Building Neural Network for Multi-class Classifcation Using Cifar10 Dataset

Asala Aljazaery

09 May 2022

University of Sunderland

Contents

1 Introduction

Machine learning has been used recently to represent vast amount of data in a suitable form depending on the task, such as email spam detection. Artificial neural networks (ANNs) are a subcategory of machine learning (ML) and deep learning (DL) algorithms. DL has succeeded in areas where ML has failed. For example, ML lacks speed, high performance, and intelligence. In contrast, DL has outstanding data processing with high capabilities (Wang *et al.*, 2020). A convolutional neural network (CNN) is a DL method known for solving problems and interpreting nonlinear and convoluted data without human interference(Hosny *et al.*, 2018), and is well known for image recognition tasks (Wang *et al.*, 2019). In this report, CNN will be used to classify the Cifar10 dataset, containing 60000 images and ten classes (also known as labels) (Krizhevsky, 2009). Each image belongs to one class, such as an airplane or automobile. This type of task is called multi-class classification because the purpose of this model is to classify the images based on their classes(Brownlee, 2020a). These networks can be used for other fields and areas where computer vision can be employed, such as healthcare, academic, and practical industrial applications. Using these networks will lead to a decrease in human intervention and an increase in accuracy by minimizing the error rate. For example, we have a shortage of professionals in the medical field and a high workload pressure. However, using these technologies will provide better health services for society, decrease the workload pressure, detect rare diseases that can not be seen with bare eyes, and find new areas to explore in the study, research, and many new findings(Hosny *et al.*, 2018). CNN has already achieved high accuracy in some fields, such as cardiology and dermatology(Wang *et al.*, 2019).

This report will focus on the model architecture to increase the network capacity and accuracy while minimizing the network computing time to achieve better results. First, present an overview of the basics of CNN and how it works and summarise some methodologies for choosing essential properties in the model. Then, prepare the dataset for the network. Then, I will start implementing some experiments with different types of model architectures. Furthermore, using random search to check the best accuracy achieved with the dataset. In addition, results will be presented, followed by a discussion on deep network architecture and a conclusion for our experiments. Lastly, discussing ethical areas for using AI in society.

2 Convolutional Neural Networks

CNNs are similar to biological neural networks (Liu *et al.*, 2015), created to mimic the human eye in extracting information and the brain in learning and solving problems. CNN was established based on functions that deliver outcomes or decisions based on their job, making the network function better than the human eye in some areas, such as detecting tumours' in medical images(Wang *et al.*, 2019). Nevertheless, CNN evolution has met GPUs limitation, which reduced the network's performance. Thus, researchers have constantly concentrated on enhancing the GPUs performance (Cheng *et al.*, 2018). CNN was designed based on the nature of the human brain and committed good results in identifying patterns and visualising the raw pixels with some training (Hosny *et al.*, 2018). Back in the 1960s, two researchers called Hubel and Wiesel realised that the cells in the cat eyes that visualise patterns move in a hierarchical mechanism while collecting data and processing it. Based on their finding, they presented the receptive field. Then in 1980, neurocognition was initiated, using neurons and local connectivity in the hierar-

chical structure of the model. Both concepts are the basic structure of CNN. Primarily, CNN was used for handwritten digit recognition. However, LeCun has endorsed that it can be used for other purposes and created the first CNN model called "LeNet-5". There were numerous challenges in the earlier phases of designing CNN, such as the number of layers inside the model and limitations due to the graphical processing unit(GPU) capability that limited the model performance. As a result, researcher lost their interest in it until Hinton et al. raised a new approach to solve the limitation problem by introducing multiple hidden layers. Those hidden layers excelled in the model learning feature. The researcher Krizhevsky, who collected the Cifar10 dataset, came up with a new CNN model called "AlexNet". AlexNet was employed for an image classification competition called ImageNet large-scale visual recognition challenge in 2012. Furthermore, it was in the top 5 winners, and the error rate was only 15.3%. When LeCun published a review paper regarding CNN's essential principles and the importance of deep learning, CNN acquired the researcher's engagement back. Later, different models were raised, such as the Visual geometry group and GoogleNet. They mainly focused on expanding the depth and width of the network to enhance the network capacity when it comes to complex images with many features. Therefore, the image was split into small patches, and each feature was assessed independently. ResNets models optimised the feature extracting technique by concentrating on the most valuable features and avoiding the noise in the images to minimise feature bottlenecks. DenseNet was introduced with short routes between layers to link and send data between layers (Wang *et al.*, 2019).

3 Introducing Methodology with Experiments

In building any computer science solution, These phases are required: first is to analyse the problem by exploring the dataset and what we can do and prepare the dataset. Then, begin developing the algorithm or the solution, which is the neural network. After that, the implementation phase is training the model, testing and checking our solution. Finally, we make any Maintenance after examining the solution to improve it. The main challenge here is to analyse an extensive dataset of images efficiently, give a precise classification of the images and uncover the connection in each class. CNN was used for the simple structure, fewer training parameters, and more adaptability, and it is well known for image classification accuracy (Liu *et al.*, 2015). In the classification problem, we predict a class (label) for the input by calculating the probability of belonging to each class already declared to the network (Brownlee, 2020b). Neurons connect CNN. Each neuron has weight, calculated with a non-linear activation function, Similar to the biological neuron Chartrand *et al.* (2017).

The CNN architecture Normally consists of convolutional layers, subsampling/ pooling layers and the fully connected layers. Each convolutional layer has multiple kernels that move across the image(input) to read it and extract information. These kernels are usually small, \$3 $\times$3 to split the image into patches and take one patch each time. Each kernel illustrates one feature exclusively, and then the output will be held in the feature maps(Chartrand *et al.*, 2017) (Figure 3.2). In CNN, the rectified linear unit (ReLU) is the most used activation function in the layers. Responsible for the forward pass, selecting the features from the feature maps to be sent to the following layers. Depending on the calculation, Relu will choose the most probability features and send them to the following layers (Figure 3.3).

The max-pooling function will be applied to the feature maps in the pooling layer. So each feature map will be downsized to a 2×2 dimension by taking the features with the

