ENHANCED KNOWLEDGE IN SCIENCES AND TECHNOLOGY
e-ISSN: 2773-6385

EKST

Vol. 4 No. 2 (2024) 161-170
https://publisher.uthm.edu.my/periodicals/index.php/ekst

Image Detection by Deep Learning with Convolutional Neural Network (CNN)

Khoo Kang Wei[1], Lee Siaw Chong[1*]

[1] *Department of Mathematics and Statistics, Faculty of Applied Sciences and Technology, UTHM Kampus Cawangan Pagoh, Hab Pendidikan Tinggi Pagoh, KM 1, Jalan Panchor, 84600 Pagoh, Muar, Johor, MALAYSIA*

*Corresponding Author: sclee@uthm.edu.my
DOI: https://doi.org/10.30880/ekst.2024.04.02.020

Article Info

Received: 27 December 2023
Accepted: 11 January 2024
Available online: 12 December 2024

Keywords

Image Classification, Deep Learning, Artificial Neural Network (ANN), Convolutional Neural Network (CNN)

Abstract

Image detection is the process that uses deep learning to identify an object and classify it. It is widely applied in digit recognition, facial recognition, and many other areas. The convolutional neural network (CNN) is one of the deep learning techniques that is frequently used in image detection because it can effectively extract characteristics and develop pattern recognitions of an image. In this paper, a CNN model is developed using Python to classify the images of cats and dogs from a Kaggle machine learning competition held in 2013. The impact of the training epochs and batch sizes on the performance of the CNN model has been studied. Overall, an increase in the number of training epochs will improve accuracy, but when a certain limit is reached, the CNN model may overfit the training data. The batch size can affect training speed and model optimisation. The smaller the batch size, the longer the training time. For a fixed learning rate, there is an optimal batch size that maximises the performance. In order to achieve better accuracy and lower loss values for a CNN model with a learning rate of 0.001, it is advised to use a smaller batch size for the CNN model, such as 8 or 16. It is advised to use a CNN model with a bigger batch size, such 64 or 128 though, given the training time. Larger batches lead to shorter training durations. Furthermore, in order to save computational costs, this study recommends that the number of training epochs not exceed 10.

1. Introduction

Image detection is part of the image processing problem that uses artificial intelligence (AI) algorithms to detect and classify objects in images [1]. AI is used to construct a dynamic computing environment to simulate human-like intelligence and behaviour in machines and to learn from experience. Machine learning is a technique used in AI that focuses on training algorithms to learn from data and to develop over time. Machine learning algorithms can automatically analyse enormous datasets, detect patterns, and make predictions without being explicitly programmed [2].

Deep learning is a subset of machine learning algorithms, and it focuses on training multi-layered artificial neural networks (ANN) for data analysis. The neural network is one of the deep learning techniques that is frequently used in image detection. An ANN is made up of a network of nodes, and these nodes are connected to one another by basic computations. Each node has a value of its own, which is known as bias, and the strength of their connections to one another is assigned a value that is otherwise known as weights. The neural networks use activation functions with weights and biases to compute the output. Thus, the weights and biases will reflect the significance of the corresponding features in making predictions or recognising patterns [3].

Convolutional neural network (CNN) is a subtype of ANN that has excellent performance in processing graphics. The CNN model can effectively extract characteristics from images and develop pattern recognitions from raw pixel data without any additional coding or pre-processing. A CNN model is made up of three types of layers, which are the convolutional layer, the pooling layer, and the fully connected layer which have different functions in this model.

In this research, a Python-based CNN model will be developed for detecting and classifying images of dogs and cats. Python has many neural networks libraries, such as TensorFlow, PyTorch and Keras, that are easy to learn and use but for this research, the CNN model is designed without using any Python built-in libraries for AI or machine learning. This research will use a dataset from a Kaggle machine learning competition held in 2013 [4]. This dataset has 25,000 images labelled with 12,500 dogs and the same number of cats. The size of these images will be resized to (128×128) to reduce the complexity of the computation. Supervised machine learning was used in this study since the dataset used was labelled as cats or dogs. Since the dataset has answer labels, supervised methods are easier to evaluate [5]. Many studies employing different activation functions have been conducted on the CNN model to enhance its efficiency. However, relatively little study has been done on the impact of the training epochs and batch sizes on the accuracy of the CNN model.

