{\text{Time}_{\text{original}} - \text{Time}_{\text{proposed}}}{\text{Time}_{\text{original}}} \times 100\% \qquad (20)$$

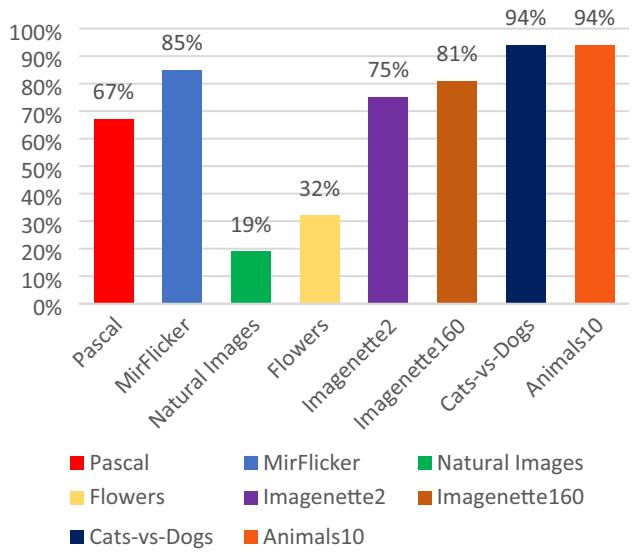



**Fig. 12** Time reduction (%) per epoch

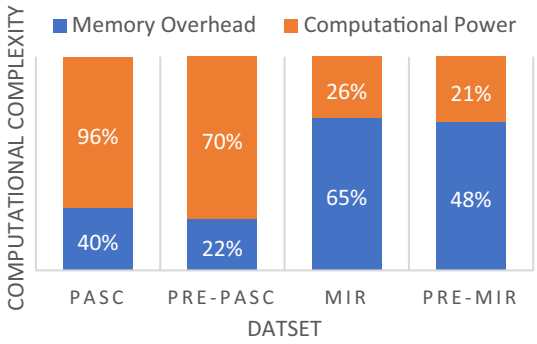



**Fig. 14** Computational complexity of PASCAL and MirFlicker

Referring to the chart shown in Fig. 12, it can be noticed that the maximum reduction in training time has reached 94% as shown in Animals10 dataset, while the minimum reduction in time has reached to 19% as shown in natural images dataset, in accordance with the original dataset. Therefore, the larger the dataset gives better reduction in time to work very well in real-time tasks.

Figure 13 shows the average training time per epoch for each dataset. It is obvious that the average training time has been reduced per epoch using the proposed model versus the original model. For example, the average time in Cats-vs-Dogs in original dataset and model is equal to 4354 s, while it is equal to 261 s.

Figure 14 shows the average computational complexity in terms of memory and power using the original SqueezeNet model for original PASCAL and MirFlicker datasets (named as PASC and MIR for short), compared with the preprocessed dataset using the proposed model. As shown in the figure, the preprocessed dataset using our proposed algorithm has a less computational complexity than the original datasets. Referring to Fig. 15, the proposed model has a high train accuracy/validation accuracy and low loss/validation loss, where the train accuracy of the model has reached 98.9% in natural-images dataset (for example) and the validation

