# Tribhuvan University

## Institute of Science and Technology

**Seminar Report**

**On**

# "A Comparative Study of Convolution Neural Network Based on Image Classification"

**Submitted to**

**Central Department of Computer Science and Information Technology**

**Tribhuvan University, Kirtipur**

**Kathmandu, Nepal**

**Submitted by**

**Akkal Bahadur Bist**

Roll No. 19/079

**In partial fulfillment of the requirement for Master's Degree in Computer Science and Information technology (M.Sc. CSIT)**

**First semester**

June, 2023

# Tribhuvan University

# Institute of Science and Technology

## Supervisor Recommendation

This is to certify that Mr. Akkal Bahadur Bist (Roll No. 19/079) has submitted the seminar report on the topic **"A Comparative Study of Convolution Neural Network Based on Image Classification"** for the partial fulfilment of Master's of Science in Computer Science and Information Technology, first semester. I hereby, declare that this seminar report has been approved.

Supervisor

Asst. Prof. Mr. Arjun Singh Saud

Central Department of Computer Science and Information Technology

# Letter of Approval

This is to certify that the seminar report prepared by Mr. Akkal Bahadur Bist entitled **"A Comparative Study of Convolution Neural Network Based on Image Classification"** in partial fulfilment of the requirements for the degree of Master's of Science in Computer Science and Information Technology has been well studied. In our opinion, it is satisfactory in the scope and quality as a project for the required degree.

Evaluation Committee

| | |
|---|---|
| ………….……………………………….. | ………..………………………………… |
| Asst. Prof. Sarbin Sayami | Asst. Prof. Arjun Singh Saud |
| (H.O.D) | (Supervisor) |
| Central Department of Computer Science and Information Technology | Central Department of Computer Science and Information Technology |

……………………………

(Internal)

## Acknowledgement

The success and final outcome of this report required a lot of guidance and assistance from many people and I am very fortunate to have got this all along the completion. I am very glad to express my deepest sense of gratitude and sincere thanks to my highly respected and esteemed supervisor **Asst. Prof. Arjun Singh Saud,** Central Department of Computer Science and Information Technology for his valuable supervision, guidance, encouragement, and support for completing this paper.

I am also thankful to **Asst. Prof. Sarbin Sayami**, Head of Central Department of Computer Science and Information Technology for his invaluable knowledge sharing and constant support throughout the period. Furthermore, with immense pleasure, I submit by deepest gratitude to the Central Department of Computer Science and Information Technology, Tribhuvan University, and all the faculty members of CDCSIT for providing the platform to explore the knowledge of interest. At the end I would like to express my sincere thanks to all my friends and others who helped me directly or indirectly.

**Akkal Bahadur Bist**

**Roll No: 19/079**

# Abstract

Apart from image processing techniques, the problem of object recognition can also be solved by using machine learning techniques. The main concern of this seminar is to classify images using machine learning techniques and compare those machine learning techniques or algorithms. To tackle such problems, artificial neural network i.e. Convolutional Neural Network has been developed. In order to design the Convolutional Neural Network, different parameters like filter size, number of convolution layer, drop out layer, etc. were taken. Careful choosing and study of these parameters shows that efficient architecture can be designed. Different neural network architecture for CIFAR-10 dataset have been developed. Being different in terms of number of hidden layers, filter size and other measures. They have been trained on Central Processing Unit. Drop out techniques has been used to reduce over-fitting issues. Accuracy of these architecture has been calculated by feeding the neural networks with the test data. Lastly, results are compared and analyzed to find out best architecture. Thus this study gives a way to design efficient architecture for image classification.

**Keywords:** *Neural Network, Convolution Layer (CONV), Pooling Layer (POOL), Fully Connected layer (FC), Confusion Matrix, Learning Rate, VGG16, VGG19, CIFAR-10, Tensor flow, Keas, CNN.*

# Table of Contents

# List of Tables

# List of Figures

# List of Abbreviations

| | |
|---|---|
| ASCII | American Standard Code for Information Interchange |
| AI | Artificial Intelligence |
| CNN | Convolutional Neural Network |
| CONV | Convolution Neural Network |
| DNN | Deep Neural Network |
| POOL | Pooling Layer |
| RELU | Rectified Linear Unit |
| RGB | Red, Green, and Blue |
| RNN | Recurrent Neural Network |
| SGD | Stochastic Gradient Descent |

# Chapter 1: Introduction

## 1.1 Background

Image classification, as a classical research topic in recent years, is one of the core issues of computer vision and the basis of various fields of visual recognition. Improving classification network performance enhances object detection, segmentation, pose estimation, video classification, object tracking, and super-resolution technology. Traditional methods for image feature extraction have limitations in generalization and portability. Artificial Neural Networks (ANN), inspired by biological vision, simulate neural network processing. Convolutional Neural Networks (CNN) emerged as a powerful tool for image classification. CNN models like VGG, LeNet-5, AlexNet, ImageNet, ResNet,m GoogleNet etc. CNNs have also found successful applications in remote sensing image analysis. Various CNN-based scene classification methods have been developed, including using pre-trained CNNs as feature extractors, fine-tuning them on datasets, or globally initializing weights for training. This report provides a comparative accuracy of CNN VGG16 and VGG19 model, their development in image classification. The review covers the basic structure and principles of neural networks, necessary network layers in CNNs, classic network models, classical model training strategies, commonly used datasets, and a future research direction. This report used a special type of Convolutional Neural Network known as VGG model for image classification using encrypted dataset known as CIFER10 datasets.

## 1.2 Problem Statement

Now a day the application of artificial intelligence is growing in various sector to make the decision automatic or to enhance the system so that they can cope with the changing environment. Neural Network is a special technique in artificial intelligence field. Deep Neural Networks are applicable to fields like image recognition, Speech Recognition, text prediction and handwriting generation, Language Generations, pattern recognitions. DNN architecture for one task does not work well with another task. For task related to specific field, it is required to define which architecture to pick. Image Classification or Object Detection task and designing its structure for given datasets is challenging job.

