–Classification of CIFAR-10 Dataset–

Ozan Can ALPER

June 2022

Contents

1 Introduction

Visual object recognition is essential for humans to interact with each other and with the natural worldHo-Phuoc [2018]. The fact that we can recognize the objects we encounter in our lives almost effortlessly shows that we have a great visual recognition ability. In particular, we can easily recognize the object even if its position, direction, scale or illumination changes DiCarlo et al. [2012].

In the field of computer vision, attempts have been made to create systems that can mimic humans' object recognition ability. Despite decades of effort, machine visual recognition was far from human performance. However, thanks to the advent of convolutional neural networks (CNN) and deep learning [LeCun et al. [2015],Schmidhuber [2015],Bengio et al. [2013],Krizhevsky et al. [2012],Bengio [2009]], machine performance has been significantly improved, thus surpassing even human performance [He et al. [2015], He et al. [2016]].

In this study, the model trained on the recognition of images from different classes and the methods used are explained with an approach using Convolutional Neural Networks (CNN). It is observed how the use of different hyperparameters affects the success of the model. In addition, the application of the transfer learning method is shown and its difference with the classical CNN method is interpreted.

2 Data set

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class. The names of the classes in the data set are as follows;

- airplane
- automobile
- bird
- cat
- deer
- dog
- frog
- horse
- ship
- truck

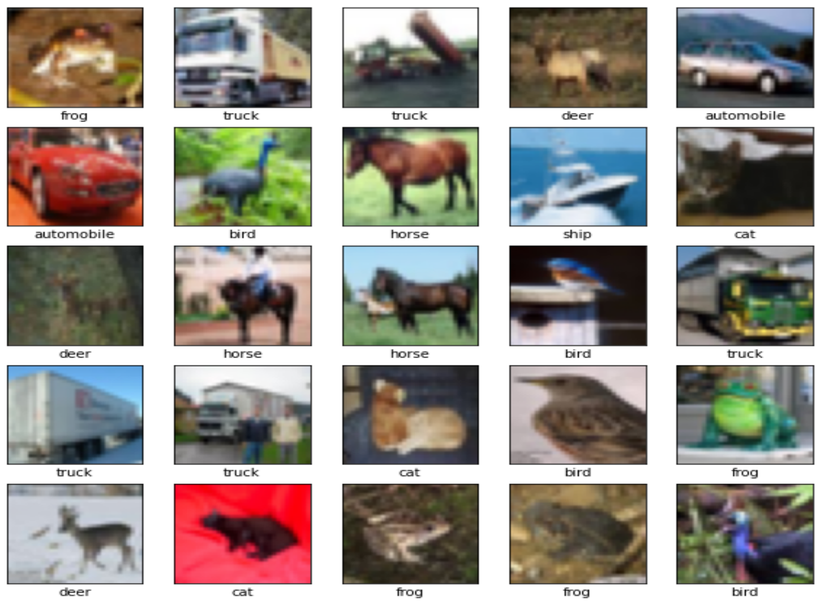


Figure 1: CIFAR-10 Dataset

3 Data Preprocessing

Data preprocessing is the process of preparing raw data and fitting it into a machine learning model. It is the first and most important step when building a machine learning model. When creating a machine learning project, it is not always possible to come across clean and formatted data. And when doing any operation with data, it is imperative to clean it and put it in a formatted way. For this, data must be preprocessed. When the image set we used is examined, it is seen that the pixel values do not have the same scale. This is a factor that can negatively affect the network we will train. For this reason, the Normalization process was applied to the values of the pixel.

3.1 Normalization

Normalization is a rescaling of the data from the original range so that all values are within the new range of 0 and 1.

$$x = \frac{value}{value_{max}} \tag{1}$$

Since the maximum pixel value is 255 here, the values are divided by 255 to normalize between 0-1.

4 Methods

This section describes the methods used during the study.

4.1 Machine Learning and Deep Learning

Interactions in constantly used social media accounts, search engines and traces left behind when searching, movements made with bank accounts, blogs, mails, sensors in technological devices, biomedical data, images, music are seen as elements that support the concept of big data. Moreover, it seems that these data sources will continue to increase. This deluge of data calls for automated methods of data analysis, which is what machine learning provides. In particular, machine learning is defined as a method that can automatically detect patterns in data using mathematical and statistical methods, makes predictions about the future patterns with these inferences, or to perform other kinds of decision making under uncertainty. Hence, this method becomes more significant day by dayMurphy [2012].

Deep Learning is a subfield of machine learning that deals with algorithms inspired from the structure and function of the brain called artificial neural networks. This method allows computers to learn from experiences and define each concept by considering its hierarchical relationship with other concepts. In this approach, which collects information from experience, people do not need to specify all the information computers need for machine learning. The hierarchy of concepts enables the computer to learn complex concepts by building them from simpler ones. Thus, a deep structure is formed with many layers. To understand deep learning well, it is necessary to have a solid understanding of the fundamentals of machine learningGoodfellow et al. [2016].

4.2 Artificial Neural Network

Artificial neural networks are computational networks that try to simulate the neuronal networks in the biological nervous system in a large way. It is the foundation of artificial intelligence (AI) and solves problems that prove impossible or difficult by human or statistical standards. ANNs have self-learning capabilities that enable them to produce better results as more data is available. ANNs use backpropagation algorithms, to perfect their output results Graupe [2013].

4.3 Classification

The concept of classification is to distribute data among various classes defined on a data set. Classification algorithms learn this way of classification by applying certain algorithms to the training data, and then they try to correctly classify the test data for which the class label is not certain.

4.4 Deep Learning-Convolutional Neural Network (CNN)

Convolutional neural network has a strong ability to extract features of image. Currently widely used convolutional neural network structure includes input layer, convolution layer, activation layer, pooling layer, full connection layer and output layerWang et al. [2020]. The input layer is the first layer of the convolutional network. In this layer, single-channel images or multi-channel images are the inputs. The convolution layer consists of a series of filters. These filters are applied to images to create feature maps. The activation function is used to activate each element of the convolution layer, It is a nonlinear mapping process that does not change the size of the input-output matrix of the network. If the activation function is not applied, the output signal becomes a simple linear function. Linear functions are only first-order polynomials. Hence, learning power of a neural network without activation function will be limited. The function of the pooling layer is to reduce the spatial size of the representation to reduce the number of parameters and calculations in the network. With this function, it also increases the calculation speed of the model. Maximum pooling can reduce the error of convolution process and keep the structure information of image effectively, so it is widely used. Fully connection layer is a feature vector consisting of deep layer properties after feature extraction with multi convolution and pooling layers. The output layer is used to match the feature vector in the full connection layer with the category probability of the input image and determine the classification resultKetkar and Santana [2017].