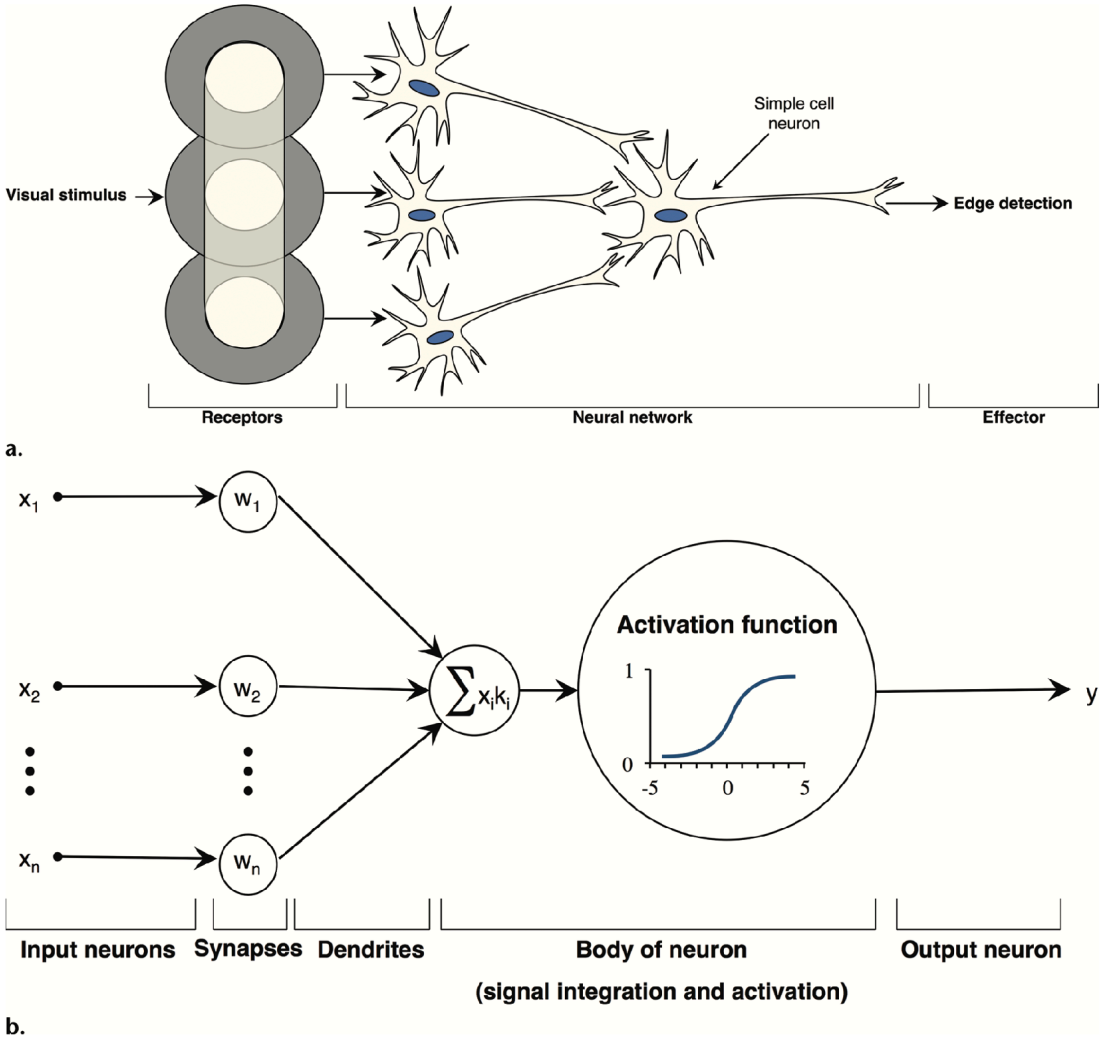


Figure 3.1: Biological neurons and artificial neuron(Chartrand *et al.*, 2017).

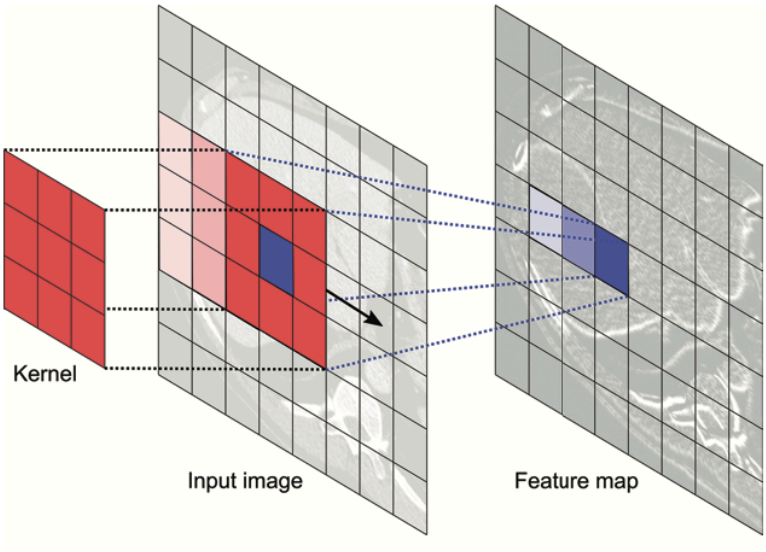


Figure 3.2: Kernels extracting features from the image (Chartrand *et al.*, 2017).

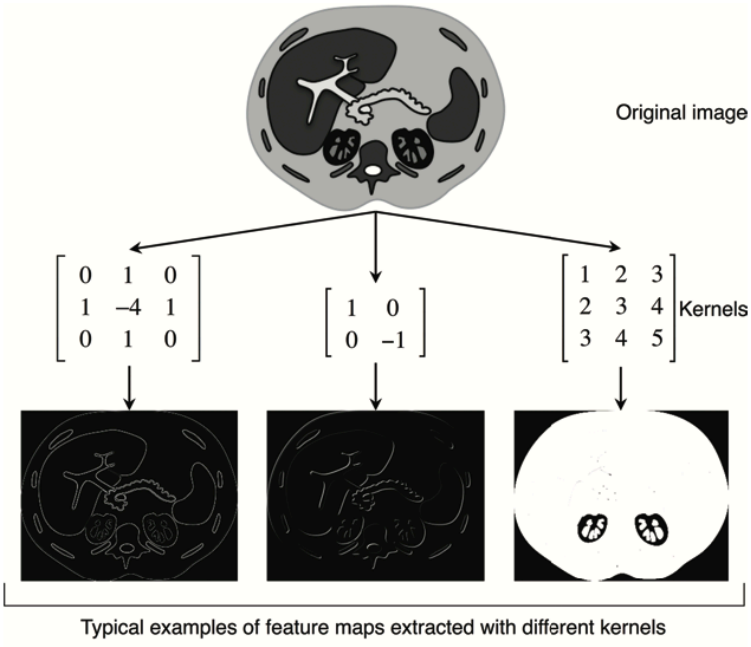


Figure 3.3: Example of convolution operation on an image (Chartrand *et al.*, 2017).

highest weights, as it is considered necessary in the image. Later, all feature weights will be re-arranged in a one-dimensional array and connected with all previous layers in the fully connected layer. Lastly, The softmax function in the last layer will calculate the output to provide probabilities of the different classes and classify the images(Chartrand *et al.*, 2017) (Figure 3.4). When the network gives N output for each class, softmax normalizes the outputs by converting the values of the weights to probabilities that all sum to one, and each one represents the probability of each class (Brownlee, 2020b).

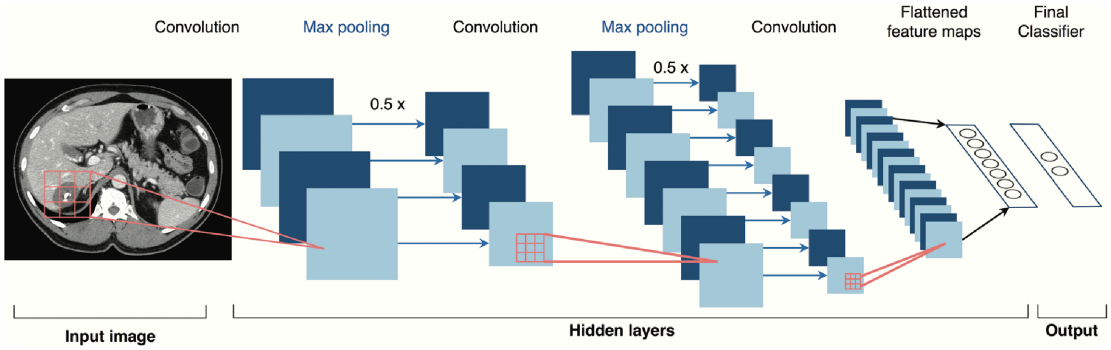


Figure 3.4: Convolutional neural network (Chartrand *et al.*, 2017).

The loss function will be used for the learning feature in the model, used for calculating the difference between the forecast results and the actual results to optimize the decision parameters based on that. Then, these parameters are optimized with gradient descent algorithms. We keep repeating these steps till we achieve the minimum amount of loss (Figure 3.5) (Chartrand *et al.*, 2017).

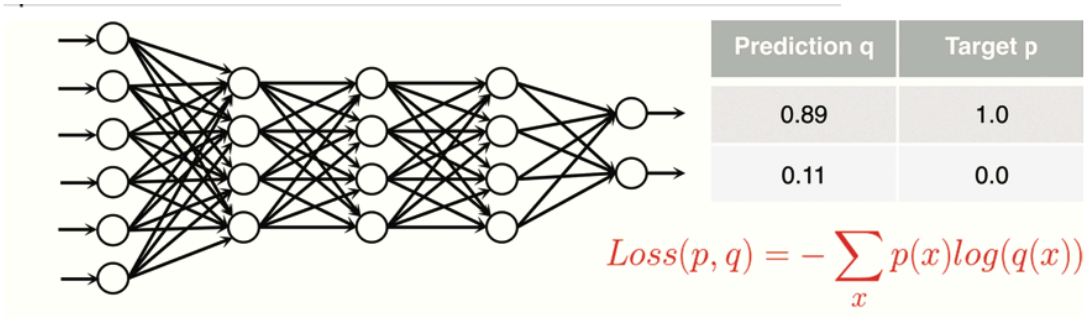


Figure 3.5: Parameters adjusted with the loss function (Chartrand *et al.*, 2017).