Developing the neural network architecture-based image classification is a challenging job and also a booming topic in research community. This motivates me to take this seminar as an opportunity to work in this sector. I realize computer vision is more related to recognition and classification using Neural Network Algorithms.

## 1.3 Objective

The main objective of this seminar are:

- To classify image using Convolutional Neural Network.
- To compare and evaluate the accuracy of image classification between the VGG16 and VGG19 models.

# Chapter 2: Background Study and Literature Review

## 2.1 Background Study

### 2.1.2 Convolutional Neural Network

A CNN consists of an input layer, output layer, as well as multiple hidden layers. The hidden layer of a CNN typically consists of a series of convolutional layers that convolves with a multiplication or other dot product. The activation function is commonly a ReLU layer and is subsequently followed by additional convolutions such as pooling layers, fully connected layers and normalization layers, referred to as hidden layers because their inputs and outputs are masked by the activation function and final convolution.

The final convolution, in turn, often involved back-propagation in order to more accurately weight the end product. Though the layers are colloquially referred to as convolutions, this is only by convention. Mathematically, it is a sliding dot product or cross-correlation. This has significance for the indices in the matrix, in that it affects how weight is determined at a specific index point. Example of a typical CNN is depicted in *Figure 1*.

In the first layer, CNN matched parts rather than the whole image, therefore breaking the image classification process down into smaller parts (features). A 3x3 grid was defined to represent the features extraction by the CNN for evaluation.

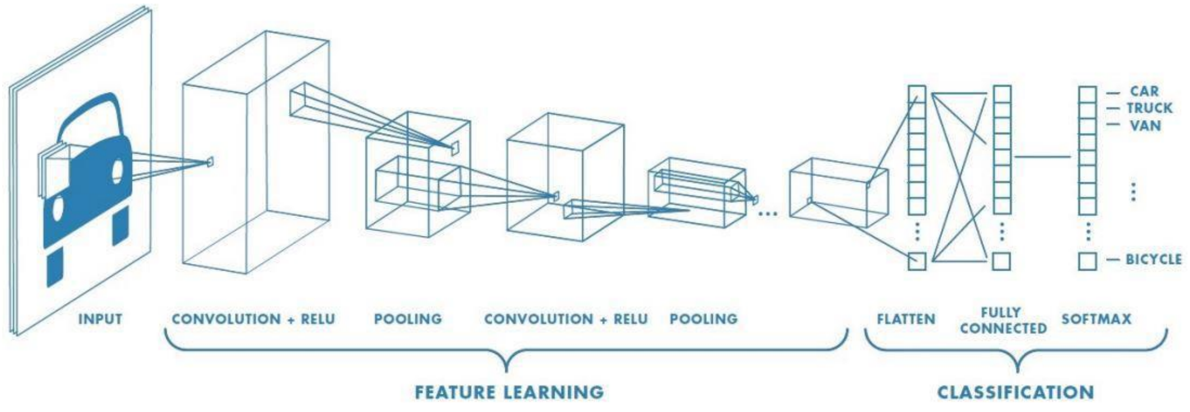


*Figure 1 Typical Structure of CNN*

The following process, known as filtering, involved lining the feature with the image patch. One-by-one, each pixel was multiplied by the corresponding feature pixel, and once completed, all the values were summed and divided by the total number of pixels in the feature space. The final value for the feature was then placed into the feature patch. This process was repeated for the remaining feature patches followed by trying every possible match repeated application of this filter, which is known as a convolution.

| | |
|---|---|
| 1 | 1 |
| 1 | 1 |

(a) A 2 x 2 kernel

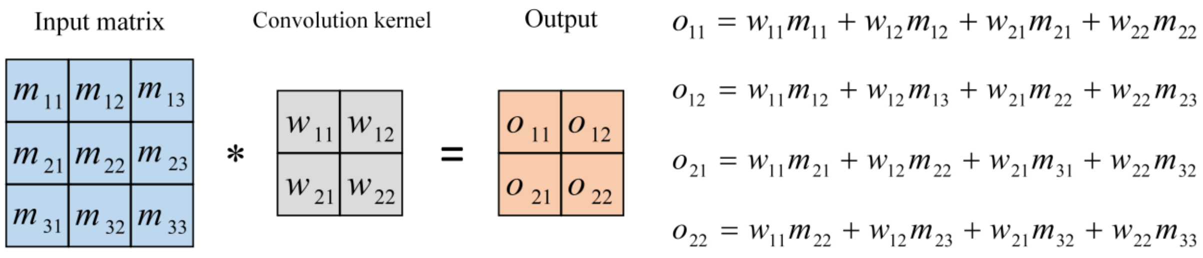
| | | | |
|---|---|---|---|
| 5 | 4 | 7 | 6 |
| 3 | 7 | 9 | 2 |
| 6 | 2 | 1 | 3 |

(b) The convolution input

| | | |
|---|---|---|
| 19 | 27 | 24 |
| 18 | 19 | 15 |

(c) The convolution output

*Table 1: Kernel Matrix, Convolution Input and Output matrix*



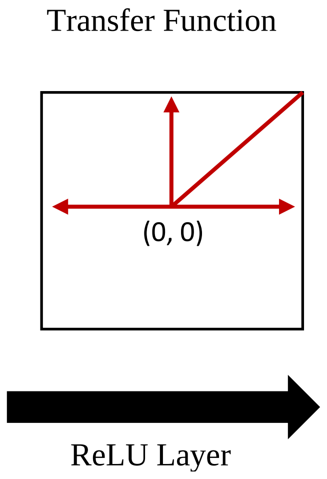
$$\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} = \begin{bmatrix} o_{11} & o_{12} \\ o_{21} & o_{22} \end{bmatrix}$$

$$o_{11} = w_{11}m_{11} + w_{12}m_{12} + w_{21}m_{21} + w_{22}m_{22}$$
$$o_{12} = w_{11}m_{12} + w_{12}m_{13} + w_{21}m_{22} + w_{22}m_{23}$$
$$o_{21} = w_{11}m_{21} + w_{12}m_{22} + w_{21}m_{31} + w_{22}m_{32}$$
$$o_{22} = w_{11}m_{22} + w_{12}m_{23} + w_{21}m_{32} + w_{22}m_{33}$$