4.4.1 Hyperparameters

Convolutional neural networks (CNNs) are the most intensively researched Sultana et al. [2018] models for image recognition because they are more accurate than human judgement Bergstra et al. [2011]. With the combination of three layers (that is, the convolution, pooling, and fully connected layers) and one function, CNNs have considerable flexibility to allow users to make modifications according to their needs. The history of CNNs can be traced back to 1962 Hubel and Wiesel [1962]; however, the model that is closest to the present definition of a CNN is the LeNet proposed by Yann LeCun in 1989, and it has been revised repeatedly since then LeCun et al. [1989].

After the proposal of AlexNet Krizhevsky et al. [2012] in 2012, there have been significant advancements in CNNs. Many CNN models, such as VGG, ResNet, and GoogLeNet, have been developed successively. Increasing the number of layers leads to less accurate results He et al. [2016]. Hence, numerous studies have investigated methods to improve CNN performance without changing the architecture.

Hyperparameters include the size of kernels, number of kernels, length of strides, and pooling size, which directly affect the performance and training speed of CNNsYeh et al. [2021]. Moreover, the impact of hyperparameters increases as the complexity of the network increases. Hence, the most popular method of improving CNN performance is to optimise the hyperparameters [Hazan et al. [2017], Zhang et al. [2019]].

4.5 Transfer Learning

Transfer Learning (TL) is method of machine learning (ML) that focuses on storing the knowledge gained while solving a problem and applying it to a different but related problemtra [2021]. This method was also inspired from the people. It can be explained as the ability to evaluate future situations with the experience or knowledge gained in the past. There are models (VGG-16, ResNet50, Inception, AlexNet etc.) prepared by training with images from 1000 different classes beforehand for transfer learning. Among these models, we use VGG-16 in this application because of its layer properties and success rate.

4.5.1 VGG-16 architecture

VGG-16 is a 16-layer network proposed by Visual Geometric Group of Oxford University. The VGG network architecture was introduced by Simonyan and Zisserman in their 2014 article that is named Very Deep Convolutional Networks for Large Scale Image RecognitionSimonyan and Zisserman [2014]. The structure of the VGG-16 model we use for Transfer Learning is shown in the Figure 1.

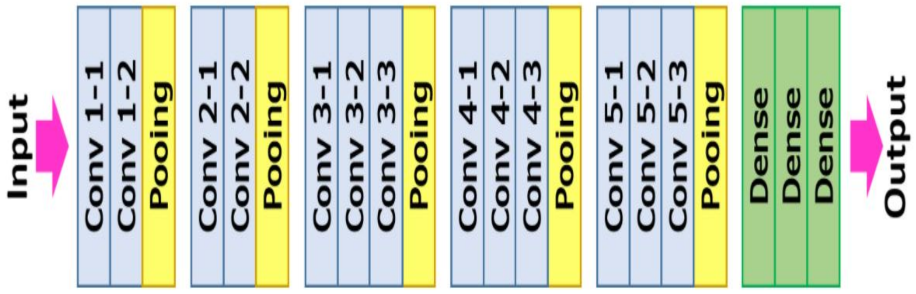


Figure 2: VGG-16 Architectureul Hassan

5 Evaluation Metrics

Some evaluation metrics are used to evaluate the performance of trained classifiers. In this section, these evaluation metrics are introducedHossin and Sulaiman [2015]. The meaning of the expressions used in the explanations are as follows;

- TP = True Positive
- TN = True Negative
- FP = False Positive
- FN = False Negative

5.1 Accuracy

The accuracy metric measures the ratio of correct predictions number over the total number of samples. Accuracy formula can be represented as Equation (2).

$$\frac{TP+TN}{TP+FP+TN+FN} \tag{2}$$

5.2 Precision

The precision metric measure the ratio of the positive patterns that are correctly predicted over the total predicted patterns in a positive class. Precision formula can be represented as Equation (3).

$$\frac{TP}{TP+FP} \tag{3}$$

5.3 Recall

The recall (also known as sensitivity) metric is used to measure the fraction of positive patterns that are correctly classified. Recall formula can be represented as Equation (4).

$$\frac{TP}{TP+TN} \tag{4}$$

5.4 F1-Score

F1-Score metric represents the harmonic mean between recall and precision values. For imbalanced dataset, this metric provides a more robust decision about the model. F1-Score formula can be represented as Equation (5).

$$2 * \frac{Precision * Recall}{Precision + Recall} \tag{5}$$

5.5 Confusion Matrix

The confusion matrix summarizes the classifier's classification performance based on some test data. A confusion matrix has two-dimensions, one dimension is indexed by the actual class, the other is indexed by the class that the classifier predictsSammut and Webb [2011].

		Prediction	
		0	1
Actual	0	TN	FP
	1	FN	TP

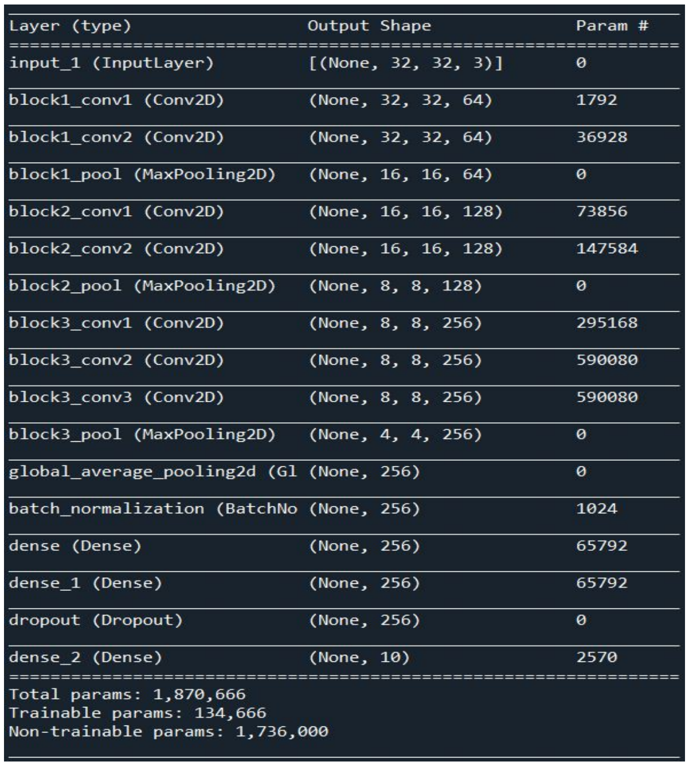
Table 1: Confusion Matrixcon [2021]

6 Applications & Results

This is the section that contains the application and the results obtained (table, picture, graphic or numerical).

