SYMBIOSIS UNIVERSITY OF APPLIED SCIENCES

INDORE

An INTERNSHIP REPORT

ON

"Comparative analysis of different Convolutional Neural Network algorithm for Image Classification"

Submitted to "Symbiosis University of Applied Sciences, Indore
As an Internship report for the partial fulfillment of the award of degree of

BACHELOR OF ENGINEERING

IN

SCHOOL OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

Submitted to:
Dr. Ashish Bansal

Submitted By:
Name: Harshit Agrawal
Roll No: 2016AB001031

SYMBIOSIS UNIVERSITY OF APPLIED SCIENCES

INDORE

CERTIFICATE

This is to certify that the Internship report entitled "Comparative analysis of different Convolutional Neural Network algorithm for Image Classification", submitted by Mr. Harshit Agrawal , student of fourth year towards partial fulfillment of the degree of Bachelor of Engineering in School of Computer Science and Information Technology in year 2016-2020, Symbiosis University of Applied Sciences , Indore (M.P.) is in partial fulfillment of the requirement for the award of the degree of Bachelor of Engineering and is a bonafide record of the work carried by Harshit, during the academic semester Eighth.

Place: Indore

Date:15-May-2020

INTERNAL EXAMINER EXTERNAL EXAMINER

<u>**SYMBIOSIS UNIVERSITY OF APPLIED SCIENCES**</u>

<u>**INDORE**</u>

RECOMMENDATION

The work entitled "**Comparative analysis of different Convolutional Neural Network Algorithm for Image Classification**", submitted by **Harshit Agrawal**, student of fourth year Computer Science and Information Technology, towards the partial fulfillment for the award of degree of Bachelor of Engineering in Computer Science and Information Technology of Symbiosis University of Applied Sciences Indore(M.P.) is a satisfactory account of their Internship and is recommended for the award of the degree.

Endorsed By:

Dr. Ashish Bansal

Dean, SCSIT

SYMBIOSIS UNIVERSITY OF APPLIED SCIENCES

INDORE

ACKNOWLEDGEMENT

The successful completion of any work is generally not an individual effort. It is an outcome of dedicated and cumulative efforts of a number of people, each having its own importance to the objective. This section is a value of thanks and gratitude towards all those persons who have implicitly or explicitly contributed in their own unique way towards the completion of the project. For their invaluable comments and suggestions, I wish to thank them all.

Positive inspiration and right guidance are must in every aspect of life. Especially, when we arrive at academic stage for instance. For the success of our project a number of obligations have been taken. We have performed solemn duty of expressing a heartfelt thanks to all who have endowed us with their precious perpetual guidance, suggestions and information. Any kind of help directly or indirectly has proved importance to us.

Harshit

CONTENTS

Chapter 1: INTRODUCTION

1.1 Objective

1.2 Background

Chapter 2: THE PROJECT

2.1 Project definition

2.2 Project scope

Chapter 3: REQUIREMENTS ANALYSIS

3.1 Functional Requirements

3.2 System Requirements

Chapter 4: RESEARCH

4.1: Artificial Intelligence

4.2: Machine Learning

4.3: Deep Learning

4.4: Neural Networks

4.4.1 Theory

4.4.2 Datasets Used

4.4.3 Types of Network used (LeNET, AlexNET, VGGNET16, ResNET50, GoogLeNET)

4.4.4 Code and Output

Chapter 5: CONCLUSION

5.1 Result

5.2 Learning Outcome

BIBLIOGRAPHY AND REFERENCE

1.Introduction:

Objective:

This Internship is a research based training where he overall objective is to do a comparative study of all the famous Convolutional Neural network which are used for image classification purposes and to increase their efficaciousness so as to obtain a final conclusion on what type of network should be used for different circumstances and on different types of contextual datasets.

This process includes learning functioning of Convolutional Neural network and all the other important technologies pertaining to Artificial Intelligence and Neural networking. All the work showcased in this research and the result obtained is achieved after extensive training of different models of Convolutional neural networks on millions of datasets using high end Workstations. The results may vary according to the choice of datasets and variation in the algorithms. All the algorithms developed and used are included within the report.

