

Image compression approach for improving deep learning applications

Raed Altabeiri, Moath Alsafasfeh, Mohanad Alhasanat

Department of Computer Engineering, Al-Hussein Bin Talal University, Ma'an, Jordan

Article Info

Article history:

Received Jun 28, 2022

Revised Feb 8, 2023

Accepted Feb 10, 2023

Keywords:

Convolutional neural network

Dataset

Deep learning

Image compression

Machine learning

ABSTRACT

In deep learning, dataset plays a main role in training and getting accurate results of detection and recognition objects in an image. Any training model needs a large size of dataset to be more accurate, where improving the dataset size is one of the most research problems that needs enhancement. In this paper, an image compression approach was developed to reduce the dataset size and improve classification accuracy for the trained model using a convolutional neural network (CNN), and speeds up the machine learning process, while maintaining image quality. The results revealed that the best scenario for deep learning models that provided good and acceptable classification accuracy was one that had the following parameters: 80×80 image size, 10 epochs, 64 batch size, 40 images dataset quality (images compressed 60%), and gray image mode. For this scenario a Dog vs Cat dataset is used, and the training time was 48 minutes, classification accuracy was 86%, and images dataset size was 317 MB on storage device. This size makes up 58% of the size of the original image's dataset, saves 42% of the storage space and reduces the processing resources consumption.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Raed Altabeiri

Department of Computer Engineering, College of Engineering, Al-Hussein Bin Talal University

Ma'an, Jordan

Email: raedtbeery2017@gmail.com

1. INTRODUCTION

The number of cell phones in use in the world is estimated at 4.7 billion phones. All users of these phones take photos and videos which usually accumulate up to 80% of their storage. These images and videos should be compressed. Image compression means reducing the size of the graphics file while keeping the image quality within an acceptable level, which allows more images to be stored and reduces the time required to send and download an image over the internet [1]. The image compression is performed in five steps: color space conversion, down-sampling, discrete cosine transform (DCT), quantization, and entropy encoding [2]. The digital image is a two-dimensional array of pixels. Image manipulation used mathematical functions and transformations, including smoothing, sharpening and segmentation of the image [3]. Additionally, computer vision by using images can solve more complex problems such as facial recognition (used, for example, by Snapchat to apply filters). Unsolvably digital image processing problems are now resolved by deep learning (DL) methods such as a convolutional neural network (CNN), a prime example of this is image classification [4].

Deep learning is a subfield of machine learning concerned with algorithms that are inspired by the structure and function of the brain called artificial neural networks. Deep learning technologies have evolved from Google Brain, founded by Andrew Ng, chief scientist at Baidu Research, and via various Google services [5]. Also, the following examples show where images are used in computer vision and deep learning: behavioral tracking (customers and how they behave), inventory management (generating a very accurate

estimate of the items available in the store, and analyzing the use of shelf space and suggesting better item placement), manufacturing (predictive maintenance, by identifying potential trouble before it occurs, and defect reduction, where the system can spot defects throughout the entire production line), healthcare (medical image analysis to find anomalies such as tumors or search for signs of neurological illnesses), autonomous vehicles (computer vision allows them to perceive and understand the environment around them to operate correctly) [6]. When discussing why deep learning took off in ExtractConf 2015, it is found that unlike conventional machine-learning and data mining techniques, deep learning is able to generate very high-level data representations from massive volumes of raw data. Therefore, it has provided a solution to many real-world applications [5]. An important point regarding the topic of deep learning, is that, it is the first class of scalable machine learning algorithms, where performance continues to improve when more data is presented.

Datasets are not algorithms might be the key limiting factor to development of human-level artificial intelligence [7]. The most popular image datasets on the internet are ImageNet (14M images with 21,841 classes), Microsoft common objects in context (MS COCO) (200K images with 80 classes), CIFAR-10 (60K images with 10 classes), PASCAL visual object classes (VOC) (11.5K images with 20 classes) [8]. A large dataset for deep learning algorithms needs large storage space and consumes computer resources. In addition, a large dataset needs a long training time to train the model and to get an accepted classification. These parameters are considered an impediment to real-time applications. For that, an approach has been proposed to compress the images dataset so that it reduces the size of the dataset and improves the classification accuracy, thus speeding up the machine learning process while preserving the image quality. The importance of this paper lies in developing a compression method for the huge dataset used in machine learning so that the process of machine training in visual recognition is accelerated, thus facilitating the process of creating deep learning applications. These applications include [6]: smart surveillance and monitoring, health and medicine, sports and recreation, robotics, drones, and self-driving cars. Visual recognition tasks, such as image classification, localization, and detection, are the core building blocks of many of these applications. This paper achieved the objectives; develop an approach to compress the dataset used for deep learning, use the developed compression method for training CNN models, speed up the training time and get high accuracy for deep learning applications, and reduce the required storage space for a dataset in deep learning algorithms.

Regards to literature, many papers presented using of different image compression techniques for improving deep learning applications. Gogoi and Begum [9] used progressive resizing to increase the classification accuracy of CNNs. The authors start training the model with 64×64 images, then double that size to (128×128), use a learning transfer technique to train the previous model to the new size, then double the size again (224×224), and train the previously trained model on the new size (which in turn includes training at sizes 128×128 and 64×64). From the results obtained it may be concluded that enhancement in learning strategies and scaling up image size is more effective than increasing model size by adding layers in high accuracy regime. With increased performance, the suggested architecture outperforms the competition in all three benchmark datasets. However, more testing on multiple datasets with different image sizes and training samples is required for general applicability. It is noticeable to the researchers' work that is a lack of interest in the reducing of training time, and their focus only on classification accuracy.