2. Methodology

The convolutional neural network's (CNN) structure is based on three layers, which are the convolutional layer, the pooling layer, and the fully connected layer. In the convolutional layer, a feature map will be created by extracting the features of image arrays. Then, in the pooling layer, the feature map size is reduced, and the output is predicted in the fully connected layer. After the image array passes through these layers, a backpropagation process based on the gradient descent algorithm is used. It is used to improve the accuracy of CNN models by calculating the gradient of the loss function.

2.1 The Basic Architecture and Theorem of Convolutional Neural Network

Convolutional neural networks (CNN) have achieved remarkable success in many application areas, such as facial recognition, image classification, and object segmentation by extracting the features of an image. CNN is a subtype of artificial neural network, and its structure can be divided into three layers which are the convolutional layer, the pooling layer, and the fully connected layer. The process of using these layers for computing and transforming the input into the prediction output is called forward propagation [6]. Each layer has a different purpose in image processing:

- Convolutional layer
  The first layer of the CNN model is used to form a feature map by extracting the features of images.
- Pooling layer
  The layer after the convolutional layer is used to reduce the computational costs by decreasing the size of the convolved feature map.
- Fully connected layer
  The last layer in the CNN model is a classic neural network architecture where all the nodes are connected to all the nodes in the output layer. This layer is used to predict the output.

Fig .1 shows an image passed through the three layers of a CNN model and having its probability predicted.

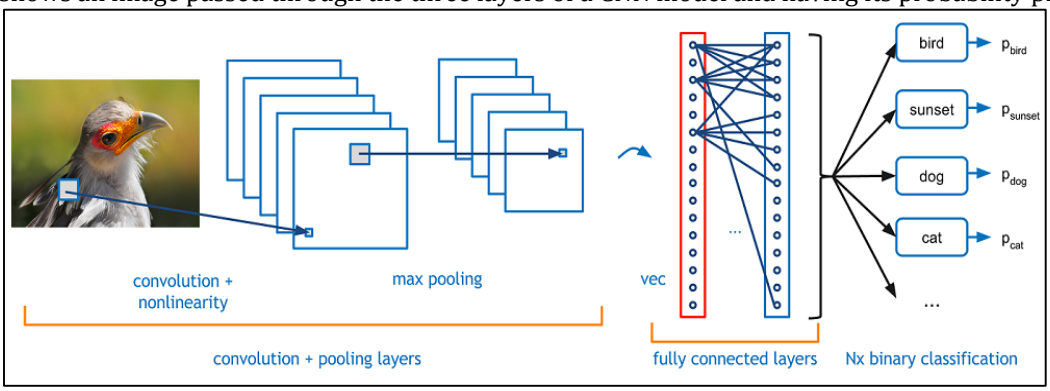


Fig. 1 The three layers of a CNN model [7]

2.1.1 The Convolutional Layer

In this layer, an *M* x *M* sized pixel array of an image will undergo a convolution operation with at least one kernel array, also known as a filter, and the value of the kernel array, which represents the weights. The kernel size is

always odd-numbered, and is typically 3 x 3 because it can reduce the computational costs. The goal of the convolution operation is to calculate an entrywise product between each element of the kernel and the pixel array for each corresponding position of the pixel array. The entrywise product is summed to obtain the output value for the corresponding position of the output array, and the output array is known as a feature map [6]. An example of a convolution operation is shown in Fig. 2.

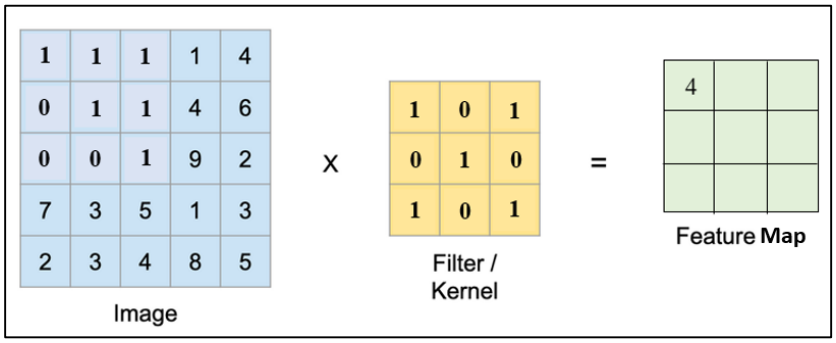


Fig. 2 An example of the convolution operation

According to Fig. 2, the size of the input array is reduced to 3 x 3 after the convolution operation. The number of convolution operations at the edges of the image is significantly less than the middle of the image, and therefore, this results in the loss of information on the edges. This extraction without any padding is called valid padding. Hence, to avoid the loss of information on the feature map, the *same padding* method is used. This method can maintain the size of the output array by adding extra rows and columns of zeroes on the outer dimensions of the images [8]. In this research, the original RGB images are converted to black and white images.

