Image Classification of CIFAR-10 Dataset using Convolutional Neural Network

Chan Ka Hing
The Univesity of Adelaide
kahing.chan@student.adelaide.edu.au

Abstract

The convolutional neural network is one of the most popular deep neural networks in image classification. It simplifies the neural network by reducing the number of parameters making it a good choice for image classification. In this paper, we build a convolutional neural network and perform image classification on CIFAR-10 dataset.

1. Introduction

Image classification is a classical task of machine learning. Convolutional neural network (CNN) is commonly used for image classification as it is much more computationally efficient than other neural networks. It is widely used in face detection, self-driving cars, handwritten character recognition, video classification, and so on. The convolutional neural network was first invented by Yann LeCun and his team in 1998 [3]. The idea comes from the study of the visual cortex of the animal brain [1]. The neurons in the visual cortex react only to visual stimuli in a restricted region called the receptive field [1]. Different neurons react to the image of different line orientations such as horizontal lines and vertical lines [1]. These observations inspired the development of the convolutional neural network.

In this paper, we use a convolutional neural network for image classification. We demonstrate our model on the CIFAR-10 dataset with a maximum validation accuracy of 71.2%. In the experiment, we study the performance of different techniques for preventing overfitting, namely dropout, L2 regularisation, and image augmentation. They all show good performance in reducing overfitting with increased validation accuracy. We also examine the effect of increasing filter numbers in convolutional layers and removing max pooling layers on the model accuracy.

The CIFAR-10 dataset contains 60000 images evenly distributed into ten classes [4]. The ten classes include cat, dog, frog, deer, and so on. Each image is a 32x32 colour image [4]. The dataset is divided into 50000 training images and 10000 test images [4]. The training images are further split into the training set and validation set in a ratio of 80% and 20% respectively. Therefore, the training, validation and testing set contain 40000, 10000, and 10000 images respectively. All three sets of images are sub-divided into mini-batches with 64 images in one batch for training and testing of the model.

2. Methodology

The methodology consists of three parts, CNN architectures, training of CNN model, and limitations and improvements.

2.1. CNN Architectures

A typical CNN architecture consists of the input layer, convolutional layers (followed by a ReLU activation function), pooling layers, fully connected layers, and the output layer [1]. The number of convolutional layers, pooling layers, and fully connected layers can vary depending on the structure. Figure 1 shows a typical CNN architecture [1]. First, the image is inserted from the input layer. As the image passes along the convolutional layers and the pooling layers, it reduces in size but increases in depth (due to more features extracted) [1]. At the last stage, the 2D or 3D image is flattened into a 1D array and fed into several fully connected layers. The fully connected layers is a regular feedforward neural network. The final output layer outputs the prediction in terms of regression value or class probabilities.

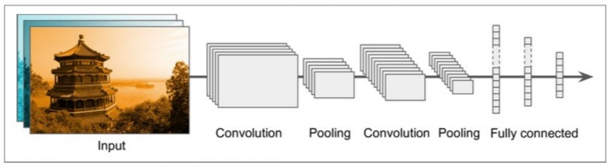

Figure 1. General CNN architecture (Geron 2019)

2.1.1 Convolutional Layers

The convolutional layer is the most important part of CNN architecture [1]. The neurons in the convolutional layer are not connected to every image pixel. [1] It is only connected to the pixels within a receptive field of a specific size [1]. The receptive field is confined by a filter which contains a set of parameters (weights) that need to be learnt. The filter has the same dimensionality (2D or 3D) as the image but is usually much smaller, for example, a 3 x 3 x 3 filter. It convolves over the image and computes the dot product between the image pixel and the weight at every overlapping position. Figure 2 shows a 5 x 5 x 3 filter convolves over the region x^r of the input image. The output h^r to a neuron in the next layer is calculated as follows:

$$h^r = \sum_{ijk} x^r_{ijk} W_{ijk} + b \tag{1}$$

i is the i-th row of the filter
j is the j-th column of the filter
k is the k-th depth of the filter
x^r_{ijk} is the image pixel at the i-th row, j-th column, k-th depth position
W_{ijk} is the weight on the filter corresponding to the i-th row, j-th column, k-th depth position
b is the bias term corresponding to that filter