6.1 Transfer Learning Model (VGG-16)

At this stage, transfer learning will be performed using the VGG-16 model, which is also explained in the method section. The model used is a model that was previously trained on 1000 different classes. For our application, we will retrain the previously trained model with our own images and observe the results.The summary of the model used is as follows.



```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 32, 32, 3)]       0

block1_conv1 (Conv2D)        (None, 32, 32, 64)        1792

block1_conv2 (Conv2D)        (None, 32, 32, 64)        36928

block1_pool (MaxPooling2D)   (None, 16, 16, 64)        0

block2_conv1 (Conv2D)        (None, 16, 16, 128)       73856

block2_conv2 (Conv2D)        (None, 16, 16, 128)       147584

block2_pool (MaxPooling2D)   (None, 8, 8, 128)         0

block3_conv1 (Conv2D)        (None, 8, 8, 256)         295168

block3_conv2 (Conv2D)        (None, 8, 8, 256)         590080

block3_conv3 (Conv2D)        (None, 8, 8, 256)         590080

block3_pool (MaxPooling2D)   (None, 4, 4, 256)         0

global_average_pooling2d (Gl (None, 256)               0

batch_normalization (BatchNo (None, 256)               1024

dense (Dense)                (None, 256)               65792

dense_1 (Dense)              (None, 256)               65792

dropout (Dropout)            (None, 256)               0

dense_2 (Dense)              (None, 10)                2570
=================================================================
Total params: 1,870,666
Trainable params: 134,666
Non-trainable params: 1,736,000
```

Figure 3: VGG-16 Model Summary

The training process was carried out for 40 epochs. After the training was completed, the test score was obtained with the help of the model.predict() command on the test data set. As a result, the accuracy score was 0.75.

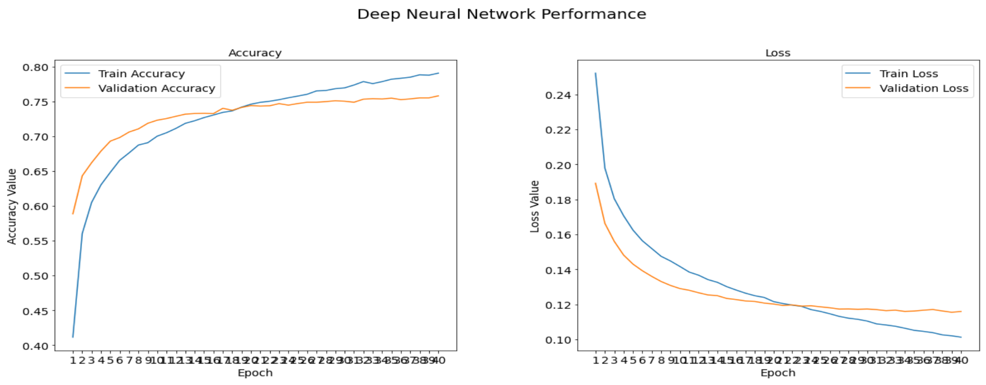


Figure 4: Accuracy & Loss Curves of VGG-16

The Confusion Matrix is shown in the figure below. Confusion matrix is an important matrix that contains the relationship between actual and predicted values. When this matrix is examined, it is clearly seen which categories cannot be separated from each other well, and for which classes the trained model is more inadequate.

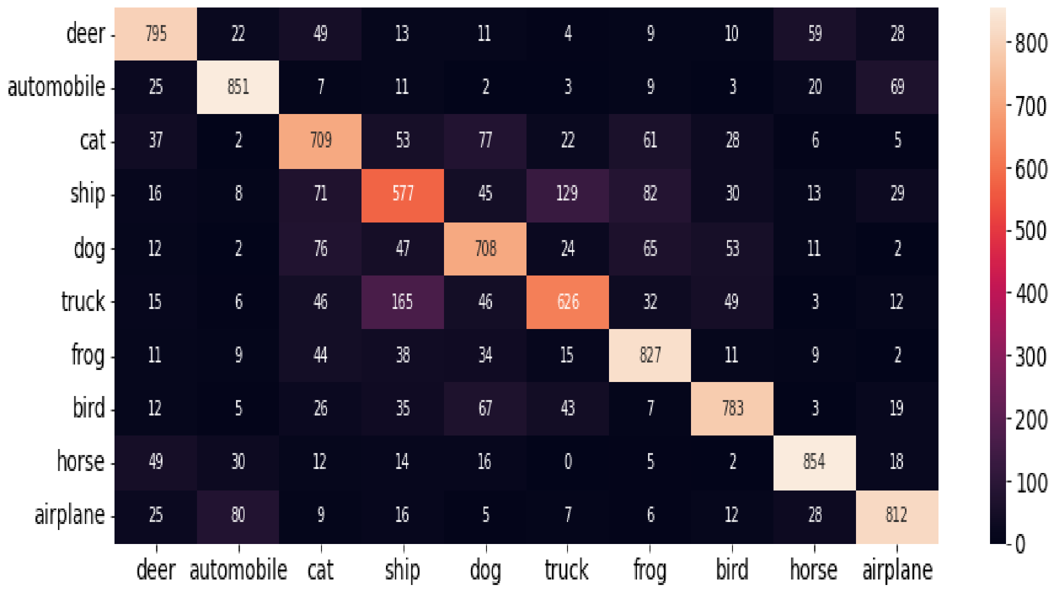


Figure 5: Confusion Matrix of Transfer Learning Model(VGG-16)

The Classification Report is shown in the table below.

Classification Report				
	Precision	Recall	F1-Score	Support
0	0.80	0.80	0.80	1000
1	0.84	0.85	0.84	1000
2	0.68	0.71	0.69	1000
3	0.60	0.58	0.59	1000
4	0.70	0.71	0.70	1000
5	0.72	0.63	0.67	1000
6	0.75	0.83	0.79	1000
7	0.80	0.78	0.79	1000
8	0.85	0.85	0.85	1000
9	0.82	0.81	0.81	1000
Accuracy			0.75	10000
Macro avg	0.75	0.75	0.75	10000
Weighted avg	0.75	0.75	0.75	10000

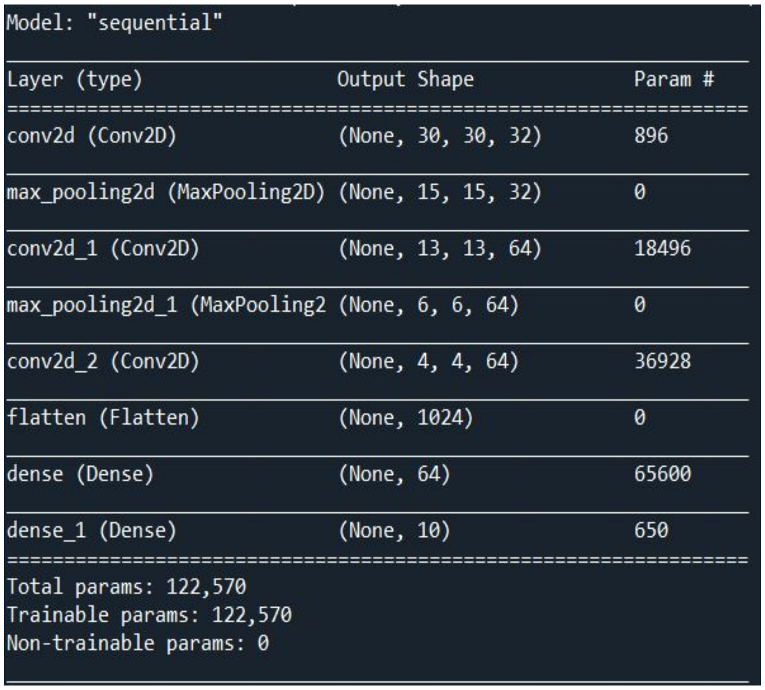
Table 2: Classification Report of Transfer Learning Model(VGG-16)

6.2 CNN Structure (Kernel=3x3)

First of all, the number of epochs was adjusted too much to determine after which epoch the system was exposed to the overfitting problem.