TensorFlow will be used as a software framework to build and train the model. Before building the model, we need to pre-process the data by ensuring that we have all the images of the same size or dimensions and use the normalization technique to make the model more efficient while processing. First, these images were normalized by dividing each pixel by 255 because each pixel ranges from 0 to 256 in the image; when the values are more minor, computation becomes faster and easier (Ponraj, 2021). Then the data types will be changed to "Float32" to make it easier for the model to process the data. After that, the dataset will be split into training and testing sets to train and test the model with different images. In Cifar10, each image size is $32 \times 32 \times 3$ pixels, where 3 represents the colours in each image (Figure 3.6).

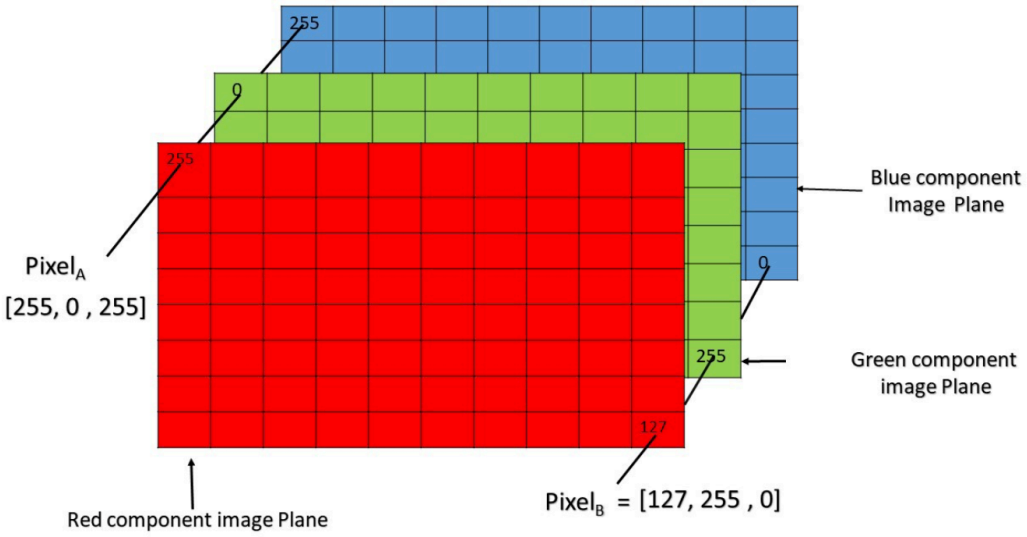


Figure 3.6: RGB(Chartrand *et al.*, 2017).

Scale the features is a crucial step for normalising our dataset. It normalises the features range between (0, 1) (Figure 3.7). Scaling will enhance the model learning feature, speed and performance. Later, all images must be in the same shape $32 \times 32 \times 3$. Finally, presenting a subplot for plotting 10 samples from our dataset (Figure 3.8).

```
array([[[0.23137255, 0.24313725, 0.24705882],
        [0.16862745, 0.18039216, 0.17647059],
        [0.19607843, 0.18823529, 0.16862745],
        ...,
        [0.61960784, 0.51764706, 0.42352941],
        [0.59607843, 0.49019608, 0.4       ],
        [0.58039216, 0.48627451, 0.40392157]],

       [[0.0627451 , 0.07843137, 0.07843137],
        [0.        , 0.        , 0.        ],
        [0.07058824, 0.03137255, 0.        ],
        ...,
        [0.48235294, 0.34509804, 0.21568627],
        [0.46666667, 0.3254902 , 0.19607843],
        [0.47843137, 0.34117647, 0.22352941]],

       [[0.09803922, 0.09411765, 0.08235294],
        [0.0627451 , 0.02745098, 0.        ],
        [0.19215686, 0.10588235, 0.03137255],
        ...,
        [0.4627451 , 0.32941176, 0.19607843],
        [0.47058824, 0.32941176, 0.19607843],
        [0.42745098, 0.28627451, 0.16470588]],

       ...,
...
        [0.70196078, 0.55686275, 0.34117647],
        ...,
        [0.84705882, 0.72156863, 0.54901961],
        [0.59215686, 0.4627451 , 0.32941176],
        [0.48235294, 0.36078431, 0.28235294]]])
```

Figure 3.7: Feature scaling.

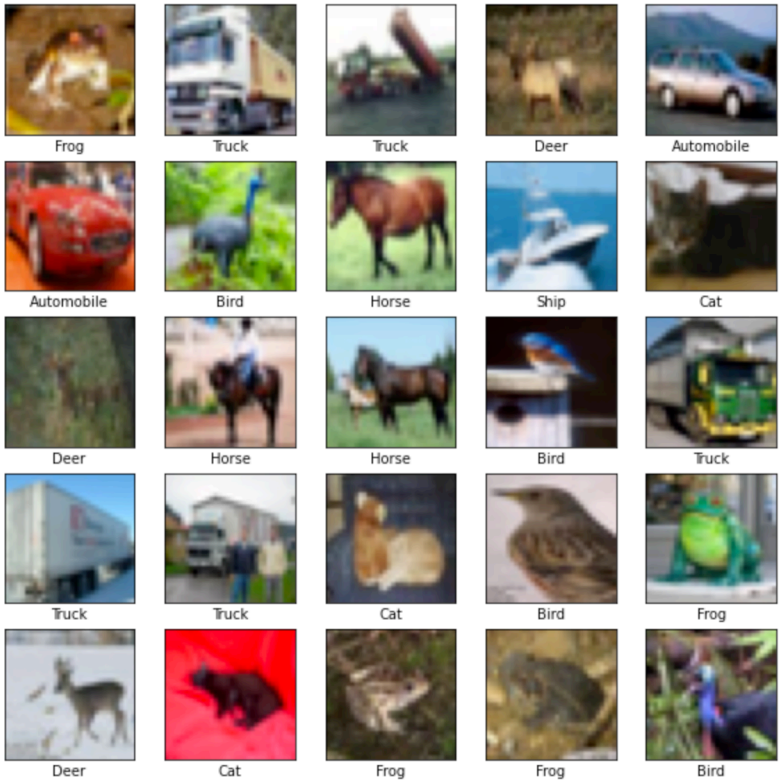


Figure 3.8: Plotting the dataset with labels

3.1 Experiments

There will always be room for CNN model architecture improvement. In this experiment, the main focus will be on higher accuracy, minimal training time, and simple structure. All models were trained with a fixed number of echos = ten and “adam”as an optimizer. Filters were in the range of 32 to 128 for each model. The kernels size was (2, 2) and (3, 3).

3.1.1 Baseline model summary

For the baseline model, the structure will be as follows (Figure 3.10):

3.1.2 Best model summary

Using Random Search to find the best model, and how much accuracy I can achieve with the dataset (Figure 3.11).

3.1.3 Model (2) summary

Baseline architecture with extra activation function layer (Figure 3.12).

3.1.4 Model (3) summary

Baseline architecture and using early stopping with data augmentation (Figure 3.13).

3.1.5 Model (4) summary

Different Architecture, more deep network with 6 batch normalisation layers and 4 dropout layers with (0.25) (Figure 3.14).