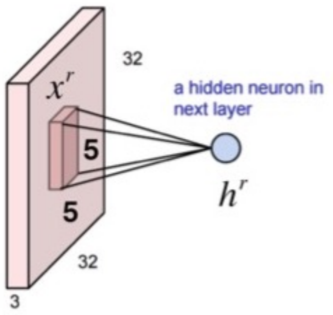

Figure 2. Demonstration of filter operation in a convolutional layer (Dick 2022)

The overall output is a feature map with reduced height and width. If we want to keep the same height and width as the previous layer, we can add zeros around the circumference of the input. This is called zero padding. Another important parameter is the stride. It is the step size used to move the filter. Convolutional filters usually have a stride of 1. In addition, multiple filters can be applied to the image to extract multiple features. The output is a stack of feature maps. In each layer, each filter creates one image by operating over the whole stack of feature maps (all channels) from the previous layer. The number of filters that operate on one layer determines the depth (number of channels) of the next layer. Each convolutional layer repeats the same process and extracts the features from low-level features to high-level features along the network.

2.1.2 Pooling Layers

The pooling layer reduces the size of the image and the number of parameters. It aims to reduce the computational load, memory usage and potentially mitigates overfitting [1]. There are several types of pooling operations such as max pooling and mean pooling. The most common type is max pooling using a 2 x 2 filter with stride of 2. At each location, the filter only outputs the maximum input value falling within the filter and drops other values. The max pooling operation is performed separately for each feature map within a stack. As a result, the output feature map stack has the same depth as the input stack.

2.1.3 Batch Normalisation

Batch normalization is a method to increase the training speed of the model and make it more stable. It is usually performed after the convolutional layer. It normalizes the layer's output by re-centring and re-scaling to give the same mean and variance. The formula for batch normalisation is as follows:

$$\hat{x_i} = \frac{x_i - \mu_B}{\sqrt{\sigma^2_B + \epsilon}} \tag{2}$$

x_i is the i-th data point in the batch
μ_B is the batch mean
σ^2_B is the batch variance
ϵ is a an arbitrarily small constant
$\hat{x_i}$ is the normalised form of x_i

The output y_i after batch normalisation is as follows:

$$y_i = \gamma\, \hat{x_i} + \beta \tag{3}$$

γ and β are the parameters to be learnt during training. Batch normalisation reduces the effect of the earlier layers on the current layer because the data are constrained to the same mean and variance, allowing the layers to learn more independently.

2.2. Training of CNN model

The training of the model involves the forward pass and the backward pass. In the forward pass, the image traverses through all neurons from the first layer to the last layer. The loss function calculates the loss of the output from the last layer. We use cross entropy loss as our loss function in this paper. Then, we perform stochastic gradient descent and backpropagation to update all the model weights in the backward pass. A combination of a forward pass and a backward pass is one training iteration.

2.2.1 Softmax Function and Cross Entropy Loss

The Softmax Function normalises the neural outputs to a probability distribution of the predicted classes. The softmax probability is calculated as follows:

$$p_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad (4)$$

z_i is the i-th raw output
K is the number of raw outputs
z_j is the j-th raw output where $j = 1, 2, ..., K$
e is the exponential term
p_i is the softmax probability of the i-th output

Each of the softmax probability is fed into the cross entropy loss function to calculate the loss. The cross entropy loss is as follows:

$$L_{CE} = \sum_{i=1}^{n} t_i \; log(p_i) \quad (5)$$

t_i is the truth label
p_i is the softmax probability
n is the number of classes
L_{CE} is the cross entropy loss function

2.2.2 Back propagation

Backpropagation calculates the gradient of the loss function with respect to each model weight using the chain rule. It starts with the last layer and goes backwards. Once we calculate the gradient with respect to the output at the last layer, we are able to calculate the gradient with respect to all the weights in the network. Since the output of one layer is a function of the output of the previous layer and the corresponding model weights, we can apply chain rule to find the gradient with respect to all the model weights between these two layers. The process repeats from the last layer to the first layer. The model weights are updated using stochastic gradient descent.