Also, as to gain the knowledge of Neural network and working of an Artificial Intelligence model, going through various certifications and courses was a must. All the courses and certifications undertaken are also included in this report.

Background:

The research project is done under the guidance of Dr. Ashish Bansal (Director and Professor, Symbiosis University of Applied Sciences). His Adeptness is Artificial Intelligence and copious knowledge in the domain of problem solving and programming became the north star throughout the tenure of my internship.

2. The Project:

The basic motive of this project is to get in-depth knowledge of Artificial Intelligence and all the important domains pertaining to it such as Machine learning, Deep learning and Convolutional Neural Networks.

The ultimate objective is to prepare a descriptive and data rich report on the working of different Convolutional Neural Network models and algorithm so that it can be used as a reference in order to perform operations on a specific type of dataset in a given real world problem.

Convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics. They have applications in image and video recognition, recommender systems, image classification, medical image analysis, natural language processing, and financial time series

The Processing of Convolutional neural network demands an extensive Tensor Processing Unit (TPU) and Graphics Processing Unit (GPU) system and takes hours to train which makes it intricate for analyzing properly for each and every permutation and combination of problem and dataset. This report is oriented to ease these kind of difficulties so as to make the process of analysis easier for many developers in the Artificial Intelligence domain.

Scope:

The research done and the result obtained by this research is applicable in the whole domain of artificial intelligence where CNN is used as a base to bolster the working of any system in an organization. Moreover, this research will assist budding developers and researchers in the proper creation of an Artificial Intelligence enabled model.

3. Requirement Analysis:

- Knowledge of a programming Language (python/java etc.)
- Familiarity with different operating systems
- Ability to stretch and skew images
- Basic working proficiency in Artificial Intelligence

System Requirements:

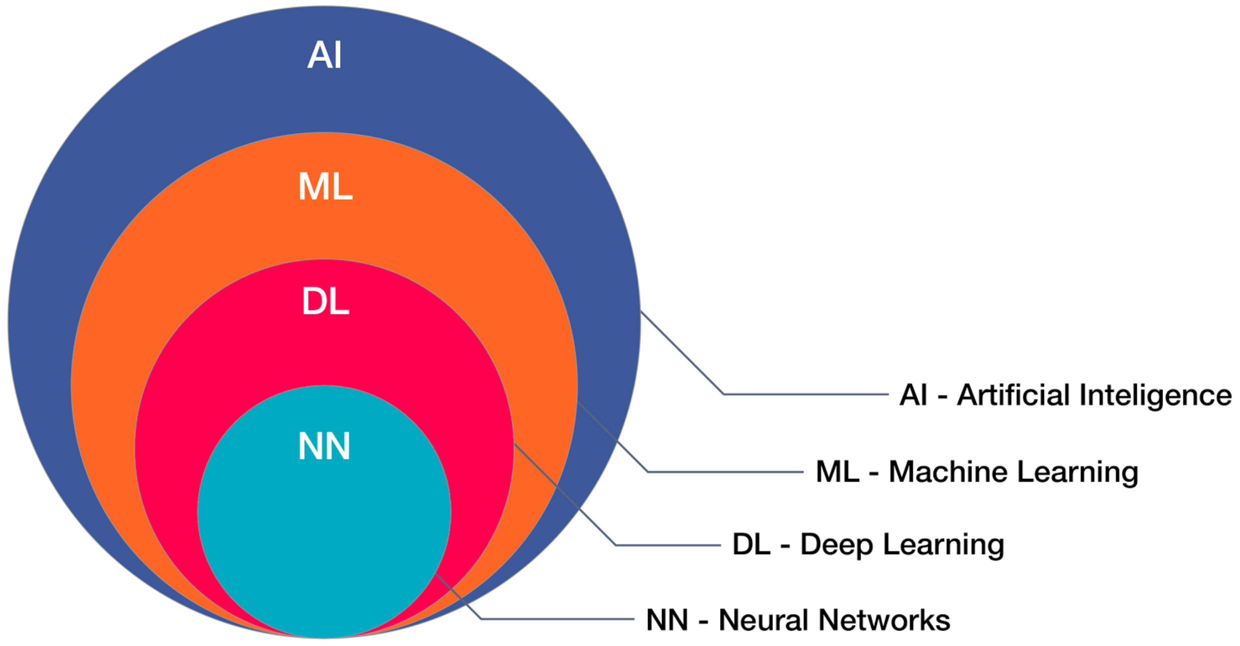
Minimum: -

- Processor: Intel i3 or AMD x4
- Memory: 8gb
- GPU: NVIDIA 1050ti (AMD is not fully compatible)
- Operating System: Windows 7 +, ubuntu, macOS

Recommended: -

- Processor: Intel i7 or AMD Ryzen 7
- Memory: 16gb DDR4
- GPU: NVIDIA RTX 2080 (AMD is not fully compatible)
- Operating System: Ubuntu

Research Theory:



Artificial Intelligence:

Artificial intelligence (AI) is the simulation of human intelligence processes by machines, especially computer systems. These processes include learning (the acquisition of information and rules for using the information), reasoning (using rules to reach approximate or definite conclusions) and self-correction.

AI can be categorized as either weak or strong. Weak AI, also known as narrow AI, is an AI system that is designed and trained for a particular task. Virtual personal assistants, such as Apple's Siri, are a form of weak AI. Strong AI, also known as artificial general intelligence, is an AI system with generalized human cognitive abilities. When presented with an unfamiliar task, a strong AI system is able to find a solution without human intervention.

AI as a Service allows individuals and companies to experiment with AI for various business purposes and sample multiple platforms before making a commitment. Popular AI cloud offerings include Amazon AI services, IBM Watson Assistant, Microsoft Cognitive Services and Google AI services.

Types of artificial intelligence:

Arend Hintze, an assistant professor of integrative biology and computer science and engineering at Michigan State University, categorizes AI into four types, from the kind of AI systems that exist today to sentient systems, which do not yet exist. His categories are as follows:

Type 1: Reactive machines. An example is Deep Blue, the IBM chess program that beat Garry Kasparov in the 1990s. Deep Blue can identify pieces on the chess board and make predictions, but it has no memory and cannot use past experiences to inform future ones. It analyzes possible moves -- its own and its opponent -- and chooses the most strategic move. Deep Blue and Google's AlphaGO were designed for narrow purposes and cannot easily be applied to another situation.

Type 2: Limited memory. These AI systems can use past experiences to inform future decisions. Some of the decision-making functions in self-driving cars are designed this way. Observations inform actions happening in the not-so-distant

future, such as a car changing lanes. These observations are not stored permanently.

Type 3: Theory of mind. This psychology term refers to the understanding that others have their own beliefs, desires and intentions that impact the decisions they make. This kind of AI does not yet exist.

Type 4: Self-awareness. In this category, AI systems have a sense of self, have consciousness. Machines with self-awareness understand their current state and can use the information to infer what others are feeling. This type of AI does not yet exist.

Machine Learning:

• Arthur Samuel (1959). Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed.

• Tom Mitchell (1998) Well-posed Learning Problem: A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.

Some machine learning methods:

Supervised machine learning algorithms can apply what has been learned in the past to new data using labeled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.

Unsupervised machine learning algorithms are used when the information used to train is neither classified nor labeled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabeled data. The system doesn’t figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data.

Semi-supervised machine learning algorithms fall somewhere in between supervised and unsupervised learning, since they use both labeled and unlabeled data for training – typically a small amount of labeled data and a large amount of unlabeled data. The systems that use this method are able to considerably improve learning accuracy. Usually, semi-supervised learning is chosen when the acquired labeled data requires skilled and relevant resources in order to train it / learn from it. Otherwise, acquiring unlabeled data generally doesn't require additional resources.