The effect of standard image compression techniques, such as joint photographic expert group (JPEG), JPEG2000, and high-efficiency video coding (HEVC) encoder on the classification performance of images using a CNN was done by [10]. Researchers studied coding parameters: color space, resolution, and quantization parameter (QP), and they studied the correlation between classification performance and image quality. They found that an image can be compressed by a factor 7, 16, 40 for a JPEG, JPEG2000, and an HEVC encoder and still maintain a correct classification by the CNN. The study was able to demonstrate that it is the JPEG encoder that causes the greatest degradation in image quality. The study also showed that the better portable graphics (BPG) encoder gives the best results. One of the observations of this study is that it does not use image encoders that reduce the "over-fitting" JPEG phenomenon, as the portable network graphics encoder (PNG); to build a more robust CNN.

Kannoja and Jaiswal [11] studied the effects of varying resolution on the performance of CNN-based image classification. The study proposed two separate training testing methods. In the first method training on original resolution images and testing on varying resolution images (TOTV), this classifier is trained on the original resolution training images dataset and evaluated on a set of varying resolution testing images dataset. In another method training on varying resolution images and testing on the corresponding resolution images (TVT), the same classifier is trained on each varying resolution training images dataset and evaluated on the corresponding resolution testing images dataset. The study shows that degradation in resolution from higher to lower decreases performance accuracy. One of the notes of the study is its lack of training time spent training CNNs on datasets.

The performance of deep neural network classifiers through a simple training strategy was implemented in [12]. They proposed dividing the internal deep neural network (DNN) parameter space into

sections and optimizing these sections individually. They implemented their proposed strategy using the popular limited memory-Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) optimization algorithm. The most prominent factor that impairs the performance of the L-BFGS algorithm is the total number of optimization parameters, which is very high. The researchers suggest a simple but very effective strategy, called the partial limited memory BFGS strategy (pL-BFGS), to implement the L-BFGS algorithm. Their classification experiments showed that the proposed method greatly enhances the training process of a DNN classifier and improves the classification accuracy. The researchers did not care to solve the problem of training time.

Gueguen *et al.* [13] proposed training the CNN directly with discrete cosine transform (DCT) coefficients computed in the middle of the JPEG codec. The researchers modified libjpeg to directly produce DCT parameters, modified ResNet-50 to accommodate inputs of different sizes and steps and evaluated performance on ImageNet. The study results were compelling: in a performance like ResNet-50 baseline, the researchers found an acceleration of 1.77x, and in performance significantly better than baseline, they obtained an acceleration of 1.3x. The results of this study may be effective for speeding up processing in a wide range of speed-sensitive applications, from processing large datasets in data centers to processing JPEG images locally on mobile devices.

Optimizing CNNs through parameter fine-tuning (learning transfer) was done by [14]. This method can improve CNN performance and can help overcome the lack of training data. The researchers investigated three main areas. The first is the optimal learning rate for transferred layers. The second is to analyze how different source datasets affect the outcome of several target datasets. Lastly, they compared an ensemble of fine-tuned networks to an ensemble of randomly initialized networks. The results of the study indicated that setting parameter finetuning always improved the accuracy of image classification, but the only possible downside is the increased training time required for pre-training.

Cheng *et al.* [15] start from that current DNN models are computationally expensive and consume intensive memory, which hinders their deployment in devices with low memory resources or in applications with stringent latency requirements. Therefore, the normal procedure is to compress the model and accelerate it in deep networks without significantly reducing model performance. The researchers reviewed the recent techniques for compacting and accelerating DNN models, such as parameter pruning and quantization, low-rank factorization, transferred/compact convolutional filters, and knowledge distillation. also, they reviewed some very recent successful methods, such as dynamic capacity networks and stochastic depths networks. A good compression method should achieve almost the same performance as the original model with much smaller parameters and less computing time.

The idea of integrating source coding and DL to obtain better performance in classifying images. The author proposed a new CNN topology done by [16], which absorbs the original input along with its various compressed versions. This architecture facilitates compressed information via compression inputs from low quality to high quality and allows the device to learn from all potential compressed information by itself. The researcher was able to increase the accuracy of the modern CNN image classification: 0.374% increase in Top-1 accuracy, 0.346% increase in Top-5 accuracy in terms of the inception V3 model, and 0.39% increase in accuracy Top-1, and 0.228% increase in Top-5 accuracy of the ResNet-50 V2 model. But like other technologies, there were some limitations to the new CNN topology: the 11 branches of this structure are needed to insert compressed versions of the originals. The total number of parameters in the new architecture is 11 times the original. This can easily deplete the graphics processing units (GPU) random-access memory (RAM) resource when training and evaluation of the entire architecture are desired. Therefore, the block-by-block training method is applied incrementally to avoid overusing computational resources at the start.