2.3. Limitations and Improvements

CNN is a complex neural network with a massive number of parameters, making it prone to overfitting. The model can easily overfit the training images and fail to generalise and make a prediction for unseen images. Several methods to reduce overfitting include dropout layers, regularisation and image augmentation. We study the performance of these methods on overfitting in three of our experiments.

2.3.1 Dropout

Dropout is a technique to avoid overfitting in a deep neural network like CNN. It can be performed at the input layer and the hidden layers. At each training step, every neuron within a layer has a probability p of being temporarily dropped out [1]. The dropped-out neuron is ignored and not updated in that training step, but it may be active in other training steps [1]. The probability p is called the dropout rate [1]. After training, all neurons will become active and used for making predictions. Figure 3 shows the dropout operation in two of the layers.

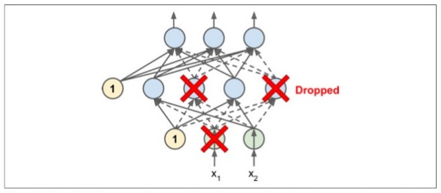

Figure 3. Dropout operation in the layers for avoiding overfitting (Geron 2019)

2.3.2 L2 Regularisation

L2 Regularisation is a technique to simplify the model by adding an extra term to the loss function. This prevents the model weights from becoming too large and potentially forces some of the weights to zero [2]. As a result, it removes some noise input and helps prevent overfitting [2]. The extra term added to the loss function is as follows:

$$L_{new} = L_{original} \; + \; \lambda \, W^T W \quad (6)$$

L_{new} is the new loss function
$L_{original}$ is the original loss function
λ is the weight decay hyperparameter
$W^T W$ is the L2 norm of the model weights

L2 regularisation can also be known as weight decay, as the model weights are reduced or decay during training [2]. The weight decay hyperparameter λ can be tuned to minimize overfitting.

2.3.3 Image Augmentation

In the training set, the object in the image is at a particular orientation and shape. The computer only learns from the training images in the image classification task. Therefore, it may not recognise the object in an unseen image if the orientation and shape are different. This problem can be solved by image augmentation. It performs different transformations on the training images, such as vertical flips, horizontal flips, random rotation, random crop, and so on. As a result, the model is more tolerant to variations in the object's position, orientation, and size [1]. It generalises the unseen images better and reduces overfitting.

3. Experiments and analysis

The experiments and analysis section consists of two parts, general analysis and the experiments.

3.1. General Analysis

The baseline model has 3 hidden layers. Each hidden layer has one convolutional layer and one max pooling layer. Batch normalisation is performed in each hidden layer. We demonstrate our baseline model using the CIFAR-10 dataset with a learning rate of 0.001 for 50 iterations. As shown in Figure 4, both training and validation accuracy shows a stable convergence after 50 iterations. The maximum training and validation accuracy are 100% and 71.3% respectively. The result shows an overfitting phenomenon which might be due to the complex structure of CNN. Since a large number of features are extracted from the training images in the complex structure, the model fits the training images too well and fail to generalise and make a highly accurate prediction for the unseen validation images.