6.2.1 Number of Epoch (40)

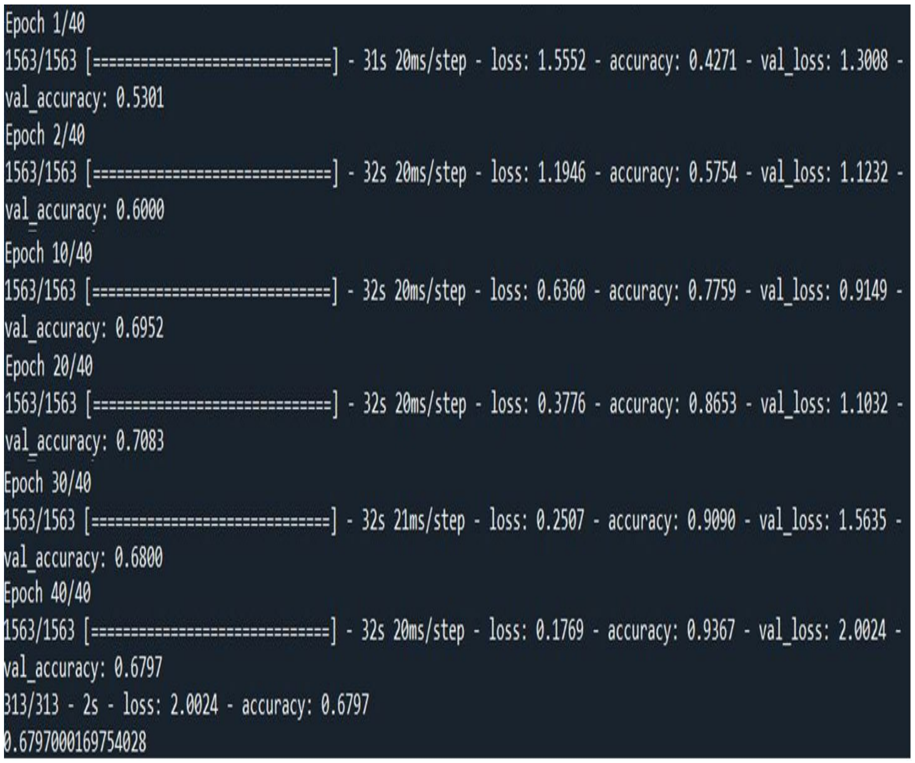
The CNN architecture used and the number of parameters are shown in Figure 6.



```
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, 64)          36928
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 64)                65600
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650
=================================================================
Total params: 122,570
Trainable params: 122,570
Non-trainable params: 0
_________________________________________________________________
```

Figure 6: CNN Model Summary

Then the training of the model for the CIFAR dataset was started. As seen in Figure 7, there is a training process that takes 40 epochs. Train accuracy stood at 93%, while Test accuracy remained at 68%.



```
Epoch 1/40
1563/1563 [==============================] - 31s 20ms/step - loss: 1.5552 - accuracy: 0.4271 - val_loss: 1.3008 -
val_accuracy: 0.5301
Epoch 2/40
1563/1563 [==============================] - 32s 20ms/step - loss: 1.1946 - accuracy: 0.5754 - val_loss: 1.1232 -
val_accuracy: 0.6000
Epoch 10/40
1563/1563 [==============================] - 32s 20ms/step - loss: 0.6360 - accuracy: 0.7759 - val_loss: 0.9149 -
val_accuracy: 0.6952
Epoch 20/40
1563/1563 [==============================] - 32s 20ms/step - loss: 0.3776 - accuracy: 0.8653 - val_loss: 1.1032 -
val_accuracy: 0.7083
Epoch 30/40
1563/1563 [==============================] - 32s 21ms/step - loss: 0.2507 - accuracy: 0.9090 - val_loss: 1.5635 -
val_accuracy: 0.6800
Epoch 40/40
1563/1563 [==============================] - 32s 20ms/step - loss: 0.1769 - accuracy: 0.9367 - val_loss: 2.0024 -
val_accuracy: 0.6797
313/313 - 2s - loss: 2.0024 - accuracy: 0.6797
0.6797000169754028
```

Figure 7: Training of CNN Model

When the accuracy curve is examined, the importance of the validation process is seen. Because the accuracy curve of the train set tends to mislead us. On the contrary, the validation set informs us whether the system has gained the ability to generalize or has not fallen into a memorization error. As seen here, it starts memorizing after about epoch 10.

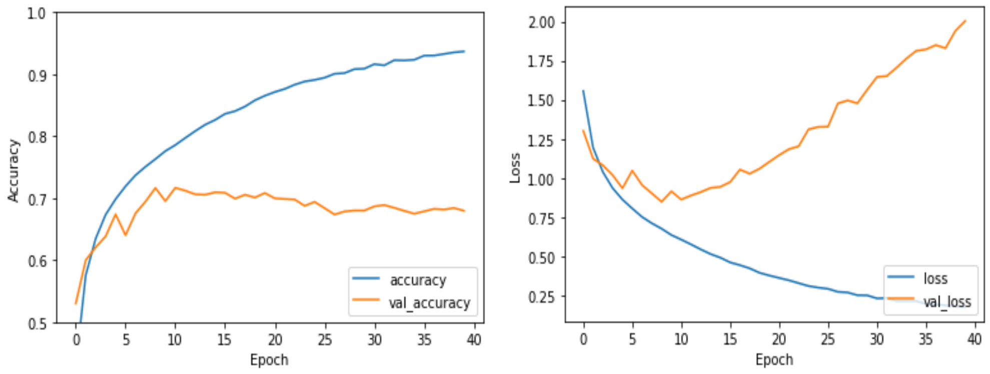
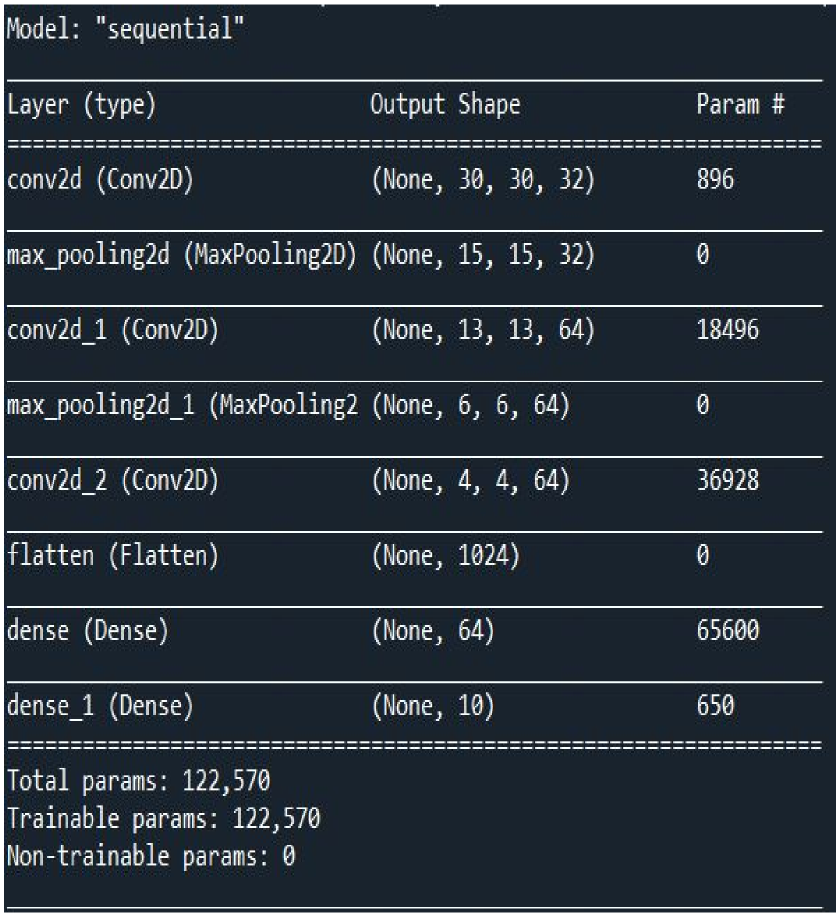