Reinforcement machine learning algorithms is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Trial and error search and delayed reward are the most relevant characteristics of reinforcement learning. This method allows machines and software agents to automatically determine the ideal behavior within a specific context in order to maximize its performance. Simple reward feedback is required for the agent to learn which action is best; this is known as the reinforcement signal.

Machine learning enables analysis of massive quantities of data. While it generally delivers faster, more accurate results in order to identify profitable opportunities or dangerous risks, it may also require additional time and resources to train it properly. Combining machine learning with AI and cognitive technologies can make it even more effective in processing large volumes of information.

Deep Learning:

Deep learning is an artificial intelligence function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence (AI) that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network.

Deep learning has evolved hand-in-hand with the digital era, which has brought about an explosion of data in all forms and from every region of the world. This data, known simply as big data, is drawn from sources like social media, internet search engines, e-commerce platforms, and online cinemas, among others. This enormous amount of data is readily accessible and can be shared through fintech applications like cloud computing. However, the data, which normally is unstructured, is so vast that it could take decades for humans to comprehend it and extract relevant information. Companies realize the incredible potential that can result from unraveling this wealth of information and are increasingly adapting to AI systems for automated support.

Neural Network:

A neural network is a network or circuit of neurons, or in a modern sense, an artificial neural network, composed of artificial neurons or nodes. Thus a neural network is either a biological neural network, made up of real biological neurons, or an artificial neural network, for solving artificial intelligence (AI) problems. The connections of the biological neuron are modeled as weights. A positive weight reflects an excitatory connection, while negative values mean inhibitory connections. All inputs are modified by a weight and summed. This activity is referred to as a linear combination. Finally, an activation function controls the amplitude of the output. For example, an acceptable range of output is usually between 0 and 1, or it could be −1 and 1.

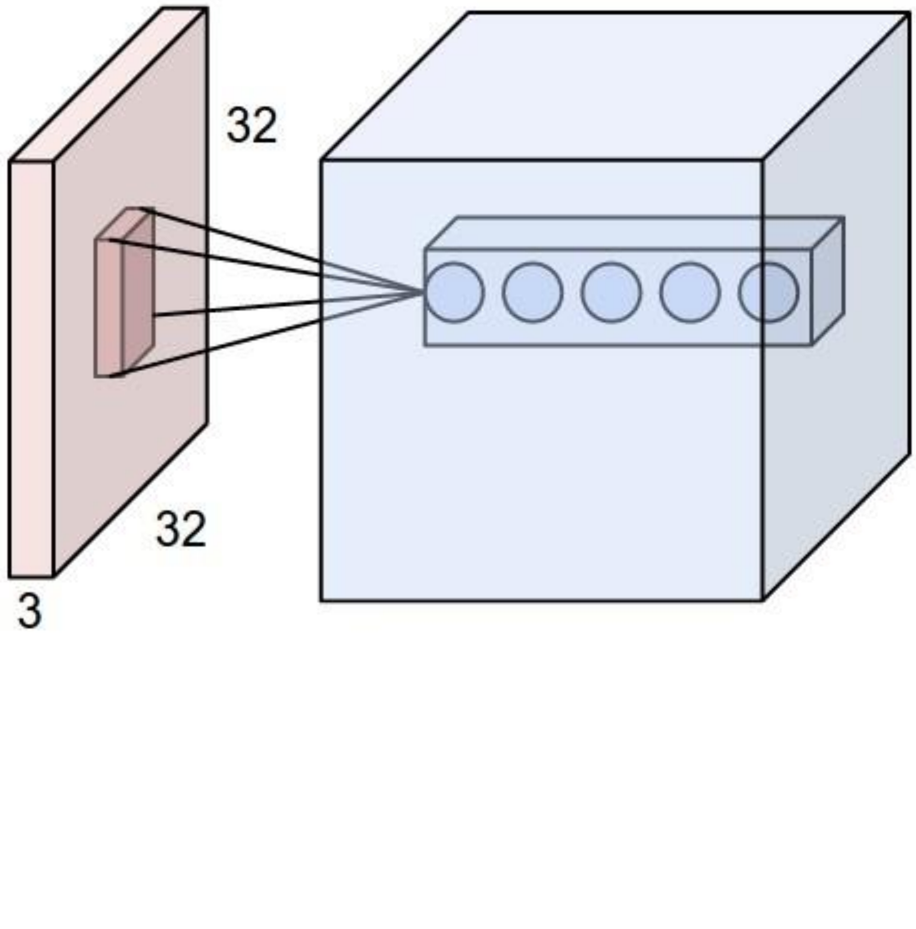
These artificial networks may be used for predictive modeling, adaptive control and applications where they can be trained via a dataset. Self-learning resulting from experience can occur within networks, which can derive conclusions from a complex and seemingly unrelated set of information.