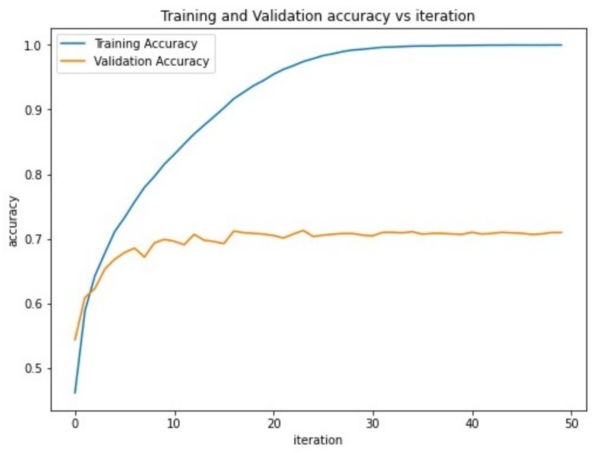

Figure 4. The training and validation accuracy versus iteration on CIFAR-10 dataset with learning rate of 0.001 for 50 iterations. The figure shows an overfitting phenomenon.

3.2. Experiment 1

In this experiment, we perform the dropout at the second and third hidden layers of the model with three different dropout probabilities. The experiment aims to study the performance of dropout in reducing overfitting and find the best probability among these three probabilities which maximizes the accuracy. Table 1 shows the maximum training and validation accuracy for the baseline model and the models with different dropout probabilities. Figures 4,5,6, and 7 show the training and validation accuracy versus the iteration for the baseline model and the models with dropout probabilities of 0.3, 0.5, and 0.7 respectively. As illustrated

Experiment 1 (Dropout in 2nd and 3rd hidden layers)		
Dropout Probability (p)	Maximum Training Accuracy (%)	Maximum Validation Accuracy (%)
Baseline model (no dropout)	100	71.3
p = 0.3	82.6	76.8
p = 0.5	74.6	73.4
p = 0.7	66.6	67.5

Table 1. The maximum training accuracy and validation accuracy of baseline model and different dropout probabilities models with learning rate = 0.001 for 50 iterations.

in Table 1, both dropout probability 0.3 and 0.5 show an increase in maximum validation accuracy. They also show a reduction in overfitting in figures 5 and 6. The model with a dropout probability of 0.3 has the highest maximum validation accuracy (76.8%). The model with a dropout probability of 0.5 has a lower maximum validation accuracy (73.4%). The model with a dropout probability of 0.7 has the lowest maximum validation accuracy (67.5%) which is even lower than the baseline model (71.3%). The dropout probability of 0.7 might be too large which deactivates too many neurons in the layer, lowering the model performance. As a result, dropout successfully reduces overfitting and increases accuracy due to the deactivation of unnecessary neurons in the hidden layers. The best dropout probability for our model is 0.3.

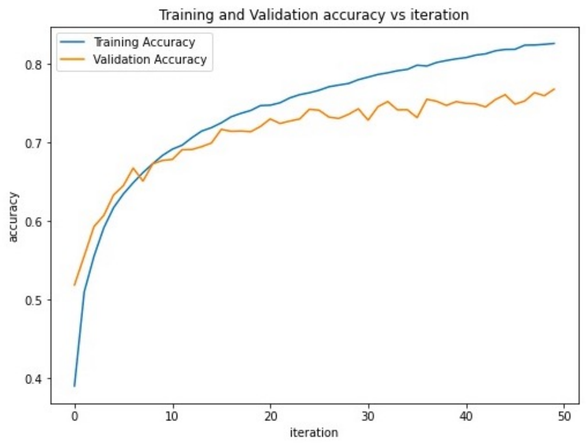

Figure 5. The training and validation accuracy versus iteration with dropout probability = 0.3, learning rate = 0.001 and 50 iterations.

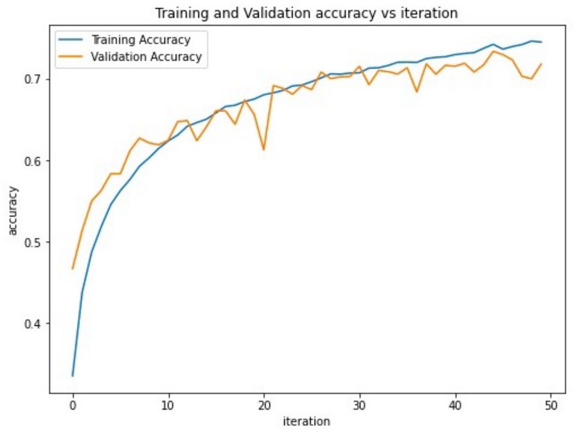

Figure 6. The training and validation accuracy versus iteration with dropout probability = 0.5, learning rate = 0.001 and 50 iterations.