Jia *et al.* [17] wanted to improve system scalability and balance the communication and computation ratio for CNN. Researchers have built a highly scalable DL training system for GPU-dense sets with three main contributions: i) they proposed a mixed-precision training method, ii) they proposed an optimization approach for an extremely large minibatch size (up to 64K), and iii) they proposed highly optimized all-reduce algorithms. As a result, when training the ImageNet dataset, the researchers achieved 58.7% of the accuracy of the Top-1 test with AlexNet (95 epochs) in just 4 minutes using 1024 Tesla P40 GPUs, and they achieved 75.8% of the accuracy of the Top-1 test with ResNet-50 (90 epochs) in 6.6 minutes using 2048 Tesla P40 GPUs, which outperform current systems. In summary, the related works showed that there was needed to develop an image compression approach to reduce the size of the dataset and improve the classification accuracy, which speeds up the machine learning process while maintaining image quality.

2. METHOD

This section introduces the dataset compression method used to improve deep learning applications. This method is presented in three main steps. First: creating a database of compressed images by applying the standard JPEG algorithm. Second: create a CNN model for image classification. Third: examine the effect of image compression on the model training time, classification accuracy, and the size of the image data set on

the storage device. The primary purpose of this method is to study the effect of the compression of the dataset on the performance and accuracy of the deep learning model. This study will help determine whether it is worth compressing the images for the training purpose in the machine learning model.

2.1. The phases of the implemented compression method

In this paper, the approach was developed based on six consecutive phases as detailed in Figure 1, and the following paragraphs. These six phases were implemented on a personal laptop computer with the following characteristics: Processor: Intel® Core™ i7-7500U CPU @2.7 GHz 2.90 GHz. Installed memory (RAM): 8.00 GB. System type: 64-bit Operating System, x64-based processor. Windows edition: Windows 10 Pro © 2019 Microsoft Corporation.