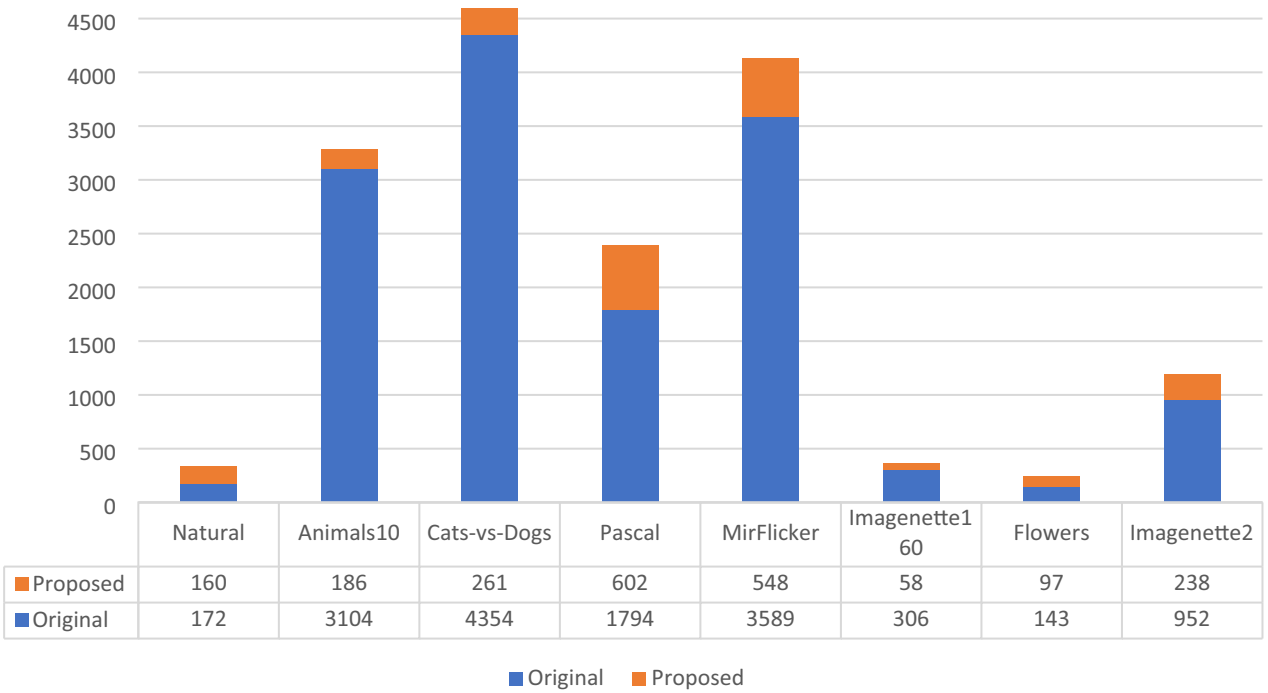



**Fig. 13** Average training time (sec) per epoch

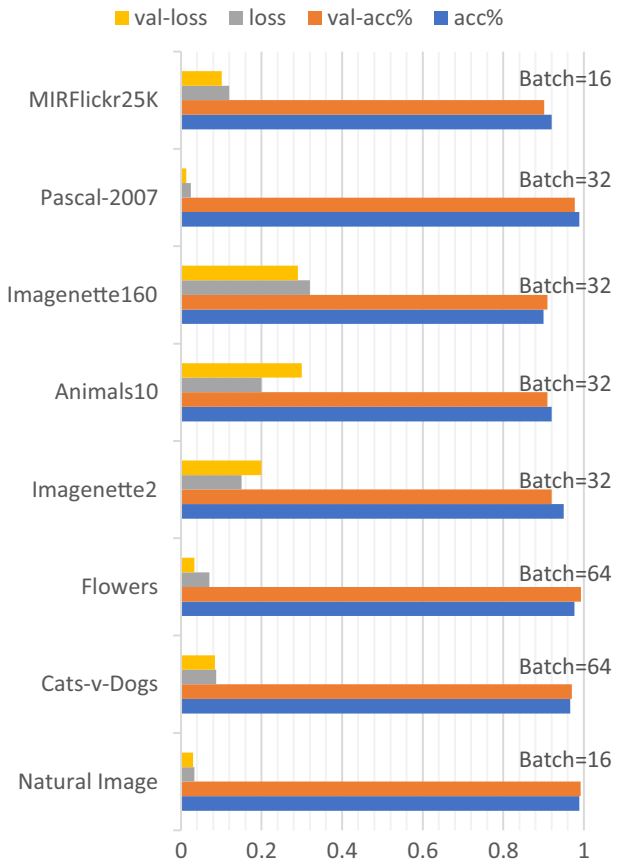



**Fig. 15** Train and validation accuracies with their losses

accuracy 99.2% with (0.033) and (0.03) for the loss and validation-loss consequentially.

## 5 Conclusions

The image classification in deep learning is widely used for recognizing the input datasets to their mutually exclusive classes using neural nets. DCNN model consists of multiple hidden layers for extracting the features and output layers for classifying the inputs. Although the DCNN proved its capability in such a field, it suffers from overfitting, low performance, and long training time, especially for high-quality/large datasets.

Therefore, in this paper, we have reduced the training time with improving the image classification using edge DCNN (E-CNN). This is extremely important for working with large image dataset and large-scale neural networks. The goal was to construct a pretrained DCNN of specified parameters that trained on the preprocessed image dataset to work fast in real-time applications. Training the model using a preprocessed image dataset has improved the overall system, where the raw images have many unnecessary details that distracted the system attention from the goal object within an image. The proposed model has reached a maximum training accuracy of 98.9% and 99.2% validation accuracy with a loss and validation loss of 0.033 and 0.03, respectively. Also, the maximum reduction of training time per epoch in our datasets has reached 94% in accordance with the normal training set of images. This approach has approved its capability in minimizing the training time, especially in the large datasets.