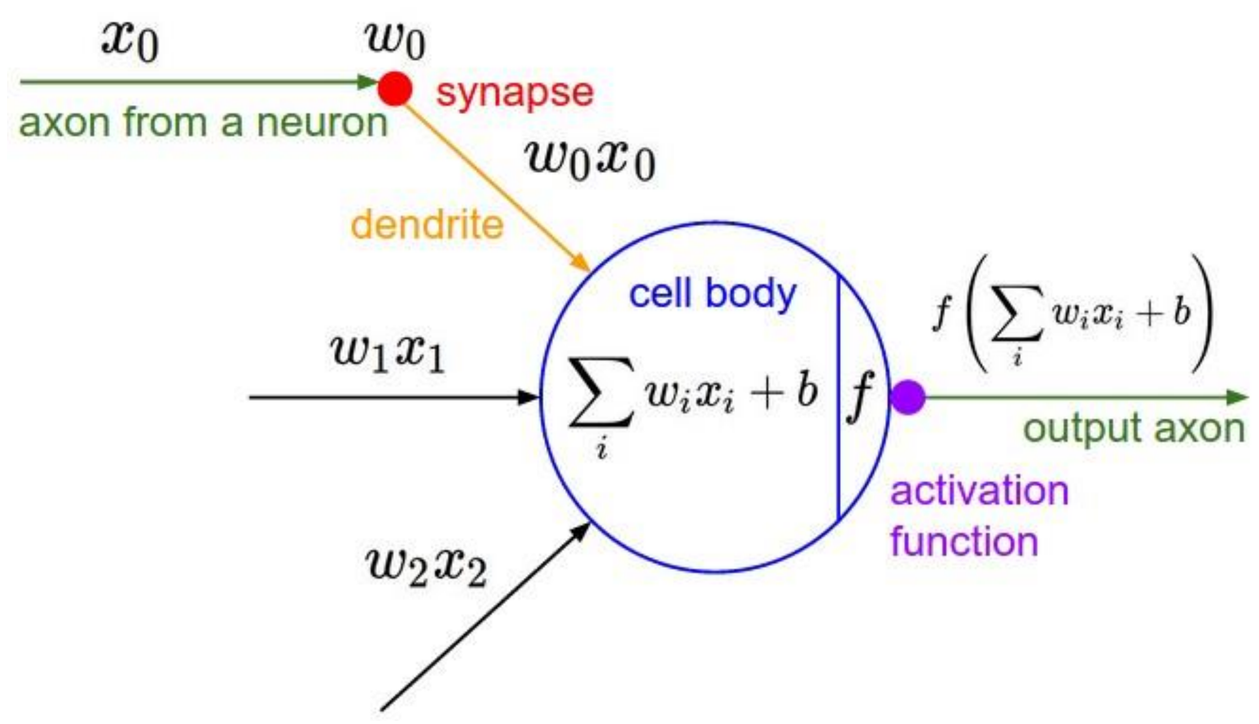
Types of Neural Network:

1. Feedforward Neural Network – Artificial Neuron
2. Radial basis function Neural Network
3. Kohonen Self Organizing Neural Network
4. Recurrent Neural Network(RNN) – Long Short Term Memory
5. Convolutional Neural Network
6. Modular Neural Network

In this project as it is based on Image Processing Convolutional Neural Network is used as it serves the purpose effectively.