After that, the feature map will be passed through a nonlinear activation function such as rectified linear unit (ReLU) function, sigmoid function, softmax function, and tanh function. These functions will then easily produce the CNN model to adapt to a variety of data and differentiate between the outcomes. The ReLU function is used in this research since its computing speed is much faster than the other functions [9]. This is because it only involves comparisons between its input and the value zero. The ReLU function, $R(x)$ is

$$R(x) = \begin{cases} x, & x \geq 0, \\ 0, & x < 0. \end{cases} \tag{1}$$

Even though the computing speed of the ReLu function is faster than the other function, it may cause the dying ReLU problem. When the input range of the ReLu function is negative, it will return a value of zero, and the neural network's learning cannot be recovered. To overcome this problem, the initial weight and biases need to be specified with the use of the He Initialisation function. The new weights or biases with the He Initialisation function are

$$\text{new weight} = \text{initial weight} * \frac{\sqrt{2}}{n},$$

and

$$\text{new bias} = \text{initial bias} * \frac{\sqrt{2}}{n},$$

where n is number of inputs.

2.2 The Pooling Layer

After the pixel images are extracted by the convolution operation and passed through a ReLU function, the feature map will then pass to the next layer, the pooling layer. Since the extraction applies the *same padding* method, the size of the feature map is the same as the pixel images, and the computation cost of the feature maps is higher when the size of pixel images is large. To overcome this, the down-sampling operation will be conducted on the feature maps to reduce the dimensionality of the feature maps without losing any information about the images while lowering the cost of computation [10].

In the pooling layers, there are many pooling operations, such as max pooling, average pooling, and sum pooling. For this research, the pooling operation that will be used is max pooling, which extracts the maximum value in patches from feature maps through a kernel. The kernel will usually be a size of 2 x 2 or 3 x 3 in size to minimise the loss of information in the images.

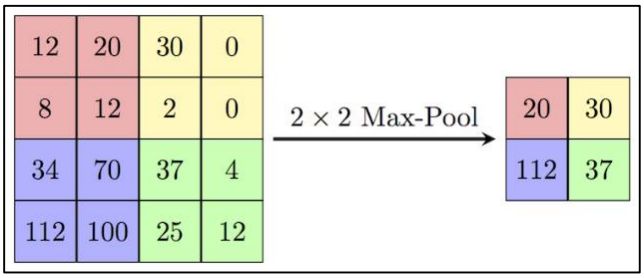


Fig. 3 An example of max pooling with a 2 x 2 kernel

Fig. 3 demonstrates a max pooling operation with a kernel of size 2 x 2 for a 4 x 4 feature map, with all the positions of the maximum value being recorded. Then, each recorded value is encoded into a 2 x 2 output array [11]. For example, the maximum value of the red patch is 20, so, the output array for the red patch is 20. The output array is the new feature map for this example.

2.2.1 The Fully Connected Layer

After the last convolution or pooling layer, the feature map will pass through to the fully connected layer known as artificial neural networks (ANN). This layer is made up of three layers which are the input layer, the output layer, and the hidden layers. The input and output layers of nodes are connected via one or more hidden layers of nodes. Neurons are assigned a numerical value called a bias, and the connections between neurons are called weights. Both biases and weights are randomly generated values, the same as the process employed in convolutional layers. Fig. 4 is an example of a fully connected layer. From Fig. 4, x_i is the input value, w_i is the weight value, b_i is the bias value, and z_i is the output value.