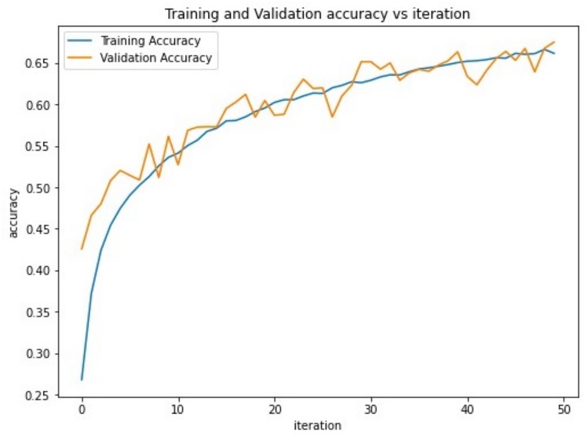

Figure 7. The training and validation accuracy versus iteration with dropout probability = 0.7, learning rate = 0.001 and 50 iterations.

3.3. Experiment 2

In this experiment, we perform the L2 regularisation with three different weight decay parameters. The experiment aims to study the performance of L2 regularisation in reducing overfitting and find the best weight decay parameter among these three parameters which maximizes the accuracy. Table 2 shows the maximum training and validation accuracy for the baseline model and the models with different weight decay parameters. Figures 4,8,9 and 10 show the training and validation accuracy versus the iteration for the baseline model and the models with weight decay parameters of 0.01, 0.05, and 0.1 respectively. As illustrated in Table 2, all three L2 regularisation models show an increase in maximum validation accuracy. They also show a reduction in overfitting in figures 8,9 and 10. The model with a weight decay parameter of 0.01 has maximum validation accuracy of 72.5%. As the weight decay parameter increases, the validation accuracy increases. The model with a weight decay parameter of 0.1 has the highest maximum validation accuracy of 77.5%. As a result, L2 regularisation successfully reduces overfitting and increases accuracy due to zeroing unnecessary weights by adding the extra term in the loss function. The best weight decay parameter for our model is 0.1.

Experiment 2 (L2 Regularisation with different weight decay parameters)		
weight decay	Maximum Training Accuracy (%)	Maximum Validation Accuracy (%)
Baseline model (no regularisation)	100	71.3
0.01	100	72.5
0.05	97.2	76.5
0.1	86.3	77.5

Table 2. The maximum training accuracy and validation accuracy of baseline model and different weight decay parameters models with learning rate = 0.001 for 50 iterations.

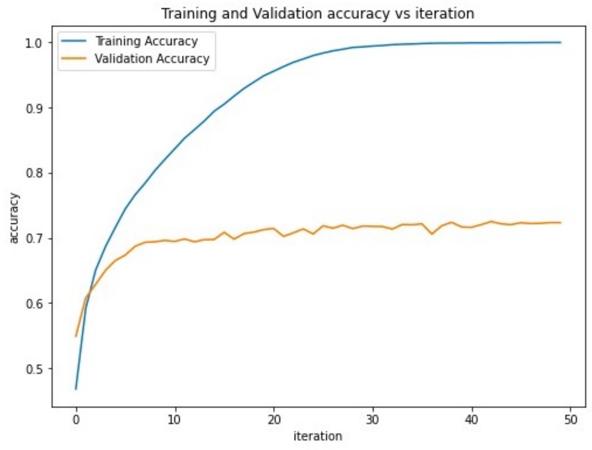

Figure 8. The training and validation accuracy versus iteration with weight decay = 0.01, learning rate = 0.001, and 50 iterations.