Convolutional Neural Network:





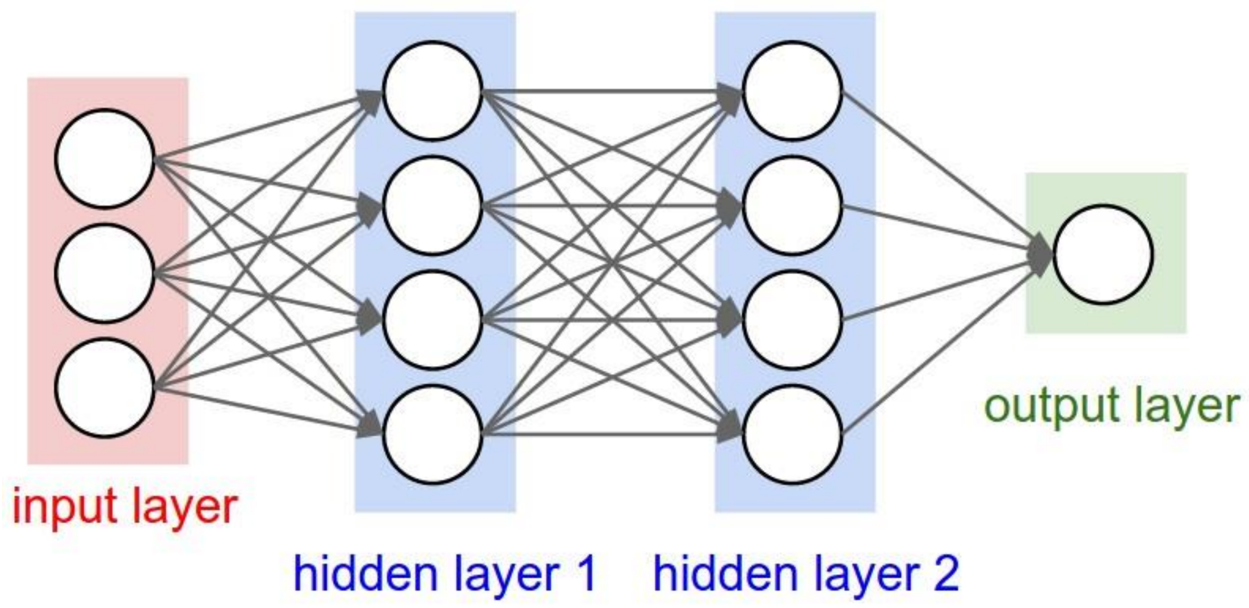
A convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics. They have applications in image and video recognition, recommender systems, image classification, medical image analysis, natural language processing, and financial time series.

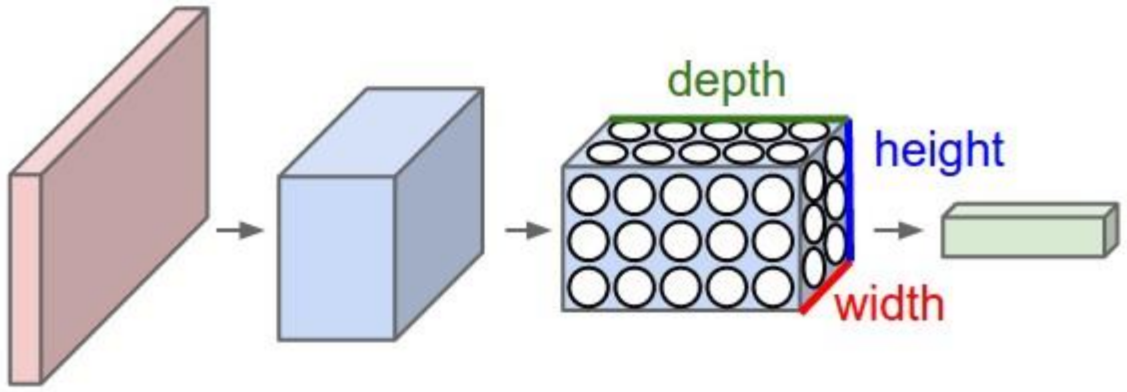
Architecture:

Neural Networks receive an input (a single vector), and transform it through a series of hidden layers. Each hidden layer is made up of a set of neurons, where each neuron is fully connected to all neurons in the previous layer, and where neurons in a single layer function completely independently and do not share any connections. The last fully-connected layer is called the “output layer” and in classification settings it represents the class scores.