Figure 8: Accuracy & Loss Curves of CNN

6.2.2 Number of Epoch (10)

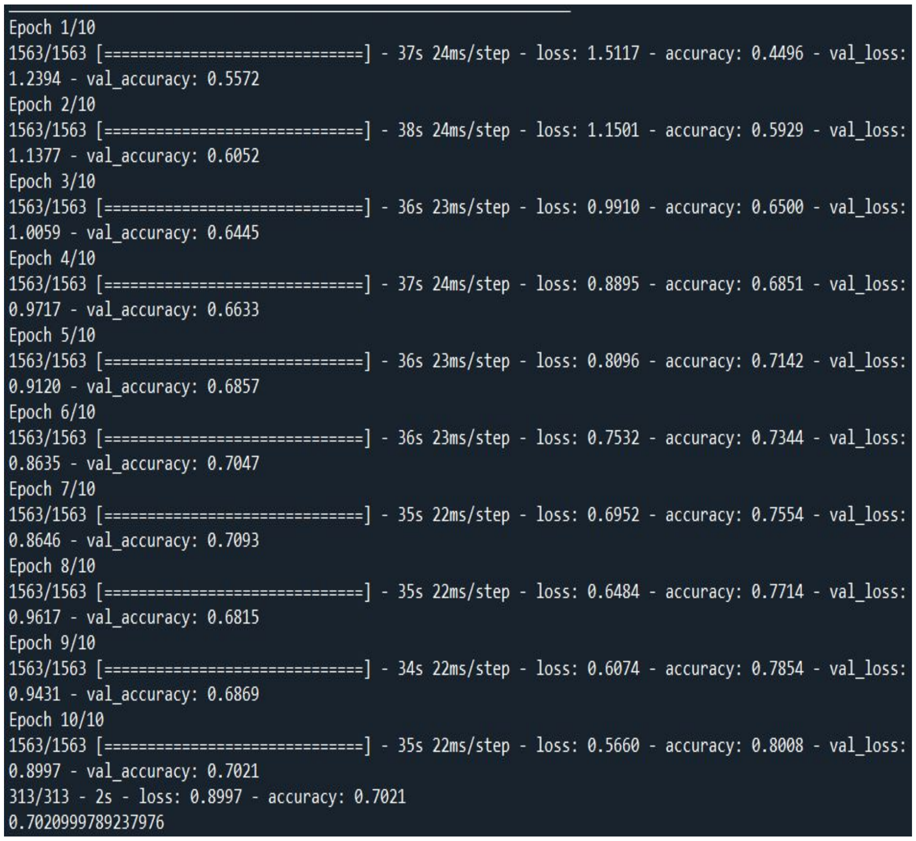
The CNN architecture used and the number of parameters are shown in Figure 9.



```
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, 64)          36928
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 64)                65600
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650
=================================================================
Total params: 122,570
Trainable params: 122,570
Non-trainable params: 0
_________________________________________________________________
```

Figure 9: CNN Model Summary

Then the training of the model for the CIFAR dataset was started. As seen in Figure 10, there is a training process that takes 10 epochs. Train accuracy stood at 80%, while Test accuracy remained at 70%.



```
Epoch 1/10
1563/1563 [==============================] - 37s 24ms/step - loss: 1.5117 - accuracy: 0.4496 - val_loss:
1.2394 - val_accuracy: 0.5572
Epoch 2/10
1563/1563 [==============================] - 38s 24ms/step - loss: 1.1501 - accuracy: 0.5929 - val_loss:
1.1377 - val_accuracy: 0.6052
Epoch 3/10
1563/1563 [==============================] - 36s 23ms/step - loss: 0.9910 - accuracy: 0.6500 - val_loss:
1.0059 - val_accuracy: 0.6445
Epoch 4/10
1563/1563 [==============================] - 37s 24ms/step - loss: 0.8895 - accuracy: 0.6851 - val_loss:
0.9717 - val_accuracy: 0.6633
Epoch 5/10
1563/1563 [==============================] - 36s 23ms/step - loss: 0.8096 - accuracy: 0.7142 - val_loss:
0.9120 - val_accuracy: 0.6857
Epoch 6/10
1563/1563 [==============================] - 36s 23ms/step - loss: 0.7532 - accuracy: 0.7344 - val_loss:
0.8635 - val_accuracy: 0.7047
Epoch 7/10
1563/1563 [==============================] - 35s 22ms/step - loss: 0.6952 - accuracy: 0.7554 - val_loss:
0.8646 - val_accuracy: 0.7093
Epoch 8/10
1563/1563 [==============================] - 35s 22ms/step - loss: 0.6484 - accuracy: 0.7714 - val_loss:
0.9617 - val_accuracy: 0.6815
Epoch 9/10
1563/1563 [==============================] - 34s 22ms/step - loss: 0.6074 - accuracy: 0.7854 - val_loss:
0.9431 - val_accuracy: 0.6869
Epoch 10/10
1563/1563 [==============================] - 35s 22ms/step - loss: 0.5660 - accuracy: 0.8008 - val_loss:
0.8997 - val_accuracy: 0.7021
313/313 - 2s - loss: 0.8997 - accuracy: 0.7021
0.7020999789237976
```

Figure 10: Training of CNN Model

When the accuracy curve is examined, it is seen that there is no overfitting problem like the 40 epoch training process.