3.4. Experiment 3

In this experiment, we perform image augmentation with horizontal flip and 30 degree rotation on the dataset. The experiment aims to study the performance of image augmentation (horizontal flip and 30 degree rotation) in reducing overfitting. Table 3 shows the maximum training and

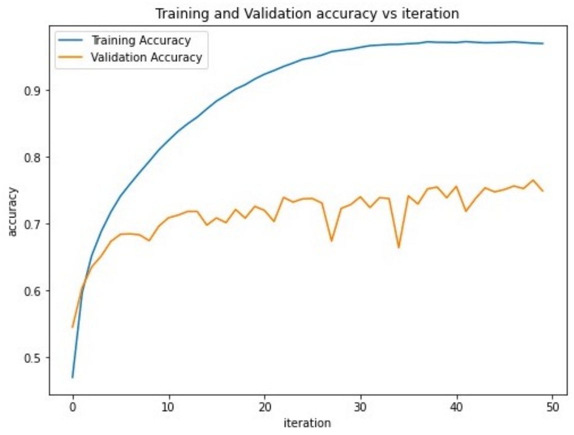

Figure 9. The training and validation accuracy versus iteration with weight decay = 0.05, learning rate = 0.001, and 50 iterations.

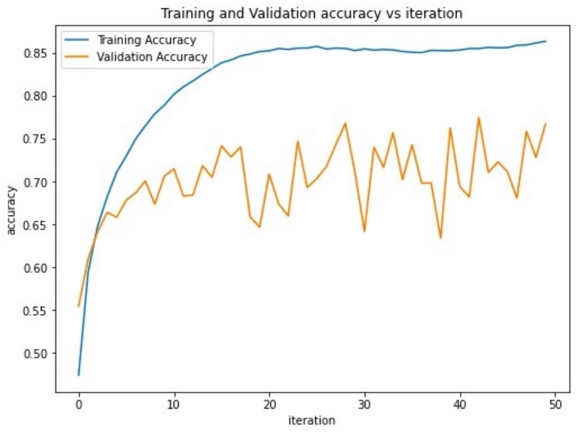

Figure 10. The training and validation accuracy versus iteration with weight decay = 0.1, learning rate = 0.001, and 50 iterations.

validation accuracy for the baseline model and the image augmentation model. Figures 4 and 11 show the training and validation accuracy versus the iteration for the baseline model and the image augmentation model respectively. As illustrated in Table 3, image augmentation increases the maximum validation accuracy from 71.3% to 73.8%. It also reduces the overfitting as shown in figures 11. Therefore, image augmentation (horizontal flip and 30 degree rotation) successfully reduces overfitting and increases accuracy by producing more generalised training images for the model.

3.5. Experiment 4

In this experiment, we study three sets of filter numbers, namely set A, B and C, in the convolutional layers. Set A has filter numbers 16, 32, 64 in the 1st, 2nd, and 3rd convolutional layers respectively. Set B (64, 128, 256) has filter numbers four times more than set A. Set C (256, 512, 1024) has filter numbers four times more than set B. The experiment aims to study the effect of increasing number of filters on the model accuracy. Table 4 shows the maximum training and validation accuracy for the models with these three filter sets. Set A has maximum training and validation accuracy of 82.2% and 69.1% respectively. As the number of filters increases, both the training and validation accuracy increase because more features were extracted for training the model. Set C has the highest maximum training and validation accuracy (100% and 74.5% respectively). Figures 12,13 and 14 show the training and validation accuracy versus the iteration for the models with filter set A, B and C respectively. The training accuracy of set A does not converge in 50 iterations as shown in figure 12. The training accuracy of set C converges to 100% faster than set B as shown in figures 13 and 14 due to more filters. Therefore,

Experiment 3 (Image augmentation)		
	Maximum Training Accuracy (%)	Maximum Validation Accuracy (%)
Baseline model (no image augmentation)	100	71.3
Image augmentation (horizontal flip and 30° rotation)	78.5	73.8

Table 3. The maximum training accuracy and validation accuracy of baseline model and image augmentation model with learning rate = 0.001 for 50 iterations.