Regular Neural Nets don’t scale well to full images. In CIFAR-10, images are only of size 32x32x3 (32 wide, 32 high, 3 color channels), so a single fully-connected neuron in a first hidden layer of a regular Neural Network would have 32*32*3 = 3072 weights. This amount still seems manageable, but clearly this fully-connected structure does not scale to larger images. For example, an image of more respectable size, e.g. 200x200x3, would lead to neurons that have 200*200*3 = 120,000 weights. Moreover, we would almost certainly want to have several such neurons, so the parameters would add up quickly! Clearly, this full connectivity is wasteful and the huge number of parameters would quickly lead to overfitting.

3D volumes of neurons. Convolutional Neural Networks take advantage of the fact that the input consists of images and they constrain the architecture in a more sensible way. In particular, unlike a regular Neural Network, the layers of a ConvNet have neurons arranged in 3 dimensions: width, height, depth. For example, the input images in CIFAR-10 are an input volume of activations, and the volume has dimensions 32x32x3 (width, height, depth respectively). The neurons in a layer will only be connected to a small region of the layer before it, instead of all of the neurons in a fully-connected manner. Moreover, the final output layer would for CIFAR-10 have dimensions 1x1x10, because by the end of the ConvNet architecture we will reduce the full image into a single vector of class scores, arranged along the depth dimension.





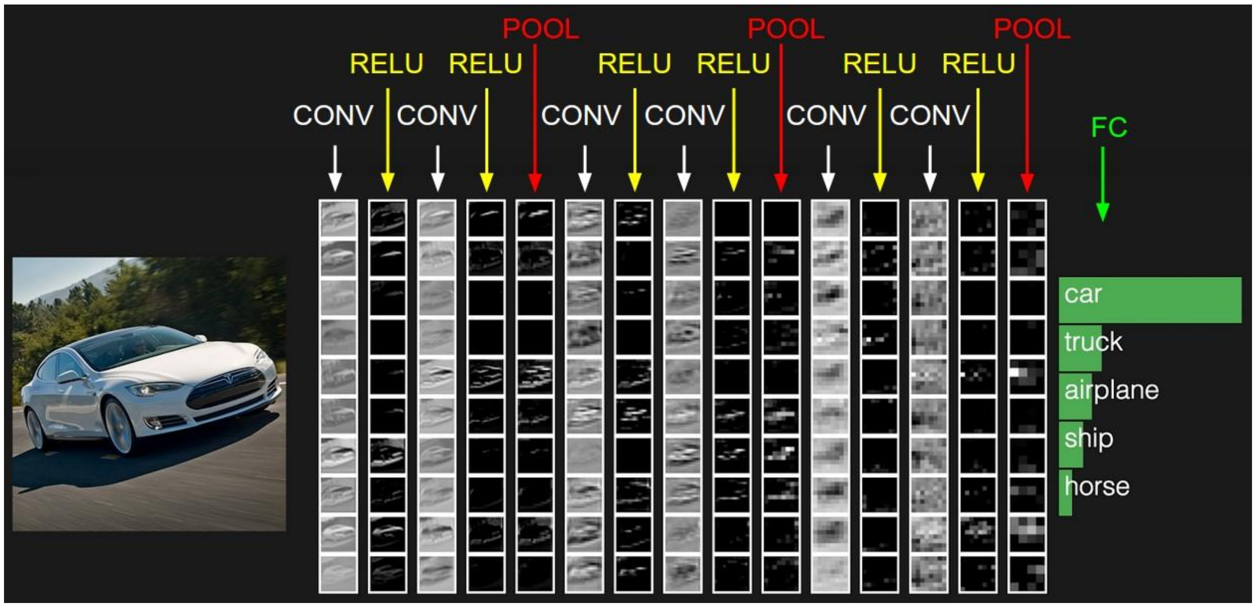
Layers used to build ConvNets:

A simple ConvNet is a sequence of layers, and every layer of a ConvNet transforms one volume of activations to another through a differentiable function. It uses three main types of layers to build ConvNet architectures: Convolutional Layer, Pooling Layer, and Fully-Connected Layer (exactly as seen in regular Neural Networks).