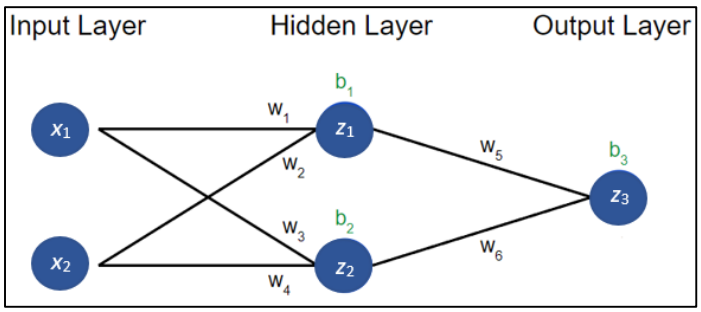


Fig. 4 An example of a fully connected layer

From Fig. 4, let a_1 be the neuron value of z_1. The value of a_1 is the weight sum value of the node and it calculated as

$$a_1 = x_1 w_1 + x_2 w_2 + b_1.$$

For n sized inputs, the general form of a_i is

$$a_i = \sum_{j=1}^{n} x_j w_j + b_i, \qquad i = 1,2, \ldots \tag{2}$$

From (2), it can be concluded that all the values of the neurons are simple linear functions. They cannot learn or recognise complex mappings from the data because of their limited complexity. To solve the problem, the nonlinear activation function is used to introduce nonlinear properties [12]. The activation function used in the hidden layers is the ReLU function, $f(z)$. Thus, the new value of z_i is

$$z_i = f(a_i) = f\left(\sum_{j=1}^{n} x_j w_j + b_i\right), \qquad i = 1,2, \ldots \tag{3}$$

For the output layer, the activation function is different from the hidden layers because this research classifies images using the output array. Thus, the softmax function is chosen as the activation function in the output layer because it can normalise the output of each neuron since the output value is limited to a value from zero to one and the sum of the outputs is equal to one [13]. The softmax function, $f(x)$ is defined as

Enhanced Knowledge in Sciences and Technology Vol. 4 No. 2 (2024) p. 161-170

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}, \qquad i = 1,2,\dots, \qquad x \in R, \tag{4}$$

where K is the number of outputs.

Lastly, for the CNN model, the fully connected layer does not necessarily contain hidden layers. However, the hidden layers will be added in this research because they can improve the performance of the process of feature extraction and increase the accuracy of image detection.

2.3 The Backpropagation of the CNN Model

The output array produced by the fully connected layer is the probability of the image classification, but the prediction of the CNN model is typically not as accurate as a human's prediction. To improve the accuracy of the CNN model, backpropagation, which is based on the Gradient Descent algorithm, is used to train the CNN model by computing the gradient of the loss function, which is the loss between the actual and predicted outputs. The purpose of backpropagation is to minimise the loss function by updating the values of each kernel, weight, and bias of every layer due to the gradient [14]. Since the loss function represents the loss between the actual output and the predicted output, the lower the loss function, the higher the accuracy.

The value calculated by the formula of backpropagation is updated with a learning rate through all training epochs. Training datasets are usually divided into batches for model training; each batch contains multiple samples, and the number of train data is known as the batch size. The batch size has an important impact on training, such as the total training time and the model's accuracy. The gradient of the loss function must be updated every time a batch is run, and this process is called an iteration. A training epoch represents a complete model training that includes updated values produced by the backpropagation. The learning rate is a critical hyperparameter that represents how fast the model learns [15]. Specifically, the learning rate is typically ranges between 0.0 and 1.0.

In this research, the mean squared error (MSE) function is the loss function, and its formula, L is defined by

$$L = \frac{1}{N}\sum_{i=1}^{N}(z_i - \widehat{z_i})^2, \tag{5}$$

where N is the number of epochs, and $(z_i - \widehat{z_i})$ is the loss between the actual and predicted output. $\widehat{z_i}$ is the expected value of z_i.

The gradient of the kernel, weights, and biases need to be calculated for the backpropagation process. By differentiating (5),

$$\frac{\delta L}{\delta z_i} = 2(z_i - \widehat{z_i}). \tag{6}$$

From (2) and (3), the derivative of z is

$$\frac{\delta z}{\delta a} = f'(a). \tag{7}$$

$\frac{\delta z}{\delta a}$ is also known as the slope of the activation function. For the ReLU function, the derivative is

$$f'(a) = \begin{cases} 1, & a \geq 0, \\ 0, & a < 0. \end{cases} \tag{8}$$

For the softmax function,

$$\frac{\delta z_i}{\delta a_j} = \begin{cases} z_i(1 - z_i), & if\ i = j, \\ z_i(z_j), & if\ i \neq j. \end{cases} \tag{9}$$