*Figure 2 Convolution operation (2-D), kernel size = 2, strides = 1, padding = 0*

When the convolution kernel was overlapped on top of the input image, the computation of the product between the numbers at the same location in the kernel and the input could be done and a single number could be obtained by summing these products together. For example, when the kernel was overlapped with the top left region in the input, the convolution result at that spatial location was 1×5+1×3+1×4+1×7 = 19. The kernel was then moved down by one pixel and the next convolution result was obtained as: 1×3+1×7+1×6+1×2 = 18. The kernel down was then moved down till it reached the bottom border of the input matrix (image). Then, the kernel was returned to the top, and moved to its right by one element (pixel). The convolution for every possible pixel location was repeated until the kernel was moved to the bottom right corner of the input image.

|  |  |  |  |
|---|---|---|---|
| 15 | 20 | -10 | 35 |
| 18 | -110 | 25 | 100 |
| 20 | -15 | 25 | -10 |
| 101 | 75 | 18 | 23 |



|  |  |  |  |
|---|---|---|---|
| 15 | 20 | 0 | 35 |
| 18 | 0 | 25 | 100 |
| 20 | 0 | 25 | 0 |
| 101 | 75 | 18 | 23 |

*Table 2: ReLU function used matrix.*

The normalization layer of a CNN, also referred to as the process of Rectified Linear Unit (ReLU), involved changing all negative values within the filtered image to 0. The purpose of ReLU was to increase the nonlinearity of the CNN. The output was f(x) = max(0,x).The next layer of a CNN was referred to as “max pooling”, which involved shrinking the image stack. In order to pool an image, the window size had to be defined (e.g. usually 2x2/3x3 pixels) and the stride had to be defined. The window was then filtered across the image in strides, with the max value being recorded for each window. Max pooling was considered as a sample-based discretization process. The objective was to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned. Max pooling reduced the dimensionality of each feature map whilst retaining the most important information.

## 2.2 Previous Works, Discussions and Findings

First successful implementation of CNN was done by Yan LeCunn in 1990’s which is used to read zip codes and digits [1]. CNN has been proved to be a powerful tool for image classification and object detection. It became more powerful when AlexNet [2] won ImageNet ILSVRC Challenge 2012 with significant performance from 2nd runner-up. AlexNet was CNN with some modification to LeNet. It was deeper, bigger and Conv Layers were stacked instead of immediate pooling layer. ZFNet [3] won ImageNet ILSVRC Challenge 2013. It modified AlexNet by varying hyper parameters and using larger filters on convolution layers [4].
Problem of over-fitting was one of disappointments for visual learning community as it degraded the performance of various networks. The network was under complex co-adaptation

on training data. Dropout technique was used by [5] so as to reduce such problem and to better train the networks [6]
Both CNN, along with Recurrent Neural Networks (RNN) are used to solve the problem like image captioning (Fie-Fie and L. [7], 2014; Vinyals, Toshev Bengio, and Erthan, [8] 2014; IIya Sutskever, 2011) [9].

Another article to compare the multi classification performance of three popular transfer learning architectures. VGG16, VGG19 and ResNet50 in this article use CIFAR-10 image dataset [10], that is popular benchmark in image classification. This comparative study by Dr. Vaibhav Kumar [11] A recent work Gender Classification Based on Asian Face using Deep Learning by Tiagrajah V et. al [5]. And main research reference paper at ICLR 2015 implementation VGG model in Very Deep Convolution Networks for Large-Scale Image Recognition. Khan et. al. [12], compressed residual-VGG16 CNN model for big data places image recognition [13].

# Chapter 3: Methodology

The two basic neural networks - CNN and RNN i.e., VGG played vital role for the implementation of the Image Activity Classification and Recognition using Image. I will compare the multi-class classification performance of the two popular transfer learning architectures – VGG16 and VGG19. Both models that I have used are pre-trained on Image Net dataset. For the training model I have to take CIFAR-10 image dataset that is a popular benchmark in image classification. The performance of both models will be compared using confusing matrices and their average accuracy implementation consisted of following steps:

## 3.1. Data Collection

In my comparative study, I will be using the CIFAR-10 dataset [2] that is a publically available image data set provided by the Canadian Institute for Advanced Research (CIFAR). It consists of 60000 32×32 colour images in 10 tracks. There are 50000 training images and 10000 test images in this dataset.

## 3.2 Tools and IDE

- Python with Sci-kit, numpy, scipy, Keras, Tensor flow in Python Jupyter
- Core i5, 8GB RAM

## 3.3 Data Analysis

**Introduction to CIFAR-10 Dataset:**

The CIFAR-10 dataset is a widely used benchmark dataset in the field of computer vision and machine learning. It consists of collection of labeled images, each labeled belonging to one of ten distinct classes like (Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, and Truck) this dataset is split into two main subsets: Training Set and Test Set. This sets of data are used for testing and developing various image classification algorithms, neural network architectures, and image processing techniques. Each Images are 32×32 pixels, making it small compared to many other datasets and it are also categorized into colored channels are RGB images.

**Dataset Details:**

CIFAR-10 dataset contains a total of 60,000 images, divided into 50,000 images for training and 10,000 images for testing.

**Training Set:** It contain 50,000 labeled images with 5,000 images for each class. These subsets of data are used for training and fine-tuning models.

**Testing Set:** It contain 10,000 labeled images with 1,000 images for each class. These subsets of data are used for each class. These subsets of data are used for evaluating the performance and generalization capabilities of the models.