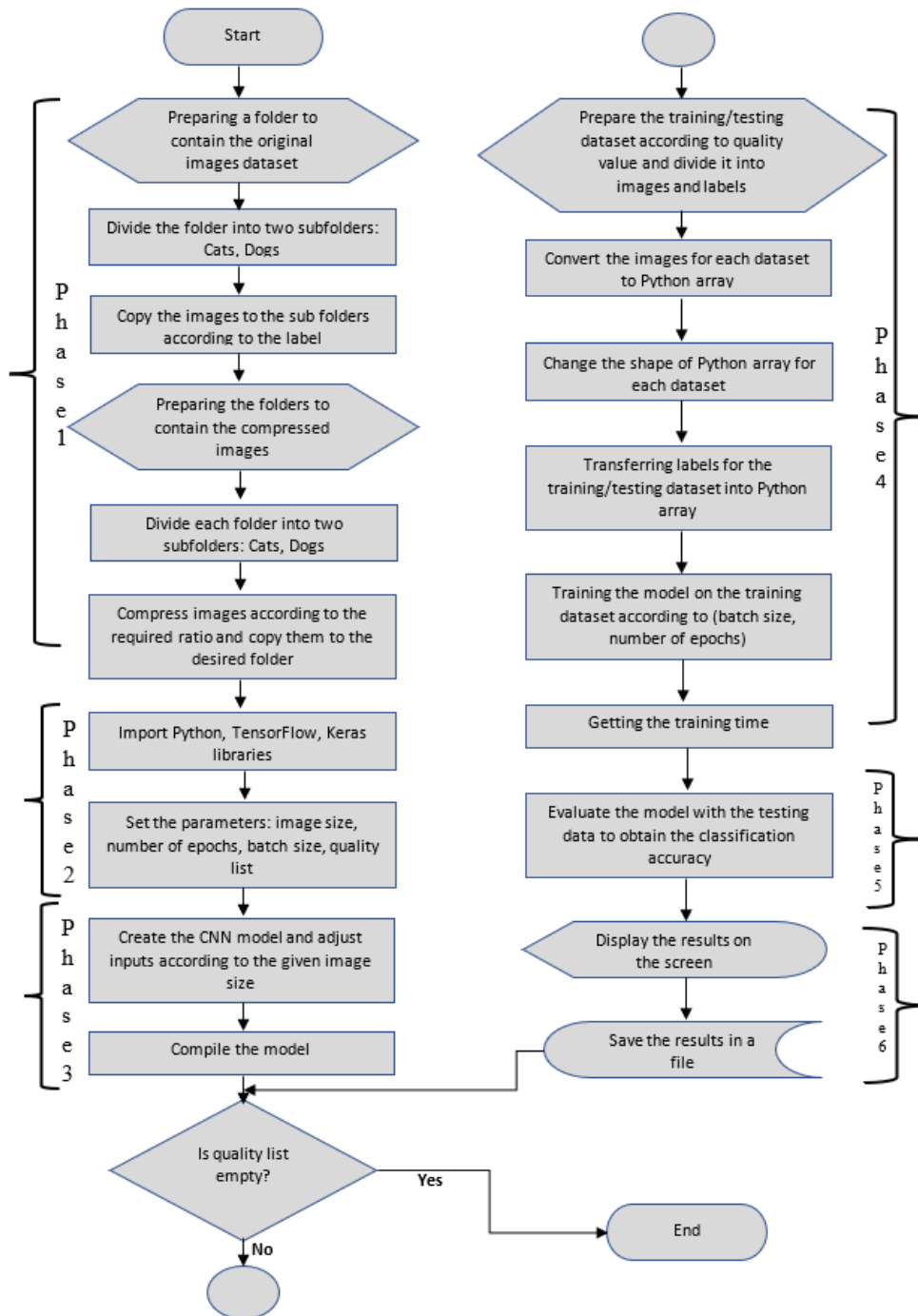


Figure 1. The proposed compression and training method

In phase 1, the Cats vs. Dogs images dataset (25,000 images) is getting and copy into the work folder, which was divided into two sub-folders: Cats and Dogs, then the original images were copied to the subfolders according to the label (name of the image file). Images were classified according to their content by their labels, using the word "dog" or "cat" followed by a number indicating the sequence of the image within the images dataset. In the meantime, the compressed image folders were prepared. Each folder was divided into two sub-folders: Cats and Dogs. Then the original images dataset is read and compressed to the desired folder according to the given quality (10, 20, 40, 80, 100). Custom code was written to load images into the memory image-by-image from the dataset folder according to the desired color pattern (color or gray), and resize them, then save them ready for modeling as a single NumPy array. For each array of images and labels, the NumPy array will look like this when printed:

Print (X_train.shape, Y_train.shape) → (25000, 120, 120, 3) (25000,)

In phase 2, the Python, TensorFlow, and Keras libraries are imported into the Jupyter framework. Python constitutes the best programming language in the artificial intelligence (AI) and machine learning (ML) fields [18]. It has become the best for developers, it is a great option for ML, data science, and the internet of things (IoT) that have been gaining momentum recently [19]. Keras is a powerful tool for developing DL models. It wraps the powerful numerical computation libraries Theano and TensorFlow [20]. Keras surpassed all libraries used in building and training DL models because of its features that made it at the forefront [21]. TensorFlow is a comprehensive system of tools and libraries that greatly ease and accelerate the application of neural network models [22]. Jupyter framework was used in DL projects because it is a great environment in which to develop code and communicate results. The primary programming languages that Jupyter supports are Julia (Ju), Python (Py), and R [23]. Jupyter Notebook has many features that have made it a popular application in the field of DL, which supports over 40 programming languages, notebooks can be shared with others, and there is an interactive output feature: hypertext markup language (HTML), images, videos, LaTeX, and custom multipurpose internet mail extensions (MIMEs) [24]. The main training parameters are chosen for covering different cases. The first parameter is epochs, which takes the following values (10, 20, 40). One Epoch is an entire dataset, that is passed forward and backward through the neural network. Epoch is divided into several smaller batches. The second parameter is the batch size, which takes the values (64, 128, 256). These values match training accuracy and test accuracy, and also reduce training time. Batch size (Bs) is the number of training images present in one iteration. When (N) represents the total number of training images, and (In) represents the number of iterations needed to complete one epoch, then the number of iterations is calculated as shown in (1).

$$In = \frac{N}{Bs} \quad (1)$$

The third parameter is the quality list. The quality list parameter takes the following values (10, 20, 40, 80, and 100). These values are taken to form a doubling ratio for image quality from 10 to 100. The quality value is the complement of the compression ratio.

In phase 3, the CNN model was created, and the inputs were adjusted (image size and color mode). The color mode takes the values 1 for gray and 3 for color, based on the number of arrays representing each mode. Then the model was compiled and become ready to do the training process. Neural network is a machine learning (ML) technique that is inspired by the human nervous system and the structure of the brain. It was structured from artificial neurons, called nodes. These nodes are stacked in three layers: the input layer, the hidden layer(s), and the output layer. Inputs are provided to each node. The node multiplies the inputs with random weights, calculates them, and adds a bias. Finally, activation functions (nonlinear functions) are applied to determine which neuron to fire to get the output, through the (2) [25].

$$Output = Activate ((weights \cdot inputs) + bias) \quad (2)$$

Bias value allows shifting of the activation function left or right, to be more flexible about the features that will be learned. Each layer needs a single bias node. A bias node can add at the first few layers and not at the last ones [26]. Bias affects output values only; so, the role of bias is to help ensure that the output fits the incoming input better. Biases are typically configured to be zero, as the fraction of inconsistency is provided by small random numbers in the weights [27]. The previous process is shown in Figure 2.

A CNN model was developed by using the Keras library. The model involves stacking convolutional layers with small 3×3 filters followed by a max-pooling layer. Together, these layers form a block, and these blocks can be repeated where the number of filters in each block is increased with the depth of the network such as 32, 64, and 128 for the three blocks of the model. The rectified linear unit (ReLU) activation function was used in each layer. As shown in (3) Shows the abbreviated mathematical expression for a ReLU function;

where for negative values of x , the value of the ReLU functions is zero, and for positive values of x , the value of the ReLU functions is x [28]:

$$\text{ReLU}(x) = \max(0, x) \quad (3)$$

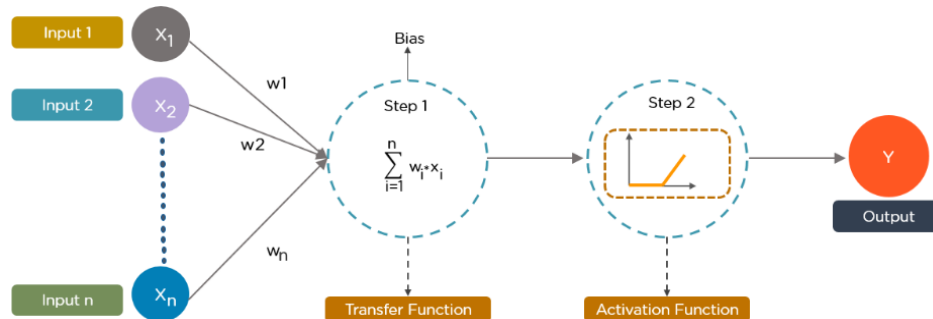


Figure 2. Data processing in neural network architecture

The most commonly used activation functions are: linear activation function, sigmoid activation function, tanh function, ReLU function, SoftMax activation function [28]. The researchers suggest that the CNN model can benefit from regularization techniques, such as Dropout [17]. This technique drops some weights to reduce the time required to train the model. The classification task here is a binary classification task, requiring the prediction of one value of either 0 “dog” or 1 “cat”. An output layer with 1 node and a sigmoid activation function has been used. In compile process, the CNN model has been optimized using the Adam optimizer and the binary cross-entropy loss function.

In phase 4, the images dataset was divided into the training dataset which contains 23,000 images for dogs and cats, and the testing dataset which contains 2,000 images for dogs and cats. Then the training/testing dataset is divided into images and labels. Then the part of the image for each dataset is converted to a Python array. Now, the images' Python array must reshape for each dataset to make it ready to use in the training/testing process. Also, the part of the label for each training/testing dataset must be transferred into the Python array but reshaping is not made. The model was trained on the training dataset according to (batch size, and number of epochs), then the training time was obtained. In phase 5, the images dataset of testing and its set of labels were passed to the model during the evaluation process to obtain the classification accuracy.

In phase 6, the results were printed on the screen and saved to a file. Printed and saved results included scenario name, file name, file creation time, image size, epochs, batch size, image quality, model training time, and classification accuracy. Phases 4-6 were repeated; so that the training, evaluating, printing, and saving processes were repeated for all compressed image datasets according to different compression ratios (10, 20, 40, 80, and 100). The main programming of the developed approach according to the previous six phases was explained in the pseudocode in Figure 3.

2.2. The executed scenarios

After preparing the images dataset within the folders, the programming code that is necessary to perform all scenarios on this data was created. The scenarios were divided to include all the possibilities of the variables used in this study. The scenarios were divided into several groups. Dataset within these groups is processed according to multiple variables, such as the image size (80×80 , 120×120), the batch size (64, 128, 256), and the image quality if it is original or gray, whether it is color compressed (10, 20, 40, 60, 100), or gray compressed (10, 20, 40, 60, 100), and according to the number of epochs (10, 20, 40). Thus, the number of groups used in performing the developed approach on the image's dataset was 12, and the number of scenarios that have been implemented was 112. The training period for the DL model on these scenarios took 289 training hours. Summary of groups and scenarios are shown in the following sections.

2.2.1. Colored image with original quality

9 scenarios are implemented for a size 80×80 , where epochs take the values (10, 20, 40), then each one of these values takes three values of the batch size (64, 128, 256). The training time is nearly doubled due to the increase in the number of epochs. Note that the size of the original dataset on the hard disk was remained 546 MB in each scenario. In addition, for size 120×120 , this includes 9 scenarios, the training time increases dramatically. Note that the size of the original dataset on the hard disk was remained 546 MB in each scenario.

```

START
1. Preparing a folder to contain the original images dataset
2. Divide the folder into two sub-folders: Cats, Dogs
3. Copy the original images to the sub folders according to the label
4. Preparing the folders to contain the compressed images
5. Divide each folder into two sub-folders: Cats, Dogs
6. Compress images according to the required ratio and copy them to the desired folder
7. IMPORT Python, TensorFlow, Keras
8. SET image size TO (80X80) or (120X120)
9. SET epochs TO 10 or 20 or 40
10. SET batch size TO 64 or 128 or 256
11. SET quality list TO (10, 20, 40, 60, 100)

DO
1. Prepare the training/testing dataset and divide it into images and labels.
2. CONVERT the images for each dataset to Python array.
3. RESHAPE Python array for each dataset
4. CONVERT labels for the training/testing dataset to Python array
5. CREATE the CNN model and adjust inputs according to the given image size
6. COMPILE the model
7. TRAIN the model on the training dataset according to batch size, epochs
8. GET training time
9. EVALUATE the model with the testing dataset
10. GET classification accuracy
11. PRINT image size, epochs, batch size, quality, training time, classification accuracy
12. SAVE image size, epochs, batch size, quality, training time, classification accuracy TO
file

WHILE quality list NOT empty
END

```

Figure 3. Pseudocode of the deep learning algorithm of the developed approach in this study

2.2.2. Gray image with original quality

9 scenarios are implemented for a size 80×80, where epochs take the values (10, 20, 40), then each one of these values takes three values of the batch size (64, 128, 256). The training time is nearly doubled due to the increase in the number of epochs. Note that the size of the original gray pattern dataset on the hard disk is 941 MB. This is illogical since the size of gray images should be less than the size of the color images. In addition, for size 120×120, this includes 9 scenarios, the training time increases dramatically. Note that the size of the original gray pattern dataset on the hard disk is 941 MB. This is illogical since the size of gray images should be less than the size of the color images.