Let w_i be the value of kernel or weights. From (2), the derivative of a is

$$\frac{\delta a}{\delta w_i} = x_i. \tag{10}$$

Let b_i be the biases. From (2), the derivative of a is

$$\frac{\delta a}{\delta b_i} = 1. \tag{11}$$

Using the chain rule,

$$\frac{\delta z}{\delta w_i} = \sum_{i=1}^{K} [\, f'(a_i) \times x_i \,], \tag{12}$$

and

$$\frac{\delta z}{\delta b_i} = \sum_{i=1}^{K} f'(a_i)\,. \tag{13}$$

By multiple of (6) with (12) and (13), it can be concluded that

$$\frac{\delta L}{\delta w_i} = 2(z_i - \hat{z_i}) \times \sum_{i=1}^{K} [\, f'(a_i) \times x_i \,],$$

and

$$\frac{\delta L}{\delta b_i} = 2(z_i - \hat{z_i}) \times \sum_{i=1}^{K} f'(a_i)\,.$$

By using these equations, the gradient of weights, kernel, and biases are calculated and updated with the learning rate, α. The formulas of the updated gradient are:

•the new value of bias, b

$$b_i \leftarrow b_i - \alpha \frac{\delta L}{\delta b_i}.$$

•the new value of weights and kernel, w

$$w_i \leftarrow w_i - \alpha \frac{\delta L}{\delta w_i}.$$

In this CNN model, the convolution layer contains only the value of the kernel. The max pooling layer does not use the activation function and does not contain any biases and weights. However, the max pooling layer also has a backpropagation process because the gradient of the input is necessary for the previous layer. Fig. 5 shows the backpropagation process in this layer. From Section 2.1.2, the size of the feature map is reduced through the pooling operation, but the position of each maximum value is recorded. The size is reverted to the original size, and the maximum values are placed in the recorded positions, but the other values are replaced with the zero.

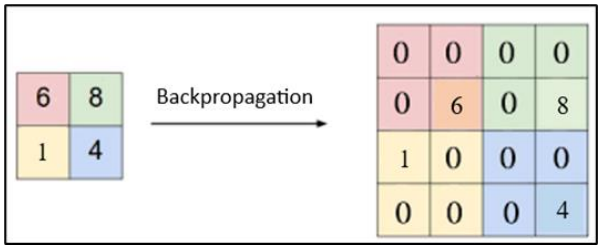


Fig. 5 The backpropagation of the max pooling layer

Enhanced Knowledge in Sciences and Technology Vol. 4 No. 2 (2024) p. 161-170

2.4 The Model Design

For this research, the dataset will be resized to a lower dimension of (128 x 128) because by doing so, the complexity of the computation and the computation time can be reduced. A set of 25,000 photos is divided into two groups, where 24,500 are used for model training, and 500 photos are allocated for testing the accuracy of the CNN model. The CNN model for image detection that will be built in Python consists of two convolution layers, two max pooling layers, and a fully connected layer. The size of the kernel used in the convolution layers is 3 x 3, and for the max pooling operation, the size of the kernel is 2 x 2. The fully connected layer contains three layers, which are an input layer, a hidden layer with 10 hidden neurons and an output layer. Accuracy is a measure of a CNN model's ability to predict outcomes. In this research, the accuracy of the CNN model is calculated using the test set of the datasets. The formula for accuracy is as follows:

$$\text{accuracy} = (\text{number of correct predictions}) / (\text{total number of predictions}) * 100\%$$