The online data set retrieve from "*(x_train, y_train), (x_test, y_test) = cifar10.load_data()*" contain all data using Keras Library for CIFAR dataset "*from keras.datasets import cifar10*" Contain index with its class. The image has been visualized as:

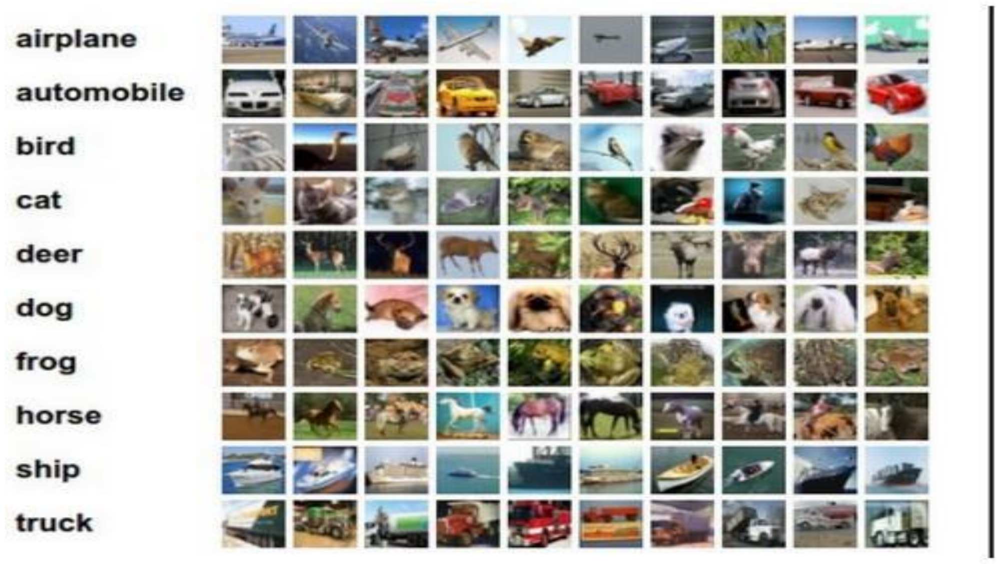


*Figure 3 CIFAR-10 Dataset*

## 3.4 Data Preparation

In this step, the data had to prepare in the manner so that it was convenient to be given as given input learning model. To splitting data set into some training set, validation set and test set for common machine learning model and provide intended to print the dimensions of the CIFER-10 dataset subsets using.

| Training Set | Validation Set | Testing Set |
|---|---|---|

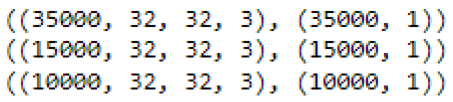
*Table 3: CNN Architecture Layer Model*

**Training Set:** Training set is used to adjust various weight and parameters of the model.

**Validation Set:** Validation Set are used to adjust the parameter of the model instead they are used to reduce over fitting problem.

**Testing Set:** Testing set are used for evaluating the predictive power of the model.

x_train,x_val,y_train,y_val=train_test_split(x_train,y_train,test_size=.3*)*



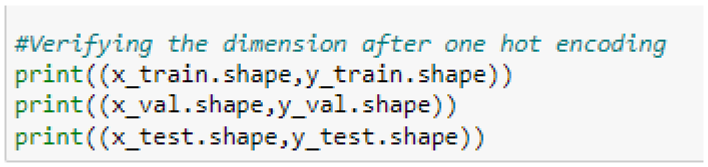
```
((35000, 32, 32, 3), (35000, 1))
((15000, 32, 32, 3), (15000, 1))
((10000, 32, 32, 3), (10000, 1))
```

and

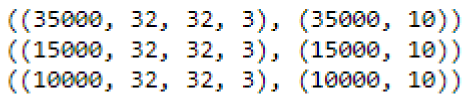
*y_train=to_categorical(y_train)*

*y_val=to_categorical(y_val)*

*y_test=to_categorical(y_test)*



```
#Verifying the dimension after one hot encoding
print((x_train.shape,y_train.shape))
print((x_val.shape,y_val.shape))
print((x_test.shape,y_test.shape))
```



```
((35000, 32, 32, 3), (35000, 10))
((15000, 32, 32, 3), (15000, 10))
((10000, 32, 32, 3), (10000, 10))
```

Also perform one-hot encoding on the target label of the dataset for divide data into categorical data like class, labels, binary vectors etc. and apply image transformation method to augmented data using:

train_generator = ImageDataGenerator(rotation_range=2, horizontal_flip=True, zoom_range=.1 )

val_generator = ImageDataGenerator(rotation_range=2, horizontal_flip=True,zoom_range=.1)

test_generator = ImageDataGenerator(rotation_range=2, horizontal_flip= True, zoom_range=.1)

## 3.5 CNN Architecture Design:

### CNN in Image Classification

CNN is design for sole purpose of Image Classification. The architecture of network will be modified; its filter each changed to see which will give the best output.

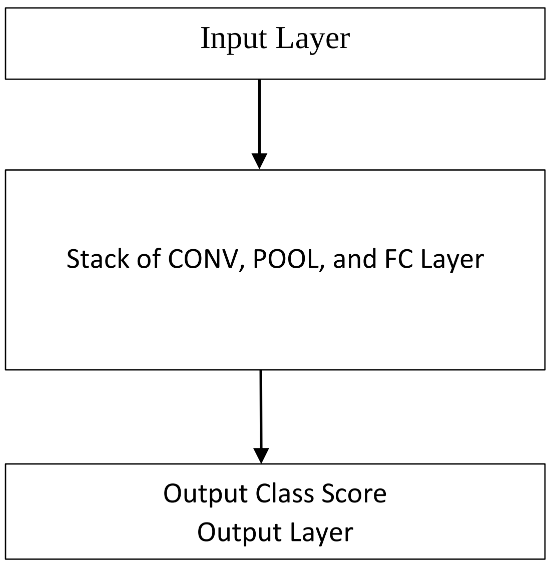


*Figure 4 CNN Design*

**Example:** [INPUT—CONV—RELU—POOL—FC]

As already mentioned, RELU and POOL layers are fixed function but CONV and FC have transformation function whose value depends upon set of parameters (weights and biases). These values are learned through gradient descent.