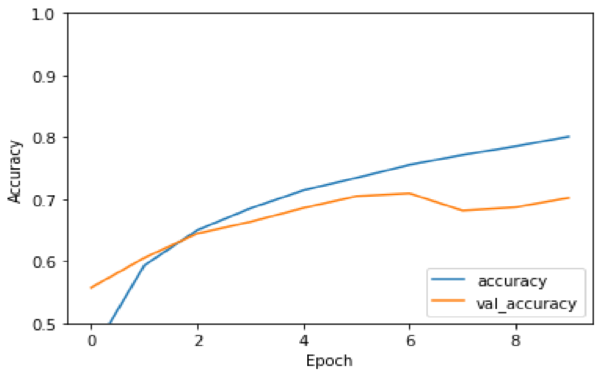


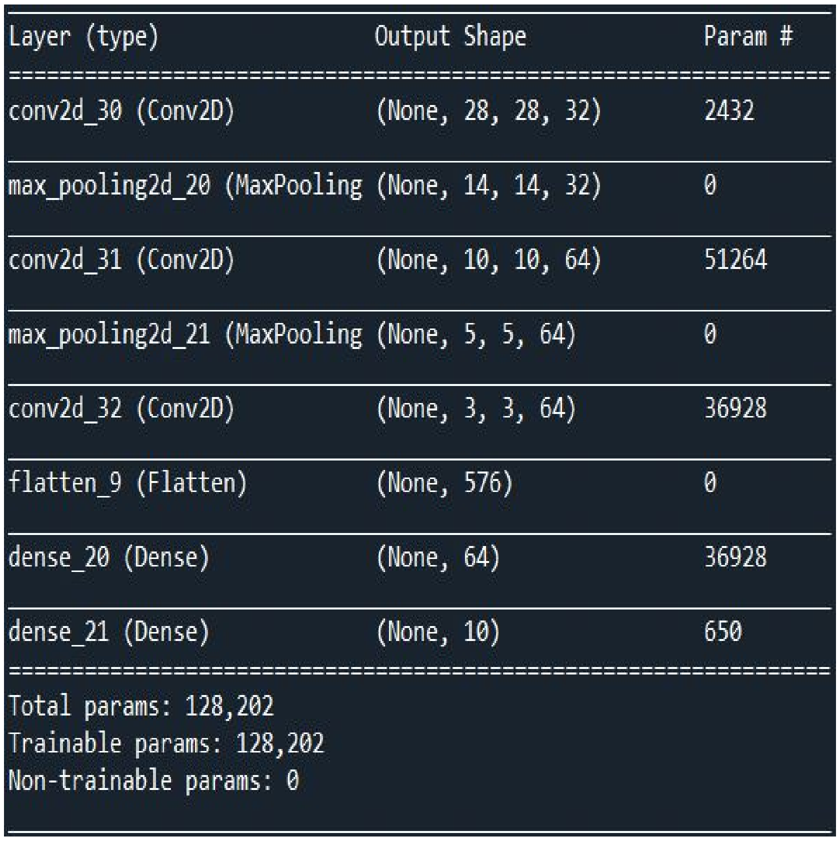
Figure 11: Accuracy Curve of CNN

6.3 CNN Structure (Kernel=5x5)

First of all, the number of epochs was adjusted too much to determine after which epoch the system was exposed to the overfitting problem.

6.3.1 Number of Epoch (40)

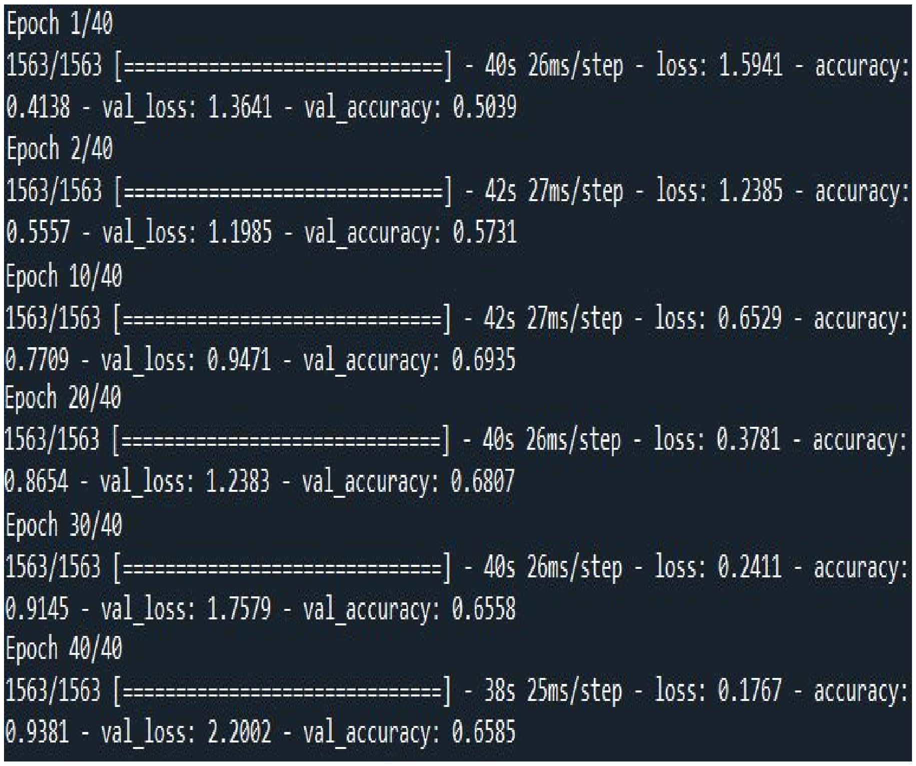
The CNN architecture used and the number of parameters are shown in Figure 12.



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_30 (Conv2D)           (None, 28, 28, 32)        2432
_________________________________________________________________
max_pooling2d_20 (MaxPooling (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_31 (Conv2D)           (None, 10, 10, 64)        51264
_________________________________________________________________
max_pooling2d_21 (MaxPooling (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_32 (Conv2D)           (None, 3, 3, 64)          36928
_________________________________________________________________
flatten_9 (Flatten)          (None, 576)               0
_________________________________________________________________
dense_20 (Dense)             (None, 64)                36928
_________________________________________________________________
dense_21 (Dense)             (None, 10)                650
=================================================================
Total params: 128,202
Trainable params: 128,202
Non-trainable params: 0
_________________________________________________________________
```

Figure 12: CNN Model Summary

Then the training of the model for the CIFAR dataset was started. As seen in Figure 13, there is a training process that takes 40 epochs. Train accuracy stood at 94%, while Test accuracy remained at 65%.