3.1.6 Model (5) summary

Using baseline architecture from the model (3), but using a different type of optimizer “RMSprop”

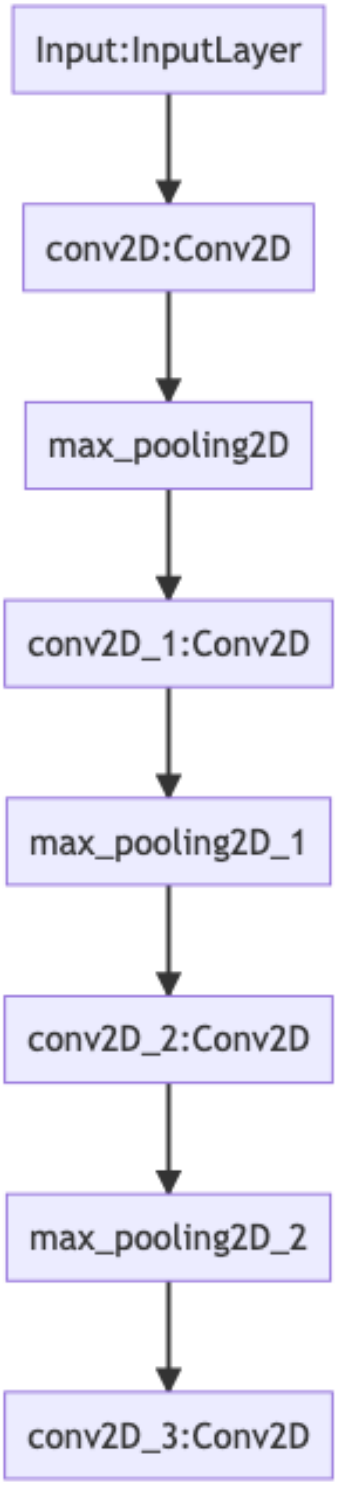


Figure 3.9: Baseline Structure

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 30, 30, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 13, 13, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 4, 4, 128)         73856
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 64)                131136
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650
=================================================================
Total params: 225,034
Trainable params: 225,034
Non-trainable params: 0
_________________________________________________________________
```

Figure 3.10: Baseline model summary

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 112)       8512
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 26, 26, 48)        48432
_________________________________________________________________
flatten (Flatten)            (None, 32448)             0
_________________________________________________________________
dense (Dense)                (None, 128)               4153472
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
=================================================================
Total params: 4,211,706
Trainable params: 4,211,706
Non-trainable params: 0
_________________________________________________________________
```

Figure 3.11: Best model summary using Random search

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 30, 30, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 13, 13, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 4, 4, 128)         73856
_________________________________________________________________
flatten_1 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                131136
_________________________________________________________________
dense_3 (Dense)              (None, 10)                650
=================================================================
Total params: 225,034
Trainable params: 225,034
Non-trainable params: 0
_________________________________________________________________
```

Figure 3.12: Model (2) summary

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 30, 30, 32)        896
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 13, 13, 64)        18496
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 6, 6, 128)         8320
_________________________________________________________________
flatten_2 (Flatten)          (None, 4608)              0
_________________________________________________________________
dense_4 (Dense)              (None, 64)                294976
_________________________________________________________________
dense_5 (Dense)              (None, 10)                650
=================================================================
Total params: 323,338
Trainable params: 323,338