## 3.6 VGG16 and VGG19 Model

VGG16 and VGG19 are the variants of the VGGNet. VGGNet is a Deep Convolutional Neural Network that was proposed by Karen Simonyan and Andrew Zisserman of the University of Oxford in their research work [3] “Very Deep Convolutional Neural Networks for Large-Scale Image Recognition” [3]. The name of this model was inspired by the name of their research group “Visual Geometry Group (VGG)”. The VGG16 has 16 layers in its architecture while the VGG19 has 19 layers.

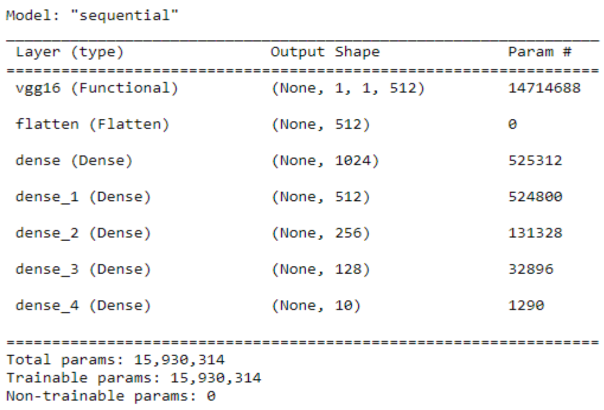
**Implementation of VGG16 and VGG19 Model in Python:**

I will used python language for raining in jupyter Notebook Python editor. Import necessary library for training model with VGG16 architecture is:

Different Layers in CNN are:

I. Input layer is 32x32 images of 3 color channel i.e 3x32x32. Convolution of layer consists of 20 5x5 patches with padding size 2 and stride equal to 1.
II. Pooling layer after first convolution consists of max function with 2x2 patch size. Hence it will reduce the image to 32x14x14.
III. Second convolution layer consists of same properties as that of first convolution layer and the image it produces will be of size 32x10x10 followed by pooling layer that reduces image to 32x5x5
IV. Two full connection layer each of 256 units and with 50% dropout is implemented.
V. Final Output layer with 10 outputs for different classes.

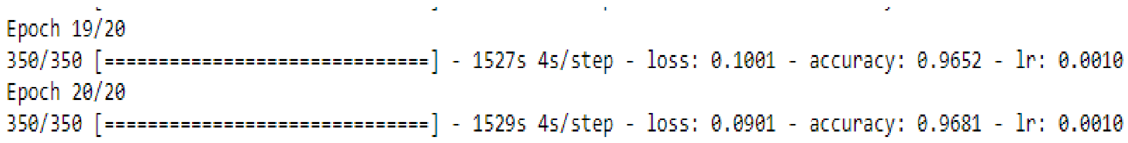
*base_model_vgg16 = VGG16(include_top = False, weights= 'imagenet',*



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 1, 1, 512)         14714688

 flatten (Flatten)           (None, 512)               0

 dense (Dense)               (None, 1024)              525312

 dense_1 (Dense)             (None, 512)               524800

 dense_2 (Dense)             (None, 256)               131328

 dense_3 (Dense)             (None, 128)               32896

 dense_4 (Dense)             (None, 10)                1290

=================================================================
Total params: 15,930,314
Trainable params: 15,930,314
Non-trainable params: 0
```

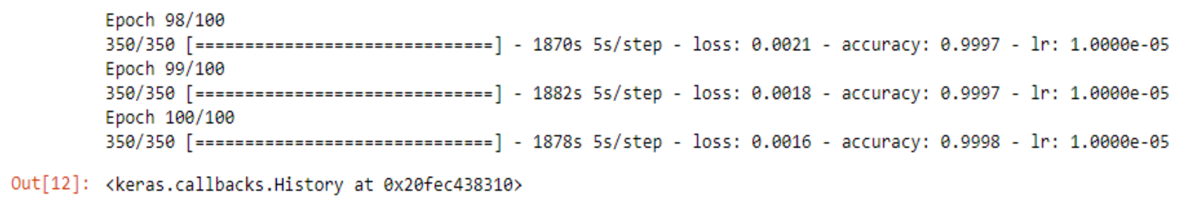
*Table 4: VGG16 Architecture Model*

The accuracy of this architecture using our dataset in 100 after training 100 epoch is very good accuracy 96% accuracy.



```
Epoch 19/20
350/350 [==============================] - 1527s 4s/step - loss: 0.1001 - accuracy: 0.9652 - lr: 0.0010
Epoch 20/20
350/350 [==============================] - 1529s 4s/step - loss: 0.0901 - accuracy: 0.9681 - lr: 0.0010
```

*Figure 5 VGG16 Accuracy in training 100 epoch*