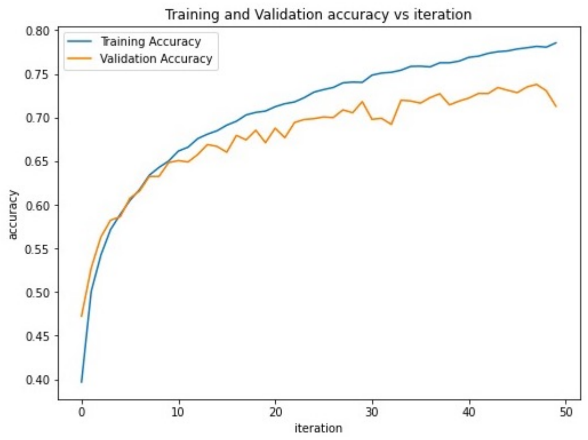

Figure 11. The training and validation accuracy versus iteration of image augmentation model with learning rate = 0.001, and 50 iterations.

Experment 4 (Different number of filters)			
Set	Number of filters (1st, 2nd, 3rd convolutional layer)	Maximum Training Accuracy (%)	Maximum Validation Accuracy (%)
A	16, 32, 64	82.2	69.1
B	64, 128, 256	100	70.8
C	256, 512, 1024	100	74.5

Table 4. The maximum training accuracy and validation accuracy of models with different number of filters at the learning rate of 0.001 for 50 iterations.

increasing the number of filters in convolutional layers will increase both the training and validation accuracy.

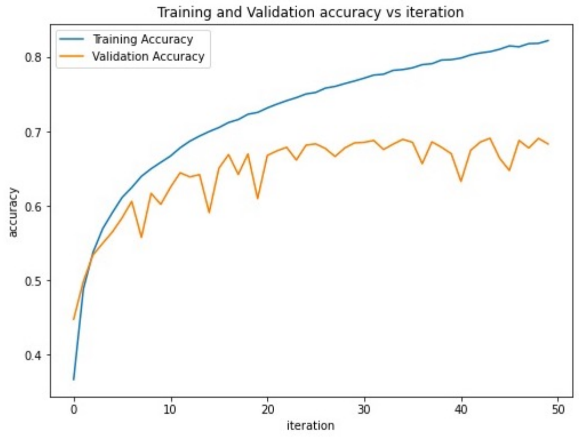

Figure 12. The training and validation accuracy versus iteration of model with number of filters 16, 32, 64 in the 1st, 2nd, and 3rd convolutional layer respectively at the learning rate of 0.001 for 50 iterations.

3.6. Experiment 5

In this experiment, we compare the difference between baseline model with and without max pooling layers. The experiment aims to study the effect of max pooling layers on the model accuracy. Table 5 shows the maximum training and validation accuracy for the baseline models with and without max pooling layers. Both models have the same maximum training accuracy (100%). However, the model with max pooling layers has a higher maximum validation accuracy (71.6%) than the model without max pooling layers (67.0%). It is because the max pooling layers select the most prominent feature in a feature map and remove other noises. This gives a better generalisation and helps reducing overfitting.

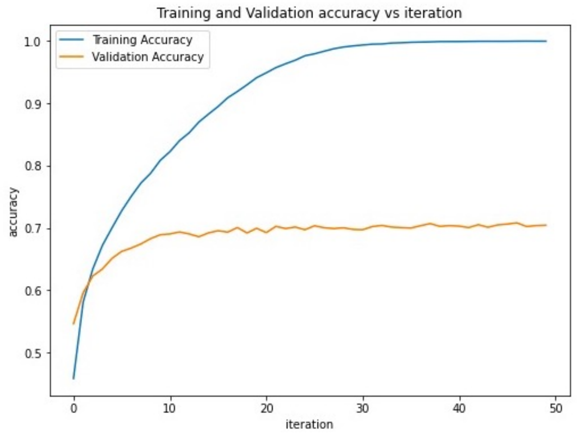

Figure 13. The training and validation accuracy versus iteration of model with number of filters 64, 128, 256 in the 1st, 2nd, and 3rd convolutional layer respectively at the learning rate of 0.001 for 50 iterations.