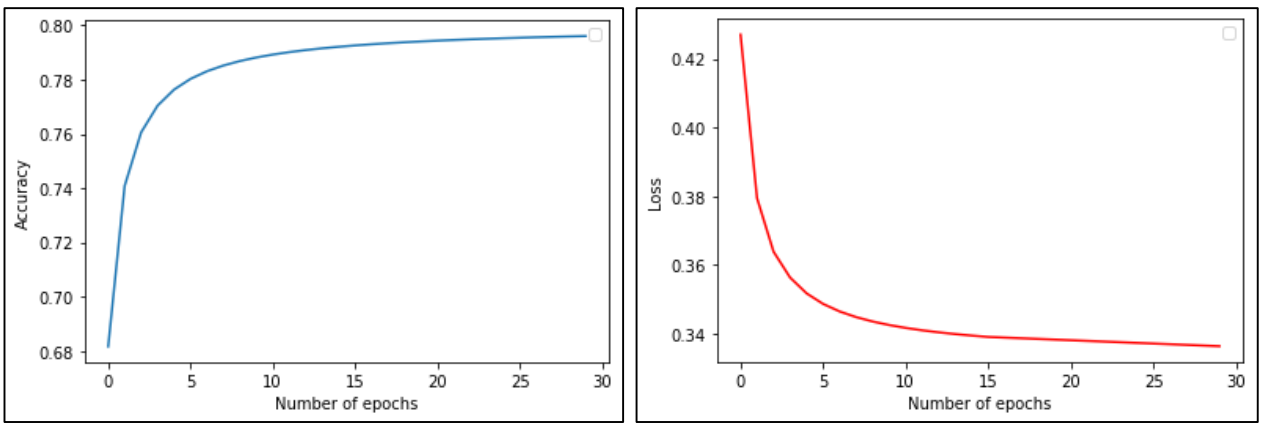
The CNN model may be overfitted during training and the loss will increase when the learning rate increases. If the learning rate is too small, it may cause slow convergence, and it cannot be too large, which may cause the optimization process to diverge. The CNN model performs well on the training set but performs poorly on the test set, and this is called overfitting. Thus, it tends to choose a lower learning rate. For simplicity, the learning rate, α is set at 0.001 in this research. The number of training epochs and the batch size of datasets are the manipulated variables in this research, where the number of training epochs will gradually increase from 1 to 30 and the batch size will increase from 8 to 128. The flowchart for the CNN model that was used in this research is illustrated in Appendix A.

3. Results and Discussion

For the research, the convolutional neural network (CNN) model designed for image detection will be built in Python without any built-in libraries for AI or machine learning. The analysis will be carried out for the relationship between the number of training epochs and the accuracy of the CNN model, as well as to find the most suitable batch size to achieve the highest accuracy. The CNN model will be tested on an Asus VivoBook S14 laptop with an AMD Ryzen 5 4500U processor and 8GB DDR4 RAM.

3.1 Performance of CNN Model with Different Number of Epochs

In this section, the relationship between the number of training epochs and the CNN model's performance are discussed. The learning rate of the CNN model was set to 0.001, and the batch size used was 32. A batch size of 32 is a suitable value for model training, a larger batch size will reduce cost time, but may cause the model to fail to capture features and patterns in the data [16]. The number of training epochs increased from 1 to 30, and the performance of the CNN model was tested by comparing the loss and model's accuracy.



(a) Accuracy **(b) Loss**

Fig. 6 The (a) accuracy and (b) loss of the CNN model with a number of training epochs

From Fig. 6, it can be seen at the beginning that the accuracy of the CNN model has increased rapidly from 68.1% to 78.8% whereas the loss has decreased quickly from 0.424 to 0.345. However, from the 11th training epoch to the 30th training epoch, the model 's accuracy has increased from 78.8% to 79.6% and the loss has decreased from 0.345 to 0.334. The CNN model after 10 times of training, the decrease in the rate of loss and increase in the rate of accuracy was very slow. Hence, it is suggested that the number of training epochs should

not exceed 10 as it will cause model overfitting. Other than that, exceeding 10 training epochs would also bring no significance to the model's performance.

3.2 Performance Comparison of Different Batch Sizes

In this section, the learning rate of the CNN model was set to a low level of 0.001 to avoid overfitting the model and the number of training epochs was set at 10. The manipulated variable, batch size was in multiples of 2, from 8 to 128. The performance was tested by comparing the loss and the accuracy of the CNN model.