The quality of the used dataset images, their quantity and light conditions affect the accuracy of the system model, which represents a system limitation. Also, for the preprocessing step, the goal object should be near the center to get precise results. However, the system has proved its robustness and reduction of training time with a high precision, especially for large-scale/large dataset. As a future plan, this system can be useful for computer vision such as real-time video conferencing for sign-language, text/object recognition, multilabel classification and unsupervised person re-identification.

**Funding** The authors have no financial or proprietary interests in any material discussed in this article.

## Declarations

**Conflict of interest** The authors did not receive support from any organization for the submitted work.

## References

1. Schmidhuber, J.: Deep learning in neural networks: an overview. Neural Netw. **61**, 85–117 (2015)
2. Coates, A., et al., An analysis of single-layer networks in unsupervised feature learning. In: Proceedings 14th International Conference on Artificial Intelligence and Statistics (AISTATS). 2011, Fort Lauderdale, FL, USA. Vol. 15 of JMLR: W&CP 15 (2011)
3. N. B. Aldin, et.al., Accuracy comparison of different batch size for a supervised machine learning task with image classification. In: 2022 9th International Conference on Electrical and Electronics Engineering (ICEEE), pp. 316–319 (2022). https://doi.org/10.1109/ICEEE55327.2022.9772551
4. Vesal, S., et.al, (2018). Classification of breast cancer histology images using transfer learning. In: ICIAR. Lecture Notes in Computer Science, vol. 10882. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93000-8_92
5. Collobert, R., et al.: Natural language processing (almost) from scratch. JMLR **12**(1), 2493–2537 (2011)
6. Tang, W., He, F., Liu, Y., Duan, Y.: MATR: multimodal medical image fusion via multiscale adaptive transformer. IEEE Trans. Image Process. **31**, 5134–5149 (2022). https://doi.org/10.1109/TIP.2022.3193288
7. Tang, W., He F., Liu, Y.: YDTR: infrared and visible image fusion via Y-shape dynamic transformer. In: IEEE Transactions on Multimedia, (2022). https://doi.org/10.1109/TMM.2022.3192661
8. Zhang, S., He, F.: DRCDN: learning deep residual convolutional dehazing networks. Vis Comput **36**, 1797–1808 (2020). https://doi.org/10.1007/s00371-019-01774-8