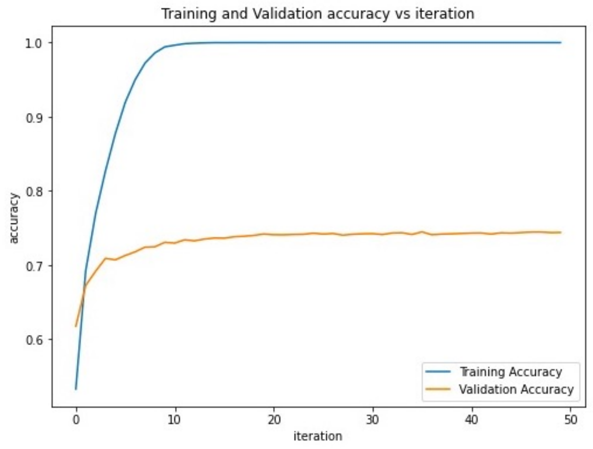

Figure 14. The training and validation accuracy versus iteration of model with number of filters 256, 512, 1024 in the 1st, 2nd, and 3rd convolutional layer respectively at the learning rate of 0.001 for 50 iterations.

3.7. Testing Accuracy

We choose the CNN model with L2 regularisation (weight decay = 0.1) to calculate the testing accuracy since it has the highest validation accuracy among all the experiments. The Testing Accuracy is 77.49%.

4. Code

https://github.com/ChanKaHing/CNN.git

Experiment 5 (Max pooling layers)		
	Maximum Training Accuracy (%)	Maximum Validation Accuracy (%)
Baseline model (with max pooling layers)	100	71.6
Baseline model (without max pooling layers)	100	67.0

Table 5. The maximum training accuracy and validation accuracy of models with and without max pooling layers at the learning rate of 0.001 for 50 iterations.

5. Conclusion

This paper presents a Convolutional Neural Network model for image classification. After the experiment and analysis, there are three main findings. First, dropout, regularisation, and image augmentation show good performances in reducing overfitting and the hyperparameters can be tuned through experiments to maximise the accuracy. Second, increasing the number of filters in convolutional layers will increase the model accuracy by extracting more features. Third, adding max pooling layers to the model increases the model accuracy by reducing model complexity and overfitting. Future work may include further improving the accuracy by increasing the number of layers of the model such as using Alexnet (8 layers) and VGG-16 (16 layers). We can also explore the ResNet which uses skip connections to build CNN up to hundreds or even thousands of layers.

References

[1] Geron, A 2019, *Hands-on machine learning with Scikit-learn, Keras, and TensorFlow: concepts, tools, and techniques to build intelligent systems,* Second edition., O'Reilly Media, Sebastopol.

[2] Zafar, I, Tzanidou, G, Burton, R, Patel, N Araujo, L 2018, *Hands-On Convolutional Neural Networks with TensorFlow: Solve Computer Vision Problems with Modeling in TensorFlow and Python*, Packt Publishing, Limited, Birmingham.

[3] Lecun, Y, Bottou, L, Bengio, Y Haffner, P 1998, 'Gradient-based learning applied to document recognition', Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324.

[4] Krizhevsky, A 2009, 'Learning Multiple Layers of Features from Tiny Images', MSc thesis, University of Toronto, Toronto.

[5] Dick, A 2022, *Deep Learning Fundamentals lecture 3: Convolutional Neural Network*, lecture notes, The University of Adelaide, delivered 19 September 2022.