In more detail:

- INPUT [32x32x3] will hold the raw pixel values of the image, in this case an image of width 32, height 32, and with three color channels R, G, B.

- CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. This may result in volume such as [32x32x12] if we decided to use 12 filters.
- RELU layer will apply an elementwise activation function, such as the max(0,x) thresholding at zero. This leaves the size of the volume unchanged ([32x32x12]).
- POOL layer will perform a downsampling operation along the spatial dimensions (width, height), resulting in volume such as [16x16x12].
- FC (i.e. fully-connected) layer will compute the class scores, resulting in volume of size [1x1x10], where each of the 10 numbers correspond to a class score, such as among the 10 categories of CIFAR-10. As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume.



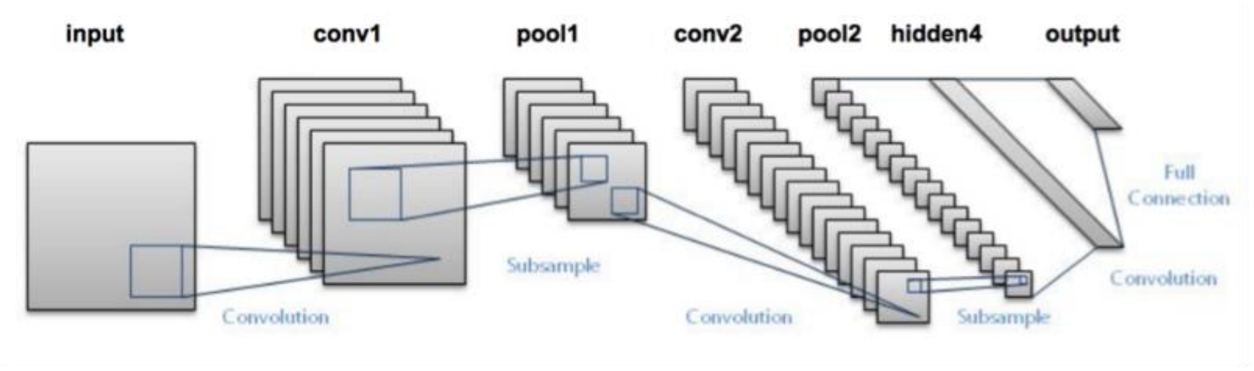
Test Datasets

Image dataset of CIFAR- 100 which has numerous super-classes of general object images and a number of subclass categories of each superclass. CIFAR-100 has 100 classes of images with each class having 600 images each. These 600 images are divided into 500 training images and 100 testing images for each class, therefore, making a total of 60,000 different images. These 100 classes are clubbed together into 20 super classes. Every image in the dataset comes with a “fine” label (depicting the class to which it belongs) and a “coarse” label (superclass to the “fine” label detected). The selected categories for training and testing are abed, bicycle, bus, chair, couch, motorcycle, streetcar, table, train, and wardrobe. For the proposed work, some wide categories of each super classes need to be used for training the networks, the super classes used are Household furniture and vehicle.

Convolutional Neural Networks implemented in this research:

- LeNet
- AlexNet
- VGGNet16
- ResNet50
- GoogLeNet

1.LeNet:



LeNet-5, a pioneering 7-level convolutional network by LeCun et al in 1998, that classifies digits, was applied by several banks to recognize hand-written numbers on checks (cheques) digitized in 32x32 pixel greyscale input images. The ability to process higher resolution images requires larger and more convolutional layers, so this technique is constrained by the availability of computing resources.

The LeNet-5 architecture consists of two sets of convolutional and average pooling layers, followed by a flattening convolutional layer