2.2.3. Compressed gray, size 80x80, different qualities

This includes 15 scenarios for epochs is 10 and the image quality (which is a complement to the compression ratio) is doubled (10, 20, 40, 80, 100), which means that it is compressed in the following ratios (90%, 80%, 60%, 20%, 0%) respectively. Then each one of quality values takes three values of the batch size (64, 128, 256), bringing the total to 15 scenarios. Note that the size of compressed data of the gray pattern on the hard disk varies according to the image quality (142, 211, 317, 594 and 1,180 MB) respectively. When the epochs are 20, this includes 15 scenarios, the training time doubles due to the increase in the number of epochs. A case of epochs is 40 and the batch size is 64, this includes 5 scenarios (each quality takes one scenario), still the training time doubles due to the increase in the number of epochs.

2.2.4. Compressed gray, size 120x120, different qualities

This includes 15 scenarios for epochs is 10 and the image quality is doubled (10, 20, 40, 80, 100). Then each one of quality values takes three values of the batch size (64, 128, 256), bringing the total to 15 scenarios. When increasing the image size from 80×80 to 120×120, the training time increases dramatically. Note that the size of compressed data of the gray pattern on the hard disk varies according to the image quality (142, 211, 317, 594 and 1,180 MB) respectively. In case epochs is 20, bringing the total to 15 scenarios, the training time doubles due to the increase in the number of epochs. A case of epochs is 40 and the batch size is 64, this includes 5 scenarios (each quality takes one scenario), the training time doubles due to the increase in the number of epochs.

2.2.5. Compressed color, size 80x80, quality 40

This includes 3 scenarios for epochs is 10 and the batch size was changed with (64, 128, 256) (each batch size takes one scenario). Note that the size of the compressed gray on the hard disk according to the image quality 40 is 317 MB. For size 120×120, increasing the image size from 80×80 to 120×120, the training time increases dramatically.

3. RESULTS AND DISCUSSION

In principle, the number of epochs is affecting the training time, while the image compression ratio is affecting the classification accuracy. Therefore, scenarios that give too long training time and scenarios that give low classification accuracy will be excluded. The scenarios that fulfill the three study objectives will be weighed together to obtain the best possible scenario. In this study, 112 scenarios were applied to the image's dataset, which took approximately 289 hours of training CNN models. The following variables and its values are used for each scenario: image size (80×80 and 120×120), number of epochs (10, 20, 40), batch size (64, 128, 256), image quality (original color, original gray, compressed color images quality (10, 20, 40, 80, 100), compressed gray images quality (10, 20, 40, 80, 100)). Therefore, the scenario is expressed in the following way: scenario (number) (image size, number of epochs, batch size, image quality). The following abbreviations are used: cc to represent the compressed color image quality and cg to represent the compressed grayscale image quality. The following scenario: Scenario (37) (80, 10, 64, 10 cg); means the scenario number is 37, the image size is 80×80, the number of epochs is 10, the batch size is 64, and the quality value is 10 (90% compression ratio) with compressed grayscale image.

3.1. The tested scenarios

When scenarios were analyzed, it is noticed that the training time is divided into four categories, which are (50 minutes, one hour and a half, three hours and a half, seven hours and a half). The highest classification accuracy in each category is as follows: 86.4, 88.2, 89.4, and 89.5, respectively. The difference in the previous four classification accuracy values is not significant value, because obtain the highest accuracy is not from the aims of the study. Therefore, the first training time category was chosen because it achieves one of the study objectives, which is to reduce the training time (not to exceed 50 minutes).

Within this category, it is noted that the scenario that achieves the highest possible accuracy is the following scenario: Scenario (39) (80, 10, 64, 40 cg); images dataset quality is 40, compressed gray (cg). This gave 48 minutes of training time, a classification accuracy of 86%, and image dataset size of 317 MB on the storage device. This size makes up 58% of the original images dataset size and saves 42% of storage space. Note that when image resizing is made for the gray compressed images dataset on the storage device to 80×80, the size of the dataset becomes 123 MB, which is approximately 23% of the original dataset size and saves 77% of the storage space.

3.2. Results summary

In the research papers reviewed in this paper, which on developing deep learning, although they have made great progress in improving deep learning algorithms and developing high-performance training systems, these have not made big attention in images dataset size-which are used in the training process-on the storage device, nor at the training time. The proposed approach in this paper reduces the size of the dataset by compressing it while preserving the image quality, thus reduces the required training time and improves the classification accuracy. Table 1 is showing a comparison between seven scenarios; the best five scenarios that achieve the study objectives, and the best- and worst-case scenarios at all, sorted in ascending order according to the classification accuracy. Figure 4 is showing the best five scenarios that achieved the study objectives, which the training time does not exceed the 50-minutes, sorted in ascending order according to the classification accuracy.