```
Epoch 1/40
1563/1563 [==============================] - 40s 26ms/step - loss: 1.5941 - accuracy:
0.4138 - val_loss: 1.3641 - val_accuracy: 0.5039
Epoch 2/40
1563/1563 [==============================] - 42s 27ms/step - loss: 1.2385 - accuracy:
0.5557 - val_loss: 1.1985 - val_accuracy: 0.5731
Epoch 10/40
1563/1563 [==============================] - 42s 27ms/step - loss: 0.6529 - accuracy:
0.7709 - val_loss: 0.9471 - val_accuracy: 0.6935
Epoch 20/40
1563/1563 [==============================] - 40s 26ms/step - loss: 0.3781 - accuracy:
0.8654 - val_loss: 1.2383 - val_accuracy: 0.6807
Epoch 30/40
1563/1563 [==============================] - 40s 26ms/step - loss: 0.2411 - accuracy:
0.9145 - val_loss: 1.7579 - val_accuracy: 0.6558
Epoch 40/40
1563/1563 [==============================] - 38s 25ms/step - loss: 0.1767 - accuracy:
0.9381 - val_loss: 2.2002 - val_accuracy: 0.6585
```

Figure 13: Training of CNN Model

When the accuracy curve is examined, the importance of the validation process is seen. Because the accuracy curve of the train set tends to mislead us. On the contrary, the validation set informs us whether the system has gained the ability to generalize or has not fallen into a memorization error. As seen here, it starts memorizing after about epoch 10.

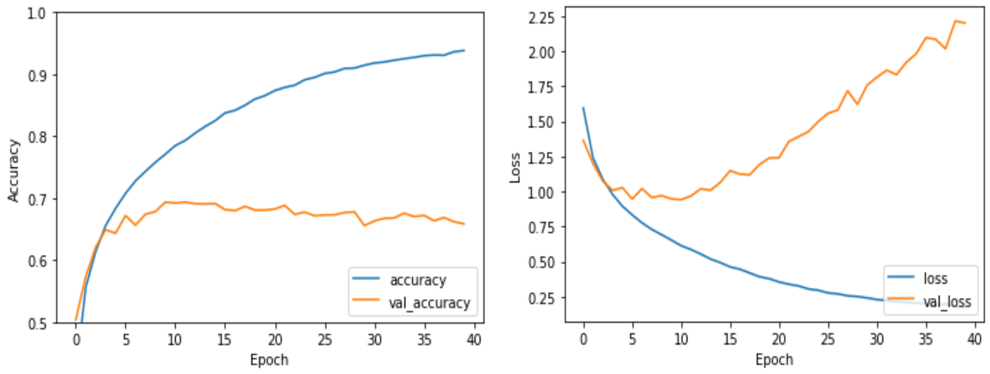
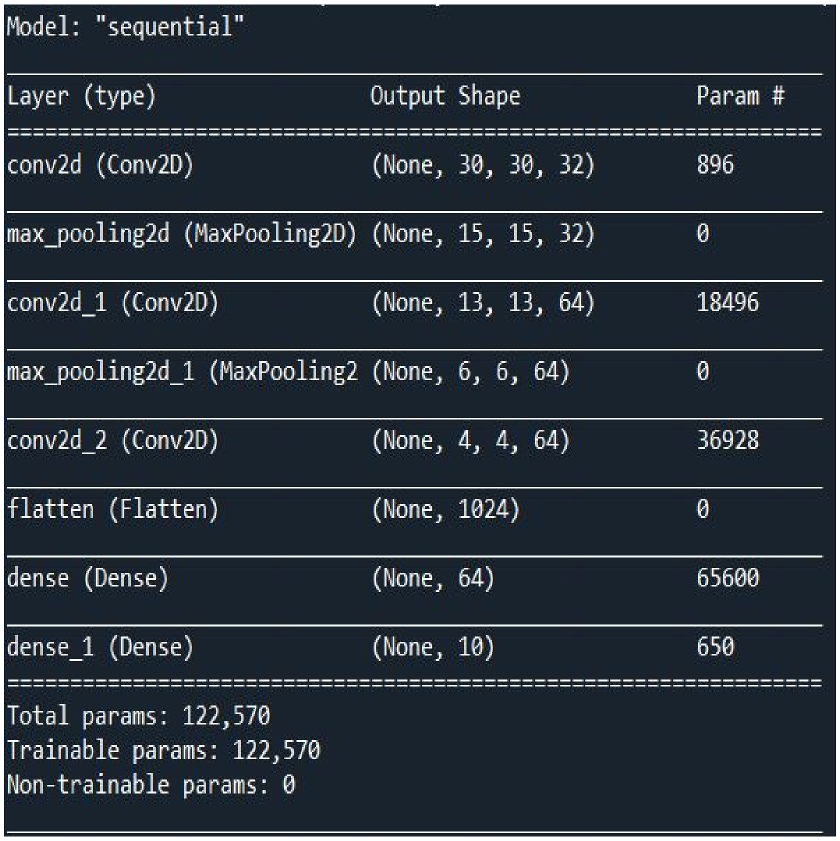


Figure 14: Accuracy - Loss Curves of CNN

6.3.2 Number of Epoch (10)

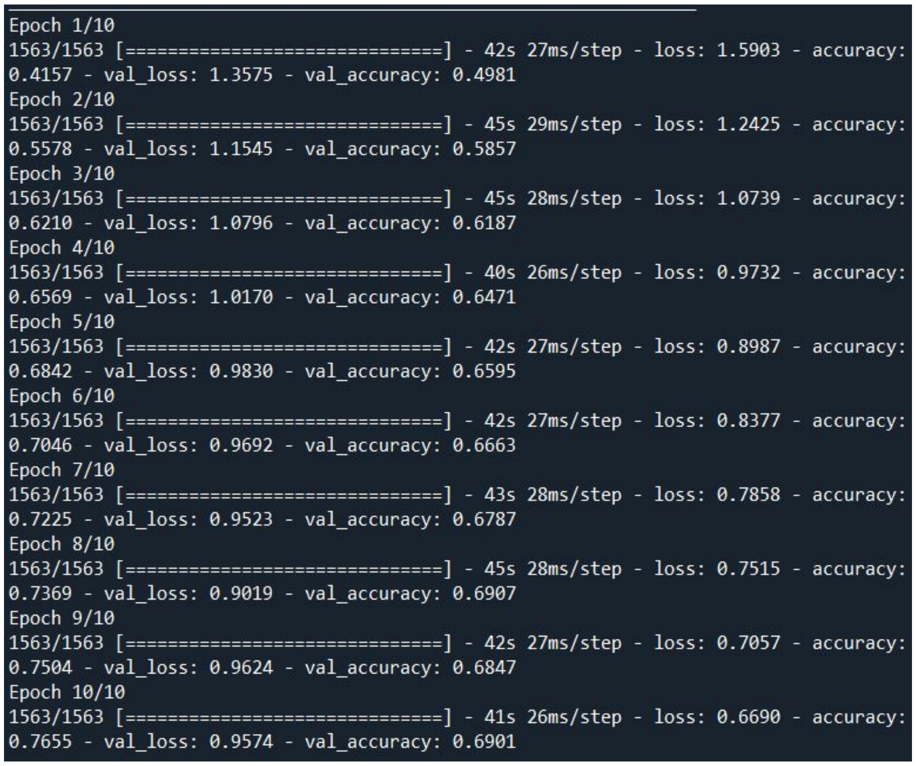
The CNN architecture used and the number of parameters are shown in figure 15.