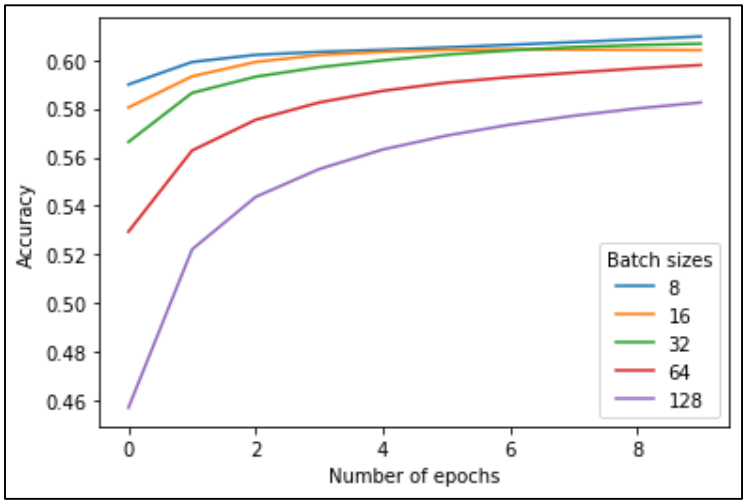


Fig. 7 The accuracy of the CNN model with different batch sizes

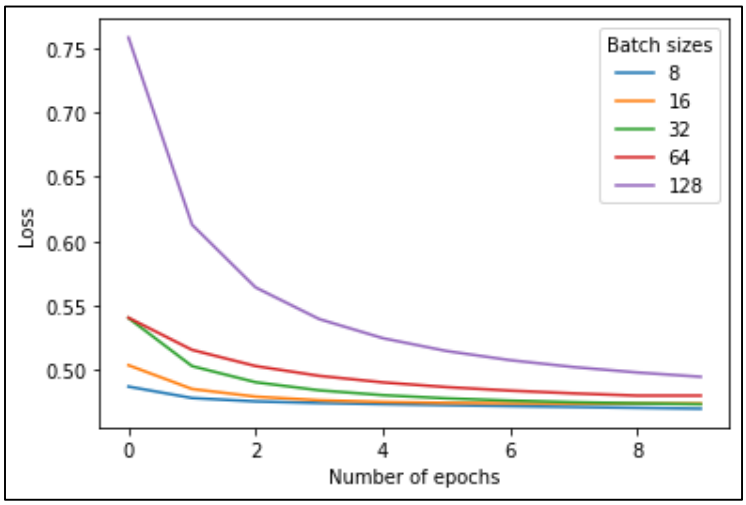


Fig. 8 The loss value of the CNN model with different batch sizes

Table 1 Performance of the CNN model with different batch sizes

Batch size	Loss value	Training time (min)	Accuracy (%)
8	0.47008	7.07375	61.03
16	0.47303	5.04302	60.55
32	0.47358	4.24631	60.68
64	0.48022	4.12761	59.59
128	0.49483	3.51924	58.25

The batch size has a great impact on the performance of the CNN model. As shown in Fig. 7, it is also observed that an increase in the batch size will cause the accuracy to decrease. The larger the batch size, the lower the initial accuracy value. It can be concluded that the higher batch size would have a higher initial loss value, and its loss value would always be higher than the lower batch size. From Table 1, it can be deduced that in terms of training time, a batch size of 128 performed the best as it took the shortest time to train. However, comparing the accuracy

and loss values of the batch sizes, a batch size of 8 outperformed the other batch sizes by returning the highest accuracy percentage and the lowest loss value. Thus, in consideration of all the three aspects, a batch size of 32 is the most recommended batch size.

4. Conclusion

In this research, starting from the first training epoch, the accuracy of the CNN model increases rapidly, while the loss decreases rapidly. However, the decrease in the rate of loss, and increase in the rate of accuracy was very slowly after training the CNN model for 10 times. Thus, it can be concluded that the number of training epoch should not exceed 10 to reduce the training time and avoid model overfitting. When the number of epochs used to train a convolutional neural network (CNN) model exceeds a certain limit, the trained model learns patterns specific to the training data set. The model performs well on the training set but performs poorly on the test set, and this is called overfitting. Besides that, when the batch size increases, the CNN model's accuracy decreases, and the loss of the model increases. From Table 1, a batch size of 128 demonstrated the most efficient training time but a batch size of 8 surpassed others by yielding the highest accuracy percentage and the lowest loss value. Therefore, in order to achieve better accuracy and lower loss values for a CNN model with a learning rate of 0.001, it is advised to use a smaller batch size for the CNN model, such as 8 or 16. It is advised to use a CNN model with a bigger batch size, such 64 or 128 though, given the training time. Furthermore, in order to save computational costs, this study recommends that the number of training epochs not exceed 10.

Acknowledgement

The authors would thank the Faculty of Applied Sciences and Technology, Universiti Tun Hussein Onn Malaysia for its support.