Table 1. The best five scenarios that achieve the study objectives, and the best- and worst-case scenarios at all, sorted in ascending order according to the classification accuracy

Scenario no.	Image size	Epochs	Batch size	Quality	Training time	Classification accuracy	Dataset size on HD (MB)
43	80×80	10	128	20 (C. GRAY)	0:45:33	48.6	211
39	80×80	10	64	40 (C. GRAY)	0:48:28	86.0	317
41	80×80	10	64	100 (C. GRAY)	0:45:16	86.2	1180
20	80×80	10	128	(O. GRAY)	0:46:25	86.3	941
1	80×80	10	64	(O. COLOR)	0:48:50	86.3	546
108	80×80	10	128	40 (C. COLOR)	0:49:56	86.4	317
17	120×120	40	128	(O. COLOR)	7:46:02	89.5	546

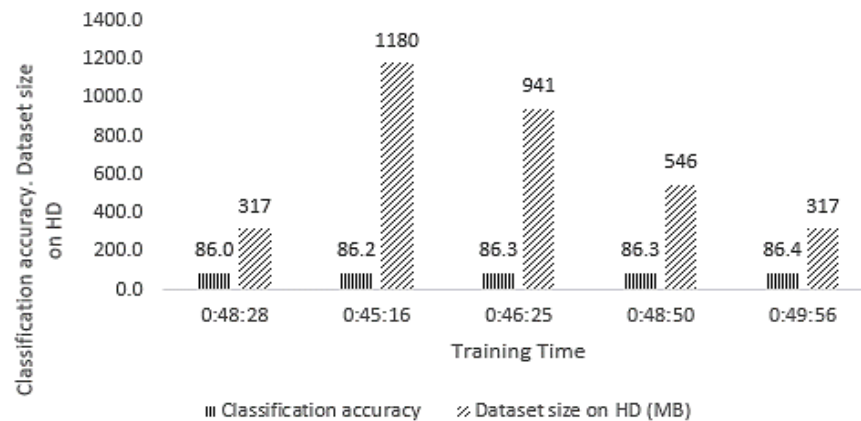


Figure 4. The best five scenarios that achieved the study objectives, which the training time does not exceed the 50-minutes, sorted in ascending order according to the classification accuracy

4. CONCLUSION

In this study, the Dogs vs. Cats image dataset from Kaggle Corporation was used, which contains 25,000 color images. 112 scenarios were applied in both the color and gray patterns. These scenarios took 289 training hours of CNN models. From this study, it was found that the best scenario that gives very good and acceptable classification accuracy is Scenario (39) (80, 10, 64, 40 cg). The training time of this scenario is 48 minutes, and the classification accuracy is 86%, and the image dataset size 317 MB. This size makes up 58% of the original images dataset size and saves 42% of storage space. Note that when image resizing is made for the gray compressed images dataset to 80×80, the size of the dataset becomes 123 MB, which is approximately 23% of the original dataset size and saves 77% of the storage space.




REFERENCES

- [1] L. Alam, P. K. Dhar, M. A. F. M. R. Hasan, M. G. B. Sarwar, and G. M. Daiyan, "An improved JPEG image compression algorithm by modifying luminance quantization table," *IJCSNS International Journal of Computer Science and Network Security*, vol. 17, no. 1, pp. 200–208, 2017.
- [2] A. Shawahna, M. E. Haque, and A. Amin, "JPEG image compression using the discrete cosine transform: An overview, applications, and hardware implementation," *Prepr. arXiv1912.10789*, Nov. 2019.
- [3] V. Tyagi, *Understanding digital image processing*. CRC Press, 2018.
- [4] N. O. Mahony *et al.*, *Advances in computer vision*, vol. 943. Cham: Springer International Publishing, 2020.
- [5] S. Pouyanfar *et al.*, "A survey on deep learning: Algorithms, techniques, and applications," *ACM Computing Surveys*, vol. 51, no. 5, pp. 1–36, Sep. 2019, doi: 10.1145/3234150.
- [6] S. Khan, H. Rahmani, S. A. A. Shah, and M. Bennamoun, "A guide to convolutional neural networks for computer vision," *Synthesis Lectures on Computer Vision*, vol. 8, no. 1, pp. 1–207, Feb. 2018, doi: 10.2200/S00822ED1V01Y201712COV015.
- [7] Š. Balogh, "Knowledge and datasets as a resource for improving artificial intelligence," in *Proceedings of the Computational Methods in Systems and Software*, 2021, pp. 828–837.
- [8] B. Aggarwal, A. Acharya, and A. Laghari, "A survey on image datasets for computer vision and deep learning convolutional neural network tasks," *Department of Electrical and Electronics Engineering*, 2020, doi: 10.13140/RG.2.2.22391.44966.
- [9] M. Gogoi and S. A. Begum, "Progressive 3-layered block architecture for image classification," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 3, pp. 499–508, 2022, doi: 10.14569/IJACSA.2022.0130360.
- [10] M. Dejean-Servières, K. Desnos, K. Abdelouahab, W. Hamidouche, L. Morin, and M. Pelcat, "Study of the impact of standard image compression techniques on performance of image classification with a convolutional neural network," Internship Report, IETR-Vaader, 2018.
- [11] S. P. Kannoja and G. Jaiswal, "Effects of varying resolution on performance of CNN based image classification an experimental study," *International Journal of Computer Sciences and Engineering*, vol. 6, no. 9, pp. 451–456, Sep. 2018, doi: 10.26438/ijcse/v6i9.451456.
- [12] A. Caliskan, M. E. Yuksel, H. Badem, and A. Basturk, "Performance improvement of deep neural network classifiers by a simple training strategy," *Engineering Applications of Artificial Intelligence*, vol. 67, pp. 14–23, Jan. 2018, doi: 10.1016/j.engappai.2017.09.002.
- [13] L. Gueguen, A. Sergeev, B. Kadlec, R. Liu, and J. Yosinski, "Faster neural networks straight from JPEG," in *Advances in Neural Information Processing Systems*, 2018, vol. 31.
- [14] N. Becherer, J. Pecarina, S. Nykl, and K. Hopkinson, "Improving optimization of convolutional neural networks through parameter fine-tuning," *Neural Computing and Applications*, vol. 31, no. 8, pp. 3469–3479, Aug. 2019, doi: 10.1007/s00521-017-3285-0.
- [15] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," *Prepr. arXiv1710.09282*, 2017.
- [16] Y. Jiang, "New convolutional neural network topology with compressed information to enhance accuracy for image classification task," M.S. thesis of Applied Science, Waterloo, Ontario, Canada, 2019.
- [17] X. Jia *et al.*, "Highly scalable deep learning training system with mixed-precision: training ImageNet in four minutes," *Prepr. arXiv1807.11205*, 2018.