Non-trainable params: 0
_________________________________________________________________
```

Figure 3.13: Model (3) summary

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_8 (Conv2D)            (None, 32, 32, 32)        896
_________________________________________________________________
batch_normalization (BatchNo (None, 32, 32, 32)        128
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 32, 32, 32)        9248
_________________________________________________________________
batch_normalization_1 (Batch (None, 32, 32, 32)        128
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 16, 16, 64)        18496
_________________________________________________________________
batch_normalization_2 (Batch (None, 16, 16, 64)        256
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 16, 16, 64)        36928
_________________________________________________________________
batch_normalization_3 (Batch (None, 16, 16, 64)        256
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 8, 8, 64)          0
...
Total params: 552,362
Trainable params: 551,466
Non-trainable params: 896
_________________________________________________________________
```

Figure 3.14: Model (4) summary

3.1.7 Model (6) summary

Using deep network from model (4) architecture with Learning rate = 3E-4

4 Results

- The Confusion matrix for baseline model (Figure 4.1).

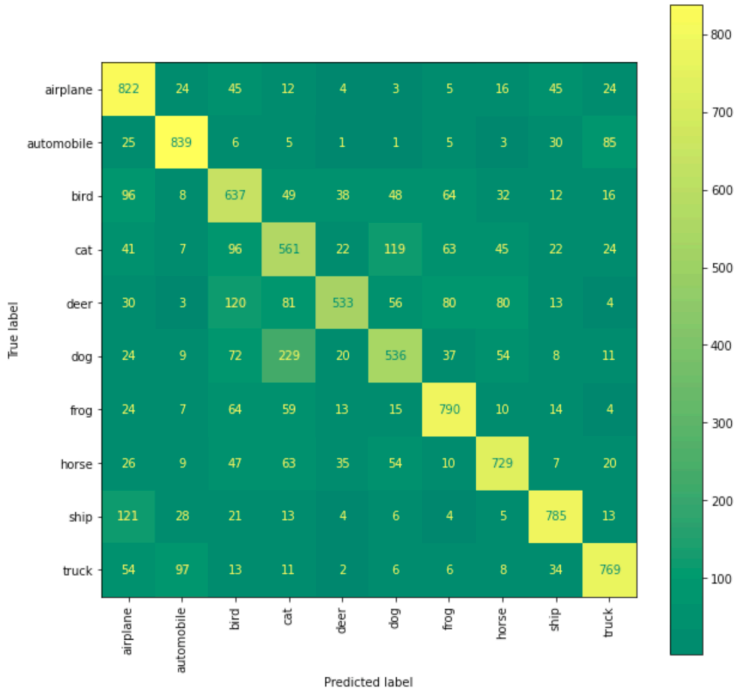


Figure 4.1: Baseline model confusion matrix

- The confusion matrix for baseline and data augmentation model (Figure 4.2).

- The classification report of baseline model (Figure 4.3).

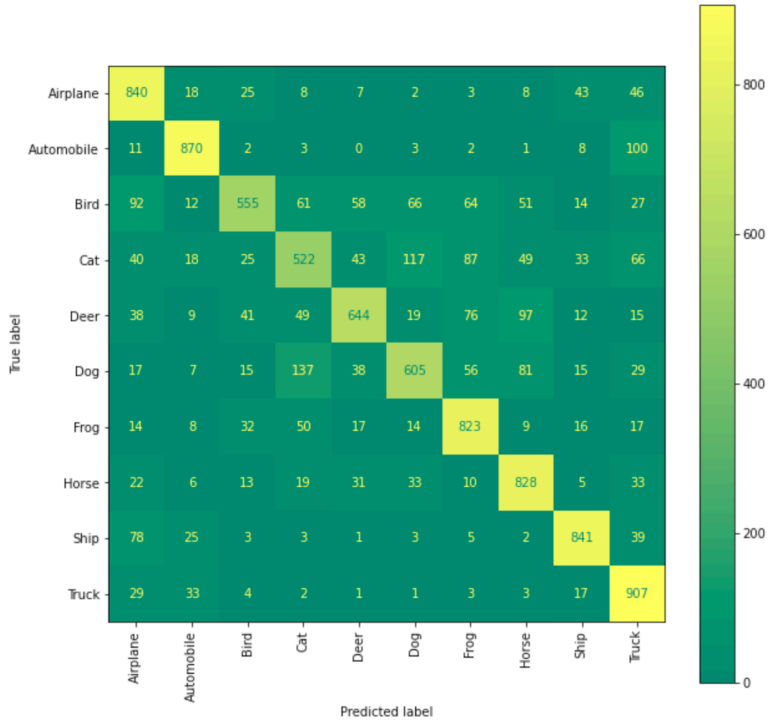


Figure 4.2: Baseline_and_Data_Augmentation confusion matirx

	precision	recall	f1-score	support
0	0.65	0.82	0.73	1000
1	0.81	0.84	0.83	1000
2	0.57	0.64	0.60	1000
3	0.52	0.56	0.54	1000
4	0.79	0.53	0.64	1000
5	0.64	0.54	0.58	1000
6	0.74	0.79	0.77	1000
7	0.74	0.73	0.74	1000
8	0.81	0.79	0.80	1000
9	0.79	0.77	0.78	1000
accuracy			0.70	10000
macro avg	0.71	0.70	0.70	10000
weighted avg	0.71	0.70	0.70	10000

Figure 4.3: The Classification Report of baseline model

- The classification report of baseline and data augmentation model(Figure 4.4).

	precision	recall	f1-score	support
0	0.71	0.84	0.77	1000
1	0.86	0.87	0.87	1000
2	0.78	0.56	0.65	1000
3	0.61	0.52	0.56	1000
4	0.77	0.64	0.70	1000
5	0.70	0.60	0.65	1000
6	0.73	0.82	0.77	1000
7	0.73	0.83	0.78	1000
8	0.84	0.84	0.84	1000
9	0.71	0.91	0.80	1000
accuracy			0.74	10000
macro avg	0.74	0.74	0.74	10000
weighted avg	0.74	0.74	0.74	10000

Figure 4.4: Baseline and Data Augmentation classification report

- Evaluating baseline model prediction (Figure 4.5).

- Evaluation one image using baseline model (Figure 4.6).

- Evaluating best model prediction (Figure 4.7).

4.1 Comparison between models

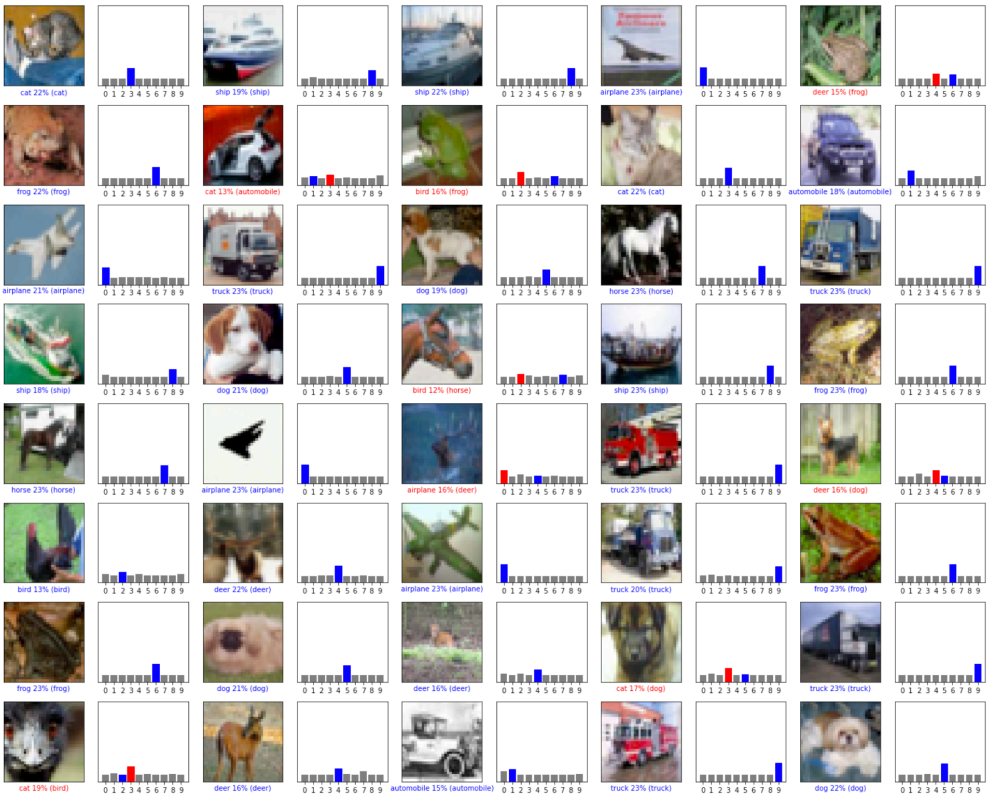
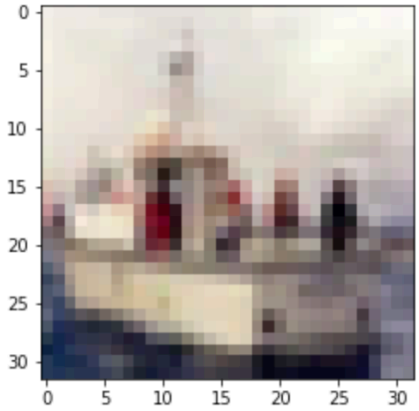


Figure 4.5: Baseline model prediction