```
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, 64)          36928
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 64)                65600
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650
=================================================================
Total params: 122,570
Trainable params: 122,570
Non-trainable params: 0
_________________________________________________________________
```

Figure 15: CNN Model Summary

Then the training of the model for the CIFAR dataset was started. As seen in Figure 16, there is a training process that takes 10 epochs. Train accuracy stood at 76%, while Test accuracy remained at 69%.



```
Epoch 1/10
1563/1563 [==============================] - 42s 27ms/step - loss: 1.5903 - accuracy:
0.4157 - val_loss: 1.3575 - val_accuracy: 0.4981
Epoch 2/10
1563/1563 [==============================] - 45s 29ms/step - loss: 1.2425 - accuracy:
0.5578 - val_loss: 1.1545 - val_accuracy: 0.5857
Epoch 3/10
1563/1563 [==============================] - 45s 28ms/step - loss: 1.0739 - accuracy:
0.6210 - val_loss: 1.0796 - val_accuracy: 0.6187
Epoch 4/10
1563/1563 [==============================] - 40s 26ms/step - loss: 0.9732 - accuracy:
0.6569 - val_loss: 1.0170 - val_accuracy: 0.6471
Epoch 5/10
1563/1563 [==============================] - 42s 27ms/step - loss: 0.8987 - accuracy:
0.6842 - val_loss: 0.9830 - val_accuracy: 0.6595
Epoch 6/10
1563/1563 [==============================] - 42s 27ms/step - loss: 0.8377 - accuracy:
0.7046 - val_loss: 0.9692 - val_accuracy: 0.6663
Epoch 7/10
1563/1563 [==============================] - 43s 28ms/step - loss: 0.7858 - accuracy:
0.7225 - val_loss: 0.9523 - val_accuracy: 0.6787
Epoch 8/10
1563/1563 [==============================] - 45s 28ms/step - loss: 0.7515 - accuracy:
0.7369 - val_loss: 0.9019 - val_accuracy: 0.6907
Epoch 9/10
1563/1563 [==============================] - 42s 27ms/step - loss: 0.7057 - accuracy:
0.7504 - val_loss: 0.9624 - val_accuracy: 0.6847
Epoch 10/10
1563/1563 [==============================] - 41s 26ms/step - loss: 0.6690 - accuracy:
0.7655 - val_loss: 0.9574 - val_accuracy: 0.6901
```

Figure 16: Training of CNN Model

When the accuracy curve is examined, it is seen that there is no overfitting problem like the 40 epoch training process.

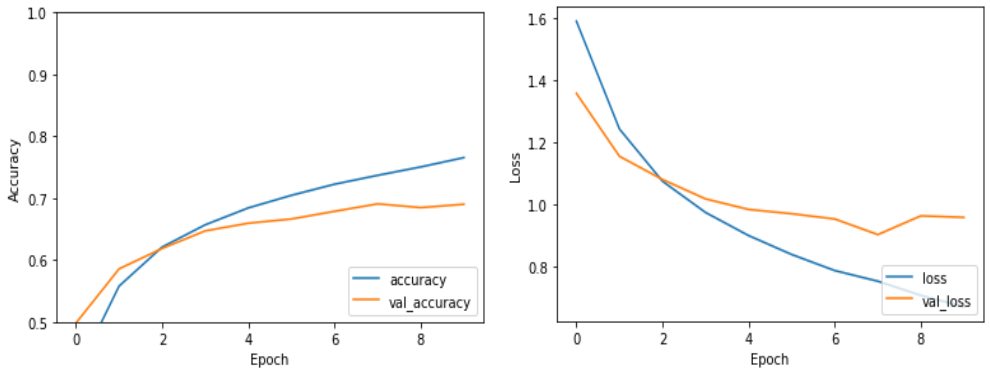


Figure 17: Accuracy - Loss Curves of CNN

7 Conclusion

This study was carried out on distinguishing images from 10 different classes in the CIFAR-10 data set. In this study, the trained model and the methods used in the recognition of images from different classes are described with an approach using Convolutional Neural Networks (CNN). It has been observed how the use of different hyperparameters affects the success of the model. In addition, the application of the transfer learning method is shown on the VGG-16 model. A comparison was made between their success and it can be said that the transfer learning method gave better results for our application than the classical CNN method. One of the reasons for the difference in success here is the number of parameters. It can be said that the number of parameters of the VGG-16 model is approximately 10 times higher than the CNN structure we have created, contributing positively to the success of the system. However, better performance can be achieved by making different hyperparameter changes (padding, stride, kernel size, depth of CNN, learning rate, momentum term) or with different CNN structures.

References

Tien Ho-Phuoc. Cifar10 to compare visual recognition performance between deep neural networks and humans. *arXiv preprint arXiv:1811.07270*, 2018.

James J DiCarlo, Davide Zoccolan, and Nicole C Rust. How does the brain solve visual object recognition? *Neuron*, 73(3):415–434, 2012.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Yoshua Bengio. *Learning deep architectures for AI*. Now Publishers Inc, 2009.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Daniel Graupe. *Principles of artificial neural networks*, volume 7. World Scientific, 2013.

Pin Wang, En Fan, and Peng Wang. Comparative analysis of image classification algorithms based on traditional machine learning and deep learning. *Pattern Recognition Letters*, 2020.

Nikhil Ketkar and Eder Santana. *Deep Learning with Python*, volume 1. Springer, 2017.

Farhana Sultana, Abu Sufian, and Paramartha Dutta. Advancements in image classification using convolutional neural network. In *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, pages 122–129. IEEE, 2018.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106, 1962.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Wei-Chang Yeh, Yi-Ping Lin, Yun-Chia Liang, and Chyh-Ming Lai. Convolution neural network hyperparameter optimization using simplified swarm optimization. *arXiv preprint arXiv:2103.03995*, 2021.

Elad Hazan, Adam Klivans, and Yang Yuan. Hyperparameter optimization: A spectral approach. *arXiv preprint arXiv:1706.00764*, 2017.

Xiang Zhang, Xiaocong Chen, Lina Yao, Chang Ge, and Manqing Dong. Deep neural network hyperparameter optimization with orthogonal array tuning. In *International conference on neural information processing*, pages 287–295. Springer, 2019.

Transfer learning — Wikipedia, the free encyclopedia, 2021.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Muneeb ul Hassan. Vgg16 – convolutional network for classification and detection.

Mohammad Hossin and MN Sulaiman. A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining & Knowledge Management Process*, 5(2):1, 2015.

Claude Sammut and Geoffrey I Webb. *Encyclopedia of machine learning*. 2011.

Confusion matrix — Wikipedia, the free encyclopedia, 2021.