- [18] S. Saabith, V. Thangarajah, and M. M. M. Fareez, "Popular python libraries and their application domains," *International Journal of Advance Engineering and Research Development*, vol. 7, no. 11, pp. 18–26, 2020.
- [19] P. N. S. Jyothi and R. Yamaganti, "A review on python for data science, machine learning and IoT," *International Journal of Scientific & Engineering Research*, vol. 10, no. 12, pp. 1–8, 2019.
- [20] B. T. Chicho and A. B. Sallow, "A comprehensive survey of deep learning models based on keras framework," *Journal of Soft Computing and Data Mining*, vol. 2, no. 2, pp. 49–62, 2021.
- [21] B. J. Erickson, P. Korfiatis, Z. Akkus, T. Kline, and K. Philbrick, "Toolkits and libraries for deep learning," *Journal of Digital Imaging*, vol. 30, no. 4, pp. 400–405, Aug. 2017, doi: 10.1007/s10278-017-9965-6.
- [22] B. Pang, E. Nijkamp, and Y. N. Wu, "Deep learning with TensorFlow: a review," *Journal of Educational and Behavioral Statistics*, vol. 45, no. 2, pp. 227–248, Apr. 2020, doi: 10.3102/1076998619872761.
- [23] J. M. Perkel, "By jupyter, it all makes sense," *Nature Journal*, vol. 563, pp. 145–146, 2018.
- [24] A. Davies, F. Hooley, P. Causey-Freeman, I. Eleftheriou, and G. Moulton, "Using interactive digital notebooks for bioscience and informatics education," *PLOS Computational Biology*, vol. 16, no. 11, Nov. 2020, doi: 10.1371/journal.pcbi.1008326.
- [25] A. Shrestha and A. Mahmood, "Review of deep learning algorithms and architectures," *IEEE Access*, vol. 7, pp. 53040–53065, 2019, doi: 10.1109/ACCESS.2019.2912200.
- [26] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, Nov. 2018, doi: 10.1016/j.heliyon.2018.e00938.
- [27] J. Collis, "Glossary of deep learning: Bias," medium.com. [Online]. Available: <https://medium.com/deeper-learning/glossary-of-deep-learning-bias-cf49d9c895e2> (accessed: Jun. 26, 2022).
- [28] S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in neural networks," *International Journal of Engineering Applied Sciences and Technology*, vol. 4, no. 12, pp. 310–316, 2020.

BIOGRAPHIES OF AUTHORS






Raed Altabeiri    got his master's degree in Computer and Networks Engineering from Al-Hussein Bin Talal University in 2021, and he got the bachelor's degree in Computer Science from Mutah University in 1992. Raed served as a teacher of Computer skills course in the Ministry of Education in Jordan from 1992 to 2022. He can be contacted at email: raedtbeery2017@gmail.com.



Moath Alsafasfeh    earned his Ph.D. in 2017 at Western Michigan University (WMU) with excellent evaluation. He is currently an Associate Professor of Electrical and Computer Engineering at Al-Hussein Bin Talal University (AHU), Jordan. The current research of Alsafasfeh is going in using computer vision and machine learning to operate, monitor, maintain, and control the different systems in an efficient processing time. Dr. Alsafasfeh was awarded several national prizes, and scholarships to get Ph.D. and bachelor's degrees. He has strong relationships with different local, regional, and international researchers and he is involved in many different international research and capacity-building projects. He can be contacted at email: moath.alsafasfeh@ahu.edu.jo.



Mohanad Alhasanat    was born in Jordan in 1981. He received his Ph.D. from Islamic Science University of Malaysia (USIM) in 2016. Currently, he is an Associate Professor in networks at the Department of Computer Engineering, Al-Hussein Bin Talal University (Jordan). His research focuses on Computer Networks including enhancing TCP congestion control mechanism over wireless networks. In addition, he has research interest in Delay Tolerant Networks, Artificial Intelligence, Computer Vision, and Deep Machin learning. He can be contacted at email: mohanadhasanat@ahu.edu.jo.