```
1  my_image = test_dataset[88]
2  plt.imshow(my_image)
```

✓ 0.3s

```
<matplotlib.image.AxesImage at 0x132319340>
```




```
1  np.argmax(model.predict(my_image.reshape(1, 32, 32, 3)))
```

✓ 0.1s

```
8
```

```
1  class_names[8]
```

✓ 0.1s

```
'Ship'
```

Figure 4.6: Image prediction

Figure 4.7: Best model prediction

Table 4.1: Models Training Times

Model	Training time, s
Baseline model	1270.78
Best model	747.842
Baseline model with Data Augmentation	1768.82
Model 2	1318.21
Model3	1243.54
Model4	6201.63
Model5	1520.63
Model6	1405.26

Table 4.2: Models Training Accuracy and Loss

Model	Training Accuracy	Training Loss
Baseline model	0.7807	0.6199
Best model	0.9842	0.0685
Baseline model with Data Augmentation	0.7508	0.7156
Model 2	0.8347	0.4648
Model3	0.8367	0.4618
Model3 with early stopping	0.6447	1.0266
Model4	0.82	0.49
Model5	0.9349	0.1861
Model6	0.9604	0.1235

Table 4.3: Models Testing Accuracy and Loss

Model	Testing Accuracy	Testing Loss
Baseline model	0.7162	0.9203

Model	Testing Accuracy	Testing Loss
Best model	0.7791	0.7093
Baseline model with Data Augmentation	0.7435	0.7663
Model 2	0.7145	0.9526
Model 3	0.7107	1.7176
Model 3 with early stopping	0.7293	0.8046
Model 4	0.81	0.54
Model 5	0.7107	1.7176

- Plotting models accuracy (Figure 4.8).

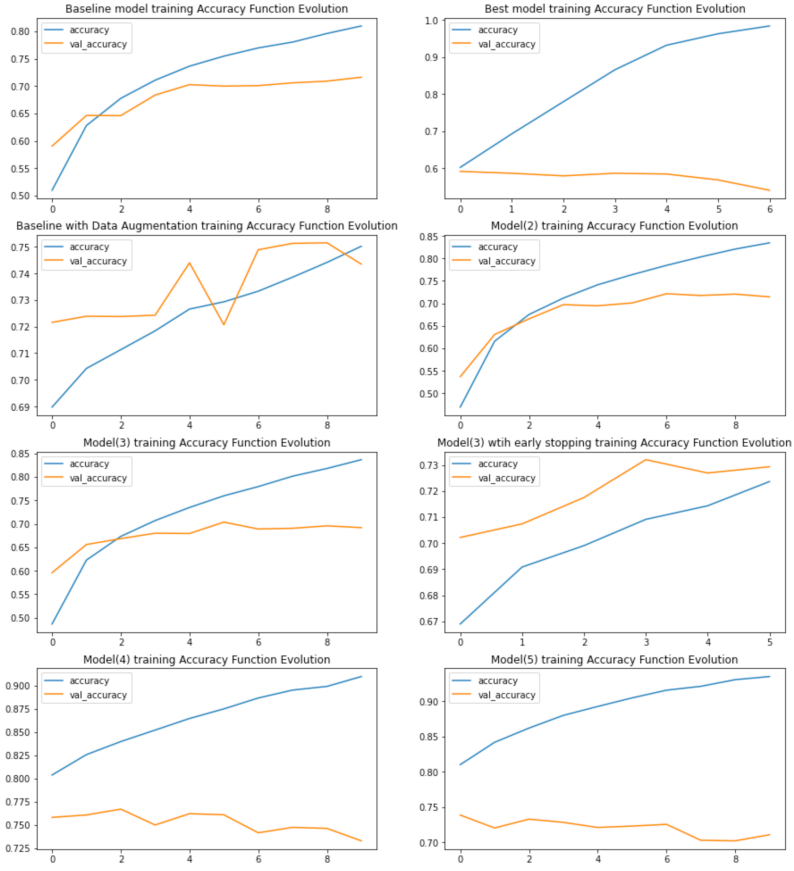


Figure 4.8: Evaluation of Models Accuracy

- Plotting models loss (Figure 4.9).

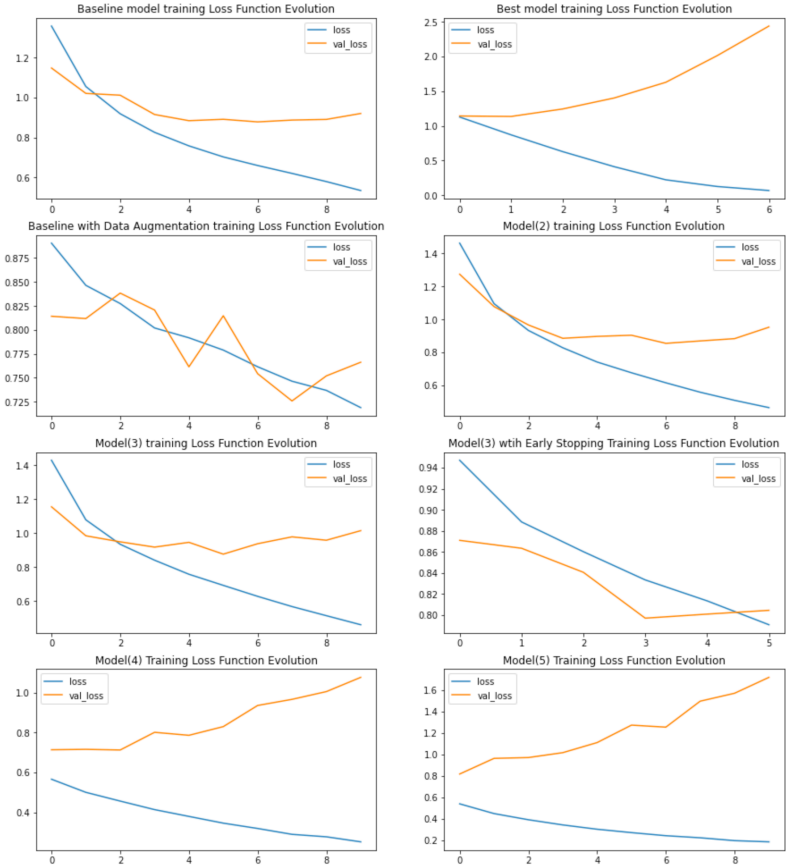


Figure 4.9: Evaluation of Models Loss

- Comparison between Model(3) and Model(5) (Figure 4.10).

- Comparison between Model(4) and Model(6) (Figure 4.11).

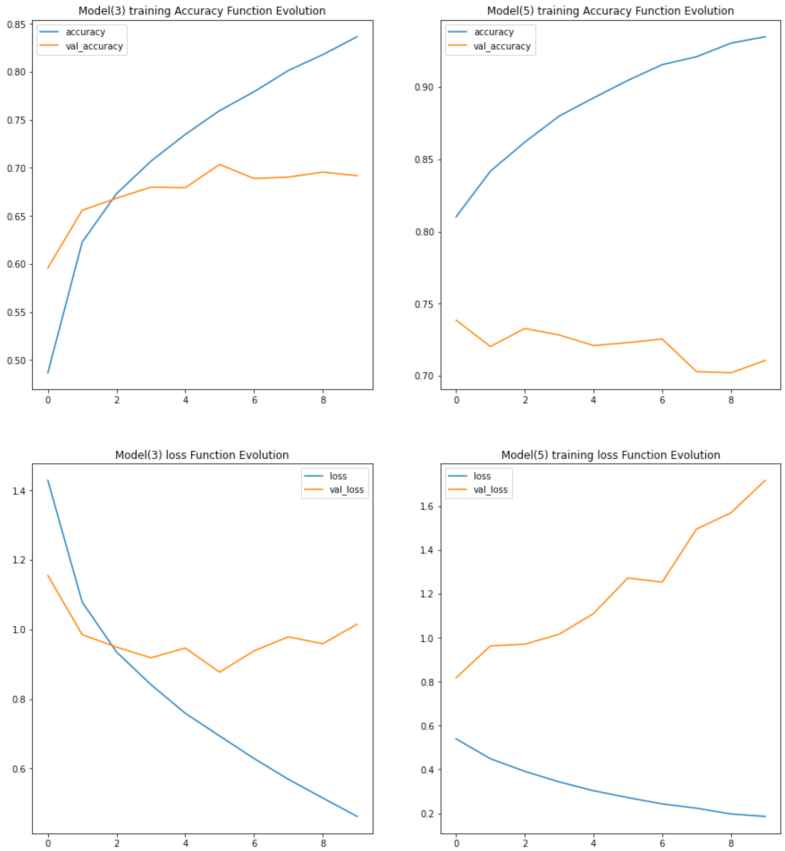


Figure 4.10: Comparison between Model(3) and Model(5)

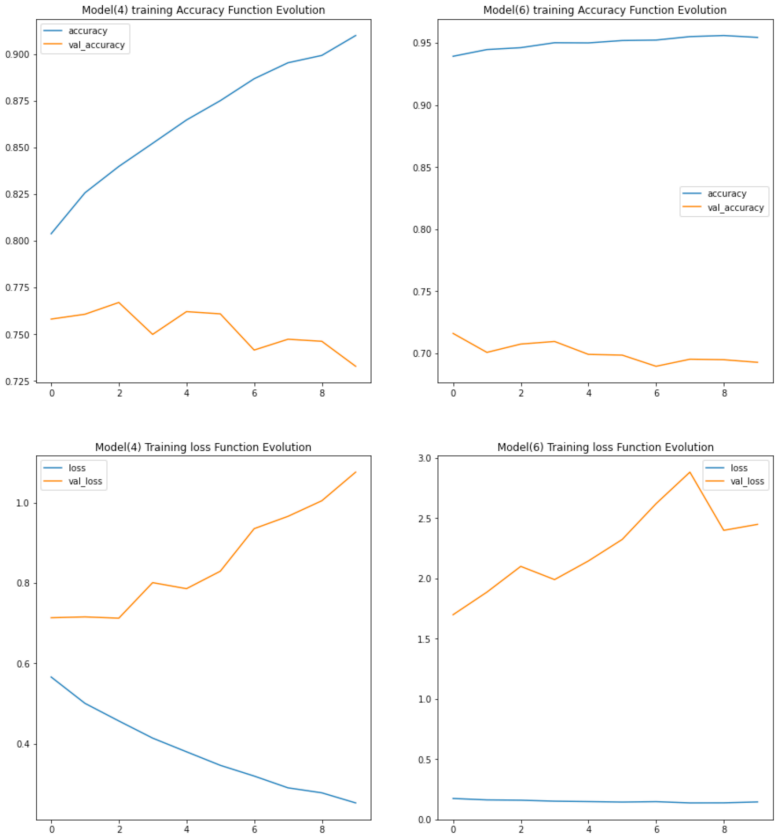


Figure 4.11: Comparison between Model(4) and Model(6)

5 Discussion & Conclusion

In multi-class classification, it is essential to assess the speed rate of our model because the model needs to classify each image for one class. Furthermore, the model must be meticulous because we have many classes and a large dataset, and images will be checked separately for the classes (Brownlee, 2020a). I used the softmax function with all model's last layers because we deal with a multi-class classification. The softmax function gives a probability distribution that includes various possible values of various classes. However, the image must belong exclusively to one class, so in our dataset, the image that belongs to an air plane can not be classified as an automobile at the same time. This softmax function is a type of sigmoid function where it gives the most probable class the most significant portion of the distribution, and the other classes take the rest. The function turns all values into probabilities. These probabilities are all measured with a relative scale. Softmax gives the probabilities of each class all summed to one (Brownlee, 2020b). Sparse Categorical cross-entropy was utilised as a loss function that compares the forecast results with the target to estimate the loss. The loss function is picked relying on the design of the labels. When it is a single-label or a multi-class classification problem, the label can only belong to one class, so it can be expressed using the one-hot embedding. The sparse categorical saves memory when we have a large number of classes, and therefore it speeds up the model at almost two times than other loss functions (Wei, 2018). Relu function algorithm enforces the data to be reasonable, so if the input is <=0, the output will be 0. Otherwise, the output value = input value. If the input is positive, the output will be the same as the input. Otherwise, the outcome will be zero. This technique creates a sparse characteristic(Brownlee, 2019). Moreover, it enhances the model computing rate to become faster without gradient problems, unlike sigmoid and tanh functions. As a result, Relu is the best activation function for forwarding propagation and is used as a standard function for CNN. This function makes the model easier to train, performs better, and gives results in less

time(Wang *et al.*, 2020). Meanwhile, in Alex Krizhevsky et al. paper, the author also used Relu for ImageNet Classification with CNN, and the network acquired high results. For dimensionality reduction, I utilised the principal component analysis, which is an approach that lets us transform a vector from m dimension to other dimensions while model processing the data, allowing us to control the data dimensions and make them smaller. In the kernels, odd numbers are used typically in CNN. In the models, I used (2, 2) with convolutional layers and (3, 3) with pooling layers, depending on the image's dimensions in our dataset. These kernels will be suitable with 32×32 dimensions. These kernels are a form of dimensionality reduction, which helps the model reduce the noise (Rosebrock, 2018). The filters usually range [32, 64, 128] in the first layers and increase in deep layers [256, 512, 1024]. These filters depend on the complexity of our dataset and the depth of our model, so I started with a small number of filters and increased it as we went in-depth (Rosebrock, 2018). Adam optimiser has been the best optimiser based on previous studies that show adam's high performance in classification tasks. I tried using "RMsProp in one of the models, and the training performance was not as expected. Adam is used widely as a default optimiser, known for its simplicity, rapid running time, and minimal space requirements and needs less tunning than other optimisers (Gupta, 2021).

- Baseline model: LeNet is the most famous architecture on CNN, which was used for the baseline model (Figure 3.10) (OpenGenus, 2021). In the confusion matrix, the model was confusing between dog and cat 229 times, and for ship and airplane 121 times (Figure 4.1). The precision was the highest metric in the classification report at 71%, and the recall and F1 -scores were 70%. For model prediction evaluation, the network performed very well, and in most cases, it was sure about the results and gave one class for each image (Figure 4.5). The model has correctly predicted an image from the dataset, the image was a ship, and the prediction class was 8, which is a ship label (Figure 4.6).
- Best Model: In a Random search, the model was the most straightforward, containing two convolutional layers, Flatten layer, and a fully connected layer, and

the accuracy has reached 99% with a low loss of 0.36% (Figure 3.12). The network performance was very accurate for model prediction evaluation, and nearly no errors were detected in the predictions (Figure 4.7).