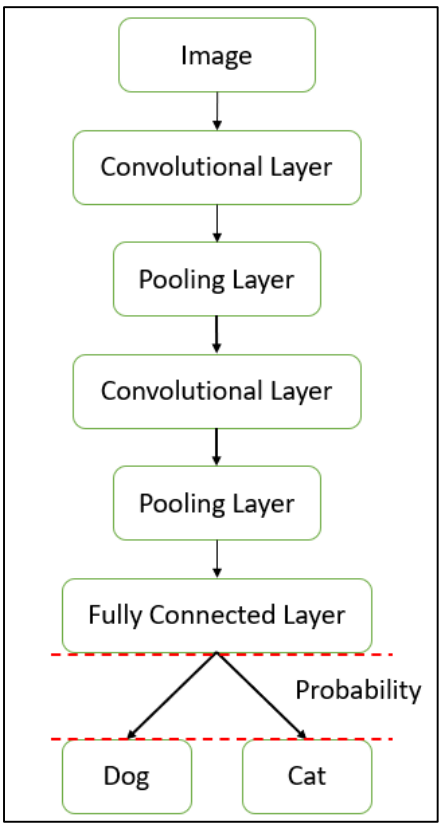
Conflict of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

The authors confirm contribution to the paper as follows: ***study conception and design:*** *Khoo Kang Wei, Lee Siaw Chong;* ***data collection:*** *Khoo Kang Wei;* ***analysis and interpretation of results:*** *Khoo Kang Wei, Lee Siaw Chong,* ***draft manuscript preparation:*** *Khoo Kang Wei. All authors reviewed the results and approved the final version of the manuscript.*

Appendix A: Flowchart of the CNN model



References

[1] Amit, Y., Felzenszwalb, P., & Girshick, R. (2020). *Object detection*. Computer Vision: A Reference Guide, pp. 1-9.
[2] Janiesch, C., Zschech, P., & Heinrich, K. (2021). *Machine learning and deep learning.* Electronic Markets, 31(3), pp. 685-695.
[3] Nwankpa, C., Ijomah, W., Gachagan, A., & Marshall, S. (2018). *Activation functions: comparison of trends in practice and research for deep learning*. arXiv preprint arXiv:1811.03378. b
[4] Will Cukierski. (2013). Dogs vs. cats. Kaggle. https://kaggle.com/competitions/dogs-vs-cats
[5] Nasteski, V. (2017). *An overview of the supervised machine learning methods*. Horizons. b, 4, 51-62.
[6] Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). *Convolutional neural networks: an overview and application in radiology*. Insights Imaging 9, pp. 611–629
[7] Springer International Publishing. (2020). *Deep learning vs. traditional computer vision*. [Infographic]. https://arxiv.org/abs/1910.13796
[8] Chauhan, R., Ghanshala, K. K., & Joshi, R. C. (2018). *Convolutional neural network (CNN) for image detection and recognition*. 2018 first international conference on secure cyber computing and communication (ICSCCC). IEEE. pp. 278-282.
[9] Szandała, T. (2021). *Review and comparison of commonly used activation functions for deep neural networks*. in Bhoi, A. K., Mallick, P. K., Liu, C. M., & Valentina, E. B. (Ed.). Bio-inspired neurocomputing. Singapore: Springer Nature, pp. 203-224.
[10] Soliman, N., Abd-Alhalem, S., El-Shafai, W., Abdulrahman, S., & Abd El-Samie, F. (2022). *An improved convolutional neural network model for DNA classification*. Computers, Materials and Continua, 70(3), pp. 5907-5927.
[11] Qian, R., Yue, Y., Coenen, F., & Zhang, B. (2016). *Traffic sign recognition with convolutional neural network based on max pooling positions*. 12th International conference on natural computation, fuzzy systems and knowledge discovery (ICNC-FSKD). IEEE. pp. 578-582.
[12] Sharma, S., & Athaiya, A. (2020). *Activation functions in neural networks. International Journal of Engineering Applied Sciences and Technology*, 4(12), pp. 310-316.
[13] Alabassy, B., Safar, M., & El-Kharashi, M. W. (2020). A high-accuracy implementation for softmax layer in deep neural networks. *15th Design & Technology of Integrated Systems in Nanoscale Era (DTIS)*. IEEE. pp. 1-6.
[14] Erb, R. J. (1993). *Introduction to backpropagation neural network computation.* Pharmaceutical research, 10, pp. 165-170.

[15] Shafi, S., & Assad, A. (2023). Exploring the relationship between learning rate, batch size, and epochs in deep learning: an experimental study. *In Soft Computing for Problem Solving: Proceedings of the SocProS 2022.* Springer Nature Singapore. pp. 201-209.
[16] Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. *In Neural Networks: Tricks of the Trade: Second Edition*. Berlin, Heidelberg: Springer Berlin Heidelberg. pp. 437-478.