```
Epoch 98/100
350/350 [==============================] - 1870s 5s/step - loss: 0.0021 - accuracy: 0.9997 - lr: 1.0000e-05
Epoch 99/100
350/350 [==============================] - 1882s 5s/step - loss: 0.0018 - accuracy: 0.9997 - lr: 1.0000e-05
Epoch 100/100
350/350 [==============================] - 1878s 5s/step - loss: 0.0016 - accuracy: 0.9998 - lr: 1.0000e-05
Out[12]: <keras.callbacks.History at 0x20fec438310>
```

*Figure 6 VGG19 Accuracy in training 100 epoch*

Both model have training and validation loss are respectively as:

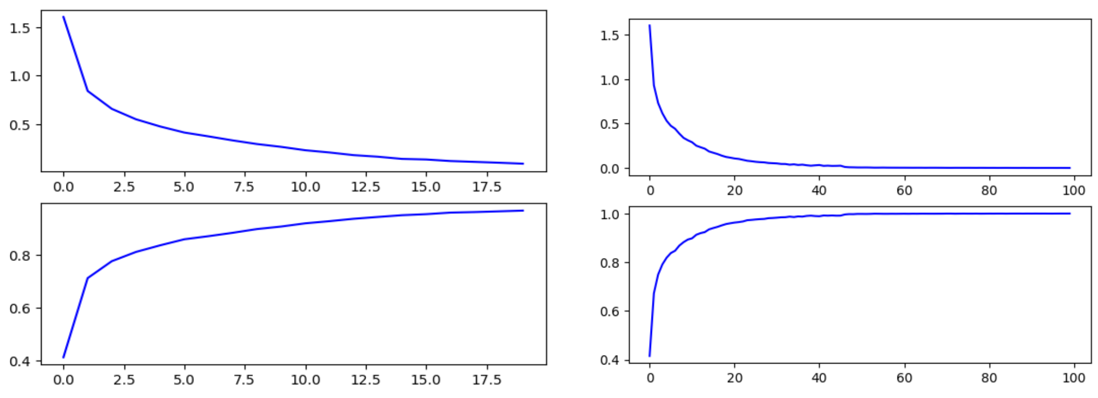


*Figure 7 Training and Validation loss in VGG16 and VGG 19 Model*

We will make the predictions through the trained VGG16 model using the test image dataset is

*y_pred_prob = model_vgg16.predict(x_test)*

*y_pred1 = np.argmax(y_pred_prob, axis=1)*

*y_true = np.argmax(y_test, axis=1)*

313/313[=======] - 81s 256ms/step

And similarly predict the model VGG19 as similar way is

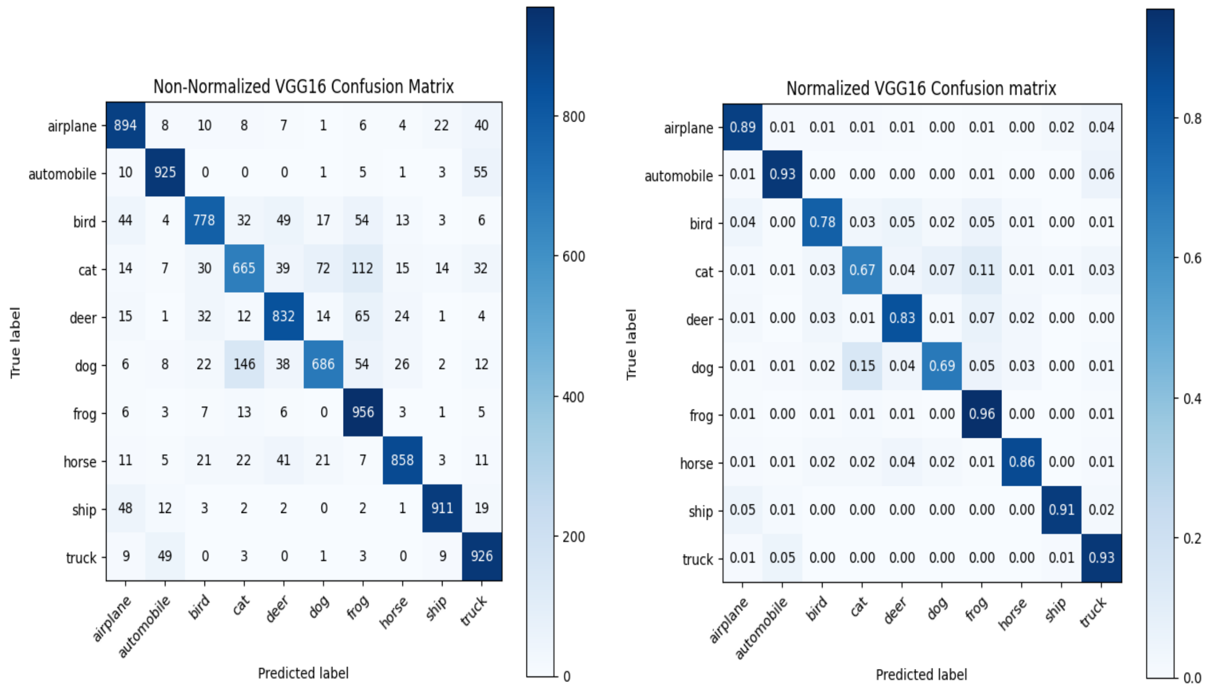
*y_pred_prob = model_vgg19.predict(x_test)*

*y_pred1 = np.argmax(y_pred_prob, axis=1)*

*y_true = np.argmax(y_test, axis=1)*

313/313 [=======] - 63s 201ms/step

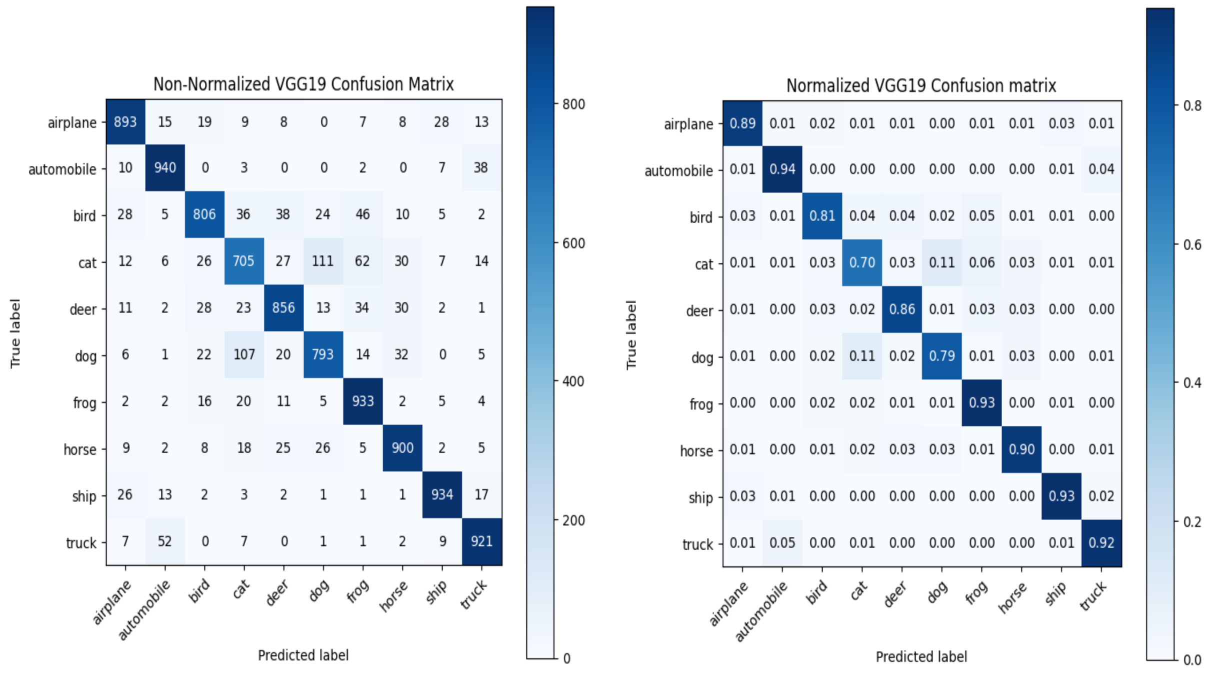
Now, we will plot the non-normalized confusion matrix to visualize the exact number of classifications and normalized confusion matrix to visualize the percentage of classifications.



| | airplane | automobile | bird | cat | deer | dog | frog | horse | ship | truck |
|---|---|---|---|---|---|---|---|---|---|---|
| airplane | 894 | 8 | 10 | 8 | 7 | 1 | 6 | 4 | 22 | 40 |
| automobile | 10 | 925 | 0 | 0 | 0 | 1 | 5 | 1 | 3 | 55 |
| bird | 44 | 4 | 778 | 32 | 49 | 17 | 54 | 13 | 3 | 6 |
| cat | 14 | 7 | 30 | 665 | 39 | 72 | 112 | 15 | 14 | 32 |
| deer | 15 | 1 | 32 | 12 | 832 | 14 | 65 | 24 | 1 | 4 |
| dog | 6 | 8 | 22 | 146 | 38 | 686 | 54 | 26 | 2 | 12 |
| frog | 6 | 3 | 7 | 13 | 6 | 0 | 956 | 3 | 1 | 5 |
| horse | 11 | 5 | 21 | 22 | 41 | 21 | 7 | 858 | 3 | 11 |
| ship | 48 | 12 | 3 | 2 | 2 | 0 | 2 | 1 | 911 | 19 |
| truck | 9 | 49 | 0 | 3 | 0 | 1 | 3 | 0 | 9 | 926 |

| | airplane | automobile | bird | cat | deer | dog | frog | horse | ship | truck |
|---|---|---|---|---|---|---|---|---|---|---|
| airplane | 0.89 | 0.01 | 0.01 | 0.01 | 0.01 | 0.00 | 0.01 | 0.00 | 0.02 | 0.04 |
| automobile | 0.01 | 0.93 | 0.00 | 0.00 | 0.00 | 0.00 | 0.01 | 0.00 | 0.00 | 0.06 |
| bird | 0.04 | 0.00 | 0.78 | 0.03 | 0.05 | 0.02 | 0.05 | 0.01 | 0.00 | 0.01 |