- For the Data augmentation model, using the baseline architecture, I considered some rotation to the images by shifting the width and height and flipping it horizontally to increase the model accuracy. In confusion, it was mainly confused between dog and cat 137 times (Figure 4.2). The metrics were all the same result of 74% in the classification report.
- Model (2) was built similar to the baseline model but with an extra layer with the "Relu" function
- Model (3) uses the baseline model with an early stopping technique to stop the model when it stops improving and data augmentation from the previous model (Figure 3.13).
- Model (4) was built using a deeper architecture with 6 batch normalisation layers and four dropout layers (Figure 3.14).
- Model (5) using baseline model and different testing optimiser "RMSprop."
- Model (6) uses a deep network in the model (4) with a learning rate = 3E-4.

In summary: - Baseline with data augmentation showed better results in the confusion matrix - Model 3 finished within 20 minutes and 72 seconds which is the best. Finally, the slowest model was model 5 (Table 4.1). - Model 5 was the worst in training time because we used a different optimiser, "RMSprop." - For model training accuracy, the highest accuracy rates were model 6 with 96%, model 5 with 93%, and model 4 with 82%. In contrast, the worst model was model(3), with early stopping at 64 %, and a baseline model with data augmentation of 75% (Table 4.2). - For model training loss, model 6 achieved the lowest loss of 12%, and model (5) was 18%. The highest model loss was in the model (3) with early stopping (Table 4.2) - Baseline model with data augmentation achieved the highest testing accuracy of 74% (Table 4.3). - For model testing loss, model 3 with data augmentation was the lowest with 80% and the highest loss was detected in models (2) and (5) with RMSprop optimiser (Table 4.3). - Model

(4) is the best overall from the metrics side: training accuracy of 91%, and training loss at 49% (Figure 4.11). Nevertheless, it was the slowest model and finished the training in 103 minutes. - Model 6 using learning rate has performed very well, but in interpreting the model performance, it shows that the model was not learning quickly (Figure 4.11). - Model 4 will be used for further improvement, and using other techniques such as data augmentation may increase the model accuracy, and decreasing the number of hidden layers may enhance the training time

5.1 conclusion

CNN is a complex network that can be improved infinitely depending on what we are looking to improve, such as classification methods, learning rate, or the layers design. There are multiple valuable methods to enhance the accuracy, but the researchers always choose the most straightforward and most apparent structure to achieve the top results. CNN model is widely used for image classification tasks. The activation function is the most crucial part of the model's effectiveness. This paper studies the influence of different CNN architecture and techniques on image classification based on the ReLu function. In comparing different models, I can demonstrate the advantages of deep neural networks in(Table 4.3) . The table shows that the image classification model proposed in this paper performs better with dropout and batch normalisation with deep networks. The performance was at the top but needed a longer time to train the network. For better accuracy, other techniques may help, such as using data augmentation and different learning rates and using fewer hidden layers may decrease the training time.A summary of this report is shown in Figure 5.1. The machine learning and deep learning fields aim to develop new techniques to solve society issues. I believe these neural networks can make considerable improvements in life quality. CNN solutions are already used clinically and improved health services quality, and these fields remain filled with tremendous opportunities. Despite the privacy issues that require further research, CNN is a good resource for educators to explore new findings in health fields. For example, some studies show that by 2053 Artificial Intelligence will achieve better results than

humans in surgeries (Fogel and Kvedar, 2018). In addition, CNN Algorithms can improve the computational power and increase the capacity of more hidden layers to process complex data. Finally, data privacy issues can be solved using blockchain technology to make it easier for patients to manage their data and grant approval for detailed uses (Brady and Neri, 2020).

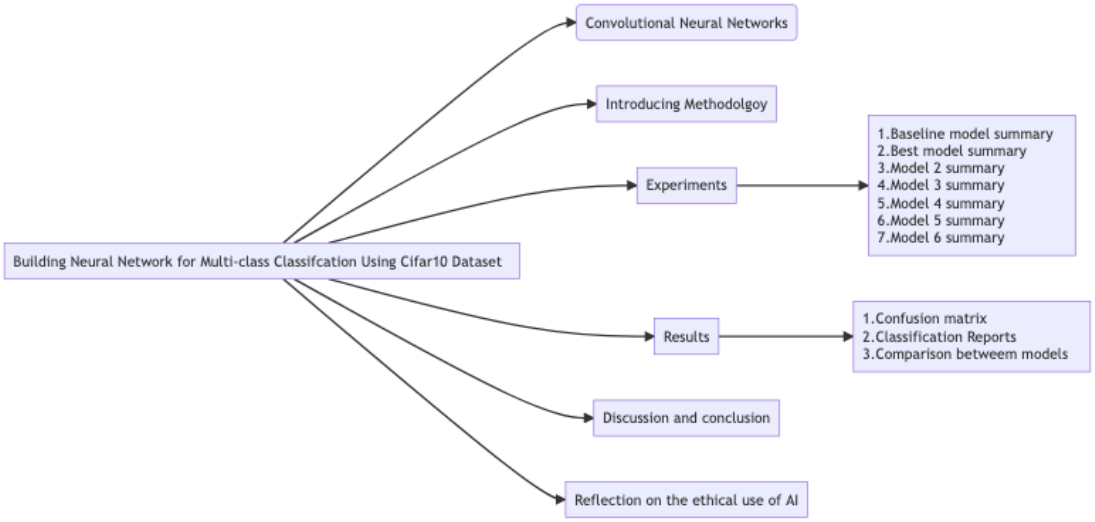


Figure 5.1: Summary

6 Reflection on the Ethical Use of AI

The main challenge of building new technologies is data access. In some cases, developers need a permit to analyze and use the data. Otherwise, developers will not be able to train and develop the system because there is no actual data to be trained on. Therefore, developers must understand the implication of developing new technologies before starting, such as privacy issues, what implications that technologies can lead to and who can get access to the data and what data they can access, what the effect of these technologies on our society and who can get access to this technologies?. Introducing new technologies to the healthcare field is a responsibility for improving the health of our community and society, not the opposite. Therefore, some institutions' main goal is to ensure that any new technologies developed to serve our society must be tested and checked before. For example, the Food and Drug Administration (FDA) is an institution that holds regulations to protect our society. FDA found that the prominent people's concerns regarding emerging AI technologies in the healthcare system are the credibility of these systems, the privacy of their data, and communication with machines instead of humans. Moreover, some researchers found that people with higher education are more likely to support adopting these new systems (FDA, 2021). On the other hand, using CNN will lead to new uncertainties, for example: 1. The interpretability of CNN and how the network makes decisions. Researchers are looking for new techniques to solve this issue, affecting system reliability (Chartrand *et al.*, 2017). 2. The shortage and Inaccessibility to labelled datasets due to privacy concerns and lacking experts to interpret images for accurate labels (Chartrand *et al.*, 2017). 3. Model overfitting due to the limitation of the dataset, which will lead to lower accuracy when the model interprets new data (Chartrand *et al.*, 2017).

In the developing phase of CNN, multiple issues were presented during the deployment process, such as: 1. Comprehend the black boxes inside the network; experimenters are still trying to understand how this network creates decisions, which has led to ethical