9. Si, T., He, F., Zhang, Z., Duan, Y., Hybrid contrastive learning for unsupervised person re-identification. In: IEEE Transactions on Multimedia. https://doi.org/10.1109/TMM.2022.3174414
10. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
11. Xie, S., et al.: Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE conference on CVPR. (2017)
12. Ioffe, S., et al.: Batch normalization: accelerating deep network training by reducing internal covariate shift, In: Proceedings of the 2015 International Conference on MLIS, Gangzhou, China (2015)
13. Srivastava, N., et al.: Dropout: a simple way to prevent neural networks from overfitting. JMLR **15**(1), 1929–1958 (2014)
14. Zhang, H., et al.: mixup: beyond empirical risk minimization. (2017). https://arxiv.org/abs/1710.09412
15. Yun, S., et al., Cutmix: regularization strategy to train strong classifiers with localizable features. (2019). https://arxiv.org/abs/1905.04899
16. Zhong, Z., et al.: Random erasing data augmentation. (2017). https://arxiv.org/abs/1708.04896
17. Yang, F., Li, B.: Unsupervised learning of spatial structures shared among images. Vis. Comput. **28**(2), 175–180 (2012)
18. Zhang, L., et al.: Extracting shared subspace incrementally for multilabel image classification. Vis. Comput. **30**, 1359–1371 (2014). https://doi.org/10.1007/s00371-013-0891-4
19. Huang, K., et al.: Image saliency detection via multi-scale iterative CNN. In: The Visual Computer, pp. 1–13 (2019)
20. Liu, Z., Xiang, Q., Tang, J., et al.: Robust salient object detection for RGB images. Vis. Comput. **36**, 1823–1835 (2020). https://doi.org/10.1007/s00371-019-01778-4
21. Carsten Rother, et al.: Grabcut: Interactive foreground extraction using iterated graph cuts. In: ACM TOG, vol. 23, pp. 309–314. ACM (2004)
22. Xiaodan Liang, et al.: Reversible recursive instance-level object segmentation. In: The IEEE Conference on CVPR, June (2016)
23. Macoadha, O., Github, Inception v3 model (2018). https://github.com/macaodha/inat_comp_2018/
24. Wan, L., et al.: Regularization of neural networks using dropconnect, In: Proceedings of the 30th, ICML, pp. 1058–1066, June (2013)
25. Krizhevsky, A. et al.: Imagenet classification with deep CNNs, In: Proceedings of the 26th Annual Conference on NeurIPS, pp. 1106–1114, USA, December (2012)
26. Julie P.: GrabCut for Automatic Image Segmentation [OpenCV Tutorial]. Sicara, 26 November 2020. https://www.sicara.ai/blog/grabcut-for-automatic-image-segmentation-opencv-tutorial
27. Huang, G., et al.: Densely connected convolutional networks. In: IEEE Conference on CVPR, pp. 2261–2269 (2017).
28. M. J. Huiskes, et al.: The MIR flickr retrieval evaluation. ACM ICMR, (MIR'08), Vancouver, Canada (2008)
29. Julie P.: GrabCut for Automatic Image Segmentation [OpenCV Tutorial]. Sicara, 26 November 2020. https://www.sicara.ai/blog/grabcut-for-automatic-image-segmentation-opencv-tutorial
30. Alessio, C.: Animals-10. Kaggle, 2019, https://www.kaggle.com/datasets/alessiocorrado99/animals10. Accessed 26 August 2020.
31. Sparsh, G.: Flowers Dataset. Kaggle, 2020 https://www.kaggle.com/datasets/imsparsh/flowers-datase. Accessed 26 August 2020
32. Roy, P., et al.: Natural images. Kaggle. (2018). https://www.kaggle.com/datasets/prasunroy/natural-images. Accessed 26 Aug 2020
33. Kane, Aditya. "Imagenette2.", Kaggle. (2019). https://www.kaggle.com/datasets/adityakane/imagenette2. Accessed 26 Aug 2020.
34. NEOZ, RO.: Kaggle. (2020) https://www.kaggle.com/datasets/roneoz/imagenette160. Accessed 26 Aug 2020
35. SACHIN. "Cats-vs-Dogs." Kaggle. (2020). https://www.kaggle.com/datasets/shaunthesheep/microsoft-catsvsdogs-dataset. Accessed 26 Aug 2020
36. Kang, G., et al.: Patchshuffle regularization. (2017). https://arxiv.org/abs/1707.07103
37. Devries, T., et al.: Improved regularization of CNNs with cutout. (2017). https://arxiv.org/abs/1708.04552
38. Nazeri, K., et al.: Two-stage CNN for breast cancer histology image classification In: Proceedings of the 15th, ICIAR 2018, pp. 717–726. P´ovoa de Varzim, Portugal (2018)
39. Aresta, G., et al.: BACH: grand challenge on breast cancer histology images. Med. Image Anal. **56**, 122–139 (2019)
40. He, K., et al.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on CVPR, pp. 770–778 (2016)
41. Srivastava, et al.: Learning representations for multimodal data with deep belief nets. In: International Conference on Machine Learning Workshop, vol. 79 (2012)
42. Wang, R., et al.: Large scale automatic image annotation based on convolutional neural network. J. Vis. Comm. Image Rep. **49**, 213–224 (2017)




**Dr. Shaima' Safa aldin** is currently a lecturer in the Continuing and Education Center, Al-Nahrain University, Iraq. She received her Ph.D. in Electrical & Electronics Engineering/Computer field from Gaziantep University, Turkey, in 2016. She received Master and Bachelor degrees from Al-Nahrain University, Iraq, in 2006 and 2003, respectively. The subjects of Ph.D. and M.Sc. theses were on designing a new teleconference system using an improved HEVC technique and a New Internet Service Provider Billing System, respectively. She has a teaching experience of more than 14 years in the computer architecture, programming, multimedia and networks. She has published several researchers in those fields to be applied in Al-Nahrain University.

**Dr. Noor Baha Aldin** received her B.S. degree with first rank in Computer Engineering from University of Technology, Baghdad, Iraq, in 2008, M.S. and Ph.D. degrees in 2012 and 2017 in Electrical and Electronics Engineering from Gaziantep University, Gaziantep, Turkey. She worked in Electrical and Electronics Engineering Department at Gaziantep University for around 5 years and is currently working as a full-time Assistant Prof. Dr. in Electrical and Electronics Engineering Department at Hasan Kalyoncu University, Gaziantep, Turkey. She has teaching experience of around 6 years. Her research interests are machine learning, image and video processing, wireless communication and networking.

**Dr. Mahmut Aykaç** received his B.S., M.S. and Ph.D. degrees in Electrical and Electronics Engineering Department from Gaziantep University, Gaziantep, Turkey, in 2006 and 2009 and 2017. He is currently working as a full-time Assistant Prof. Dr. in Communication division in Electrical and Electronics Engineering Department at Gaziantep University, Gaziantep, Turkey. He has teaching experience of longer than 15 years. His research interests are embedded systems, wireless communication and networking, image and video processing and RF energy harvesting.