| cat | 0.01 | 0.01 | 0.03 | 0.67 | 0.04 | 0.07 | 0.11 | 0.01 | 0.01 | 0.03 |
| deer | 0.01 | 0.00 | 0.03 | 0.01 | 0.83 | 0.01 | 0.07 | 0.02 | 0.00 | 0.00 |
| dog | 0.01 | 0.01 | 0.02 | 0.15 | 0.04 | 0.69 | 0.05 | 0.03 | 0.00 | 0.01 |
| frog | 0.01 | 0.00 | 0.01 | 0.01 | 0.01 | 0.00 | 0.96 | 0.00 | 0.00 | 0.01 |
| horse | 0.01 | 0.01 | 0.02 | 0.02 | 0.04 | 0.02 | 0.01 | 0.86 | 0.00 | 0.01 |
| ship | 0.05 | 0.01 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.91 | 0.02 |
| truck | 0.01 | 0.05 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.01 | 0.93 |

*Figure 9 VGG16 Model Non-Normalized and Normalized Confusion Matrix*

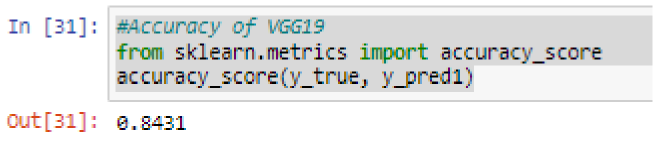


| | airplane | automobile | bird | cat | deer | dog | frog | horse | ship | truck |
|---|---|---|---|---|---|---|---|---|---|---|
| airplane | 893 | 15 | 19 | 9 | 8 | 0 | 7 | 8 | 28 | 13 |
| automobile | 10 | 940 | 0 | 3 | 0 | 0 | 2 | 0 | 7 | 38 |
| bird | 28 | 5 | 806 | 36 | 38 | 24 | 46 | 10 | 5 | 2 |
| cat | 12 | 6 | 26 | 705 | 27 | 111 | 62 | 30 | 7 | 14 |
| deer | 11 | 2 | 28 | 23 | 856 | 13 | 34 | 30 | 2 | 1 |
| dog | 6 | 1 | 22 | 107 | 20 | 793 | 14 | 32 | 0 | 5 |
| frog | 2 | 2 | 16 | 20 | 11 | 5 | 933 | 2 | 5 | 4 |
| horse | 9 | 2 | 8 | 18 | 25 | 26 | 5 | 900 | 2 | 5 |
| ship | 26 | 13 | 2 | 3 | 2 | 1 | 1 | 1 | 934 | 17 |
| truck | 7 | 52 | 0 | 7 | 0 | 1 | 1 | 2 | 9 | 921 |

| | airplane | automobile | bird | cat | deer | dog | frog | horse | ship | truck |
|---|---|---|---|---|---|---|---|---|---|---|
| airplane | 0.89 | 0.01 | 0.02 | 0.01 | 0.01 | 0.00 | 0.01 | 0.01 | 0.03 | 0.01 |
| automobile | 0.01 | 0.94 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.01 | 0.04 |
| bird | 0.03 | 0.01 | 0.81 | 0.04 | 0.04 | 0.02 | 0.05 | 0.01 | 0.01 | 0.00 |
| cat | 0.01 | 0.01 | 0.03 | 0.70 | 0.03 | 0.11 | 0.06 | 0.03 | 0.01 | 0.01 |
| deer | 0.01 | 0.00 | 0.03 | 0.02 | 0.86 | 0.01 | 0.03 | 0.03 | 0.00 | 0.00 |
| dog | 0.01 | 0.00 | 0.02 | 0.11 | 0.02 | 0.79 | 0.01 | 0.03 | 0.00 | 0.01 |
| frog | 0.00 | 0.00 | 0.02 | 0.02 | 0.01 | 0.01 | 0.93 | 0.00 | 0.01 | 0.00 |
| horse | 0.01 | 0.00 | 0.01 | 0.02 | 0.03 | 0.03 | 0.01 | 0.90 | 0.00 | 0.01 |
| ship | 0.03 | 0.01 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.93 | 0.02 |
| truck | 0.01 | 0.05 | 0.00 | 0.01 | 0.00 | 0.00 | 0.00 | 0.00 | 0.01 | 0.92 |

*Figure 10 VGG19 Model Non-Normalized and Normalized Confusion Matrix*

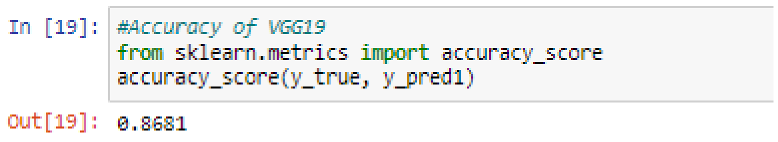
Now, the average accuracy of the model prediction using accuracy_score function in model VGG16 is:



```
In [31]: #Accuracy of VGG19
         from sklearn.metrics import accuracy_score
         accuracy_score(y_true, y_pred1)

Out[31]: 0.8431
```

*Figure 11 VGG16 model prediction accuracy*

And the average accuracy of the model prediction using accuracy_score function in model VGG19 is



```
In [19]: #Accuracy of VGG19
         from sklearn.metrics import accuracy_score
         accuracy_score(y_true, y_pred1)

Out[19]: 0.8681
```

*Figure 12 VGG19 model prediction accuracy*

Hence, the accuracy scores of both models are:

| Model | VGG16 | VGG19 |
|---|---|---|
| Accuracy in % | 84.31 | 86.81 |

Table 5: Comparison of accuracy in VGG Model

This above accuracy results the CNN VGG19 model is better than CNN VGG16 model using training CIFAR-10 dataset for image classification.

# Chapter 4: Conclusion

Finally, we are ready with all the evaluation matrices to analyze the both transfer learning-base deep convolutional neural network models. By analyzing accuracy score and confusion matrices of both the model (Model VGG16 and VGG19). We can conclude that the VGG19 has the best performance than VGG16 among all the above scores are obtained in 100 and 50 epoch of training. It is possible that the score may be improved its train model in more epochs.

## References

[1] T. Adhikari, "Designing a Convolutional Neural Network for Image Recognition: A Comparative Study of Different Architectures and Training Techniques," *Available at SSRN 4366645,* 2023.

[2] V. N. a. G. H. Alex Krizhevsky, "Alex Krizhevsky's," CIFAR-10, [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html.

[3] K. a. Z. A. Simonyan, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556,* 2014.

[4] H. a. V. A. a. F. D. Qassim, "Compressed residual-VGG16 CNN model for big data places image recognition," *IEEE,* pp. 169--175, 2018.

[5] T. V. a. S. P. Janahiraman, "Gender classification based on Asian faces using deep learning," *IEEE,* vol. IEEE 9th, no. 13, Issue 4, April-2022, pp. 84--89, 2019.

[6] T. a. G. T. K. Kaur, "Automated brain image classification based on VGG-16 and transfer learning," *2019 International Conference on Information Technology (ICIT),* no. IEE, pp. 94--98, 2019.

[7] G. a. H. T. Levi, "Age and gender classification using convolutional neural networks," pp. 34--42, 2015.

[8] C. a. P. S. Sharma, "Comparison of CNN and Pre-trained models: A Study," *Comparison of CNN and Pre-trained models: A Study. Available at https://www. researchgate. net/publication/359850786_Comparison_of_CNN_and_Pre-trained_models_A_Study},* 2022.

[9] S. Tammina, "Transfer learning using vgg-16 with deep convolutional neural network for classifying images," *International Journal of Scientific and Research Publications (IJSRP),* vol. 9, no. International Journal of Scientific and Research Publications (IJSRP), pp. 143--150, 2019.

[10] T. a. D. P. a. A. M. F. a. M. M.-F. Ahmed, "A comparative study on convolutional neural network based face recognition," *IEEE,* vol. 1556v6, no. 2020, pp. 1--5, 2015.

[11] D. V. Kumar, "Practical Comparison of Transfer Learning Models in Multi-Class Image Classification," p. 15, 2020.

[12] S. a. R. H. a. S. S. A. A. a. B. M. a. M. G. a. D. S. Khan, A guide to convolutional neural networks for computer vision, Springer, 2018.

[13] D. P. M. U. C. Analytics, "Computing, Communications and Informatics (ICACCI)," *analyticscomputing,* vol. Vol. 9, no. 10, October-2020 , pp. 1-9, 2020.