issues and algorithmic discrimination (Zhang *et al.*, 2021). 2. Algorithmic bias was detected in the network because of bias in the training dataset and difficulty identifying all the biases to ensure that the system treats all people without bias (Zhang *et al.*, 2021). 3. There have been societal concerns about human-machine interaction (FDA, 2021). For adaptive algorithms such as CNN, which evolve and continually improve, the medicine and healthcare products regulatory agency (MHRA) guidance for software as a medical device(SaMD) for these algorithms is categorized based on the risk levels. Ensuring safety rules were followed through all the development stages and improvement circles. However, these algorithms' regulations are still unclear because they are constantly changing, and it is hard to determine the impact continuously (NHSX, 2019).

To avoid algorithms issues in the model, developers can use these questions as a tool to keep our algorithms clear: 1. Do we have any bias, to which extent, if any, from the training dataset? 2. Is the algorithm fairly processed data? 3. Were any user guidance provided with the system? 4. Output is tested and validated? 5. If a regulation needs to be updated, is it updated? 6. Network decisions are transparent or need a different explanation? 7. If users need skills to use the system, is it supported? 8. The system is doing good for our society? A study has revealed that due to algorithmic bias, millions of African-American patients have been affected due to racial bias in the system used by US hospitals. These people were less likely to get personalized care (Ledford, 2019). CNN has many ethical issues, especially when working with very sensitive data such as medical records, which is very critical to our society. Society's main worries were about how the decisions were made and other related data privacy issues. For Automated decisions, society has the right to understand how the system processes there data and creates decisions. Thus, CNN still lacks access to enormous datasets due to privacy issues. In addition, the dataset size plays a massive role in the model's accuracy. Therefore, the more data accessibility, the higher the model accuracy. In some cases, patients with rare diseases refuse to give access to their crucial information, so we will not be able to train our model for this type of disease till we get access to data (Yamashita *et al.*, 2018). Regulation will always need to be revised and expanded because we still have

lag. Numerous regulation bodies attempt to achieve adequate regulations where safety rules are satisfied with technological improvements. However, CNNs are still tricky to be controlled by the regulation due to the lack of transparency (FDA, 2021).

References

Brady, A.P. and Neri, E. (2020) 'Artificial Intelligence in Radiology—Ethical Considerations', *Diagnostics*, 10(4, 4), p. 231. doi:10.3390/diagnostics10040231.

Brownlee, J. (2019) *A Gentle Introduction to the Rectified Linear Unit (ReLU)*. Machine Learning Mastery. Available at: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/ (Accessed: 27 April 2022).

Brownlee, J. (2020a) *4 Types of Classification Tasks in Machine Learning*. Machine Learning Mastery. Available at: https://machinelearningmastery.com/types-of-classification-in-machine-learning/ (Accessed: 27 April 2022).

Brownlee, J. (2020b) *Softmax Activation Function with Python*. Machine Learning Mastery. Available at: https://machinelearningmastery.com/softmax-activation-function-with-python/ (Accessed: 27 April 2022).

Chartrand, G. *et al.* (2017) 'Deep Learning: A Primer for Radiologists', *RadioGraphics*, 37(7), pp. 2113–2131. doi:10.1148/rg.2017170077.

Cheng, A.-T. *et al.* (2018) 'Software and Hardware Enhancement of Convolutional Neural Networks on GPGPUs', *Advances in Science, Technology and Engineering Systems Journal*, 3, pp. 28–39. doi:10.25046/aj030204.

FDA (2021) 'Artificial Intelligence (AI) and Machine Learning (ML) in Medical Devices', *ADMINIS TRATION*, p. 25.

Fogel, A.L. and Kvedar, J.C. (2018) 'Artificial intelligence powers digital medicine', *npj*

Digital Med, 1(1, 1), pp. 1–4. doi:10.1038/s41746-017-0012-2.

Gupta, A. (2021) *A Comprehensive Guide on Deep Learning Optimizers*. Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/ (Accessed: 8 May 2022).

Hosny, A. *et al.* (2018) 'Artificial intelligence in radiology', *Nat Rev Cancer*, 18(8, 8), pp. 500–510. doi:10.1038/s41568-018-0016-5.

Krizhevsky, A. (2009) *Learning multiple layers of features from tiny images*.

Ledford, H. (2019) 'Millions of black people affected by racial bias in health-care algorithms', *Nature*, 574(7780, 7780), pp. 608–609. doi:10.1038/d41586-019-03228-6.

Liu, T. *et al.* (2015) 'Implementation of Training Convolutional Neural Networks'. Available at: http://arxiv.org/abs/1506.01195 (Accessed: 12 November 2021).

NHSX (2019) *Artificial Intelligence: How to get it right*. Available at: https://www.nhsx.nhs.uk/media/documents/NHSX_AI_report.pdf (Accessed: 30 November 2021).

OpenGenus (2021) *Different types of CNN models*. OpenGenus IQ: Computing Expertise & Legacy. Available at: https://iq.opengenus.org/different-types-of-cnn-models/ (Accessed: 9 May 2022).

Ponraj, A. (2021) *A Tip A Day — Python Tip #8: Why should we Normalize image pixel values or divide by 255?* Analytics Vidhya. Available at: https://medium.com/analytics-vidhya/a-tip-a-day-python-tip-8-why-should-we-normalize-image-pixel-values-or-divide-by-255-4608ac5cd26a (Accessed: 28 April 2022).

Rosebrock, A. (2018) *Keras Conv2D and Convolutional Layers*. pyimagesearch. Available at: https://www.pyimagesearch.com/2018/12/31/keras-conv2d-and-convolutional-layers/ (Accessed: 27 April 2022).

Wang, W. *et al.* (2019) 'Development of convolutional neural network and its application in image classification: A survey', *OE*, 58(4), p. 040901. doi:10.1117/1.OE.58.4.040901.

Wang, Y. *et al.* (2020) 'The Influence of the Activation Function in a Convolution Neural Network Model of Facial Expression Recognition', *Applied Sciences*, 10(5, 5), p. 1897. doi:10.3390/app10051897.

Wei, L. (2018) *Multi-hot Sparse Categorical Cross-entropy - MXNet - Apache Software*

Foundation. Confluence. Available at: https://cwiki.apache.org/confluence/display/MXNET/Multi-hot+Sparse+Categorical+Cross-entropy (Accessed: 27 April 2022).

Yamashita, R. *et al.* (2018) 'Convolutional neural networks: An overview and application in radiology', *Insights Imaging*, 9(4), pp. 611–629. doi:10.1007/s13244-018-0639-9.

Zhang, Y. *et al.* (2021) 'A Survey on Neural Network Interpretability', *IEEE Transactions on Emerging Topics in Computational Intelligence*, 5(5), pp. 726–742. doi:10.1109/TETCI.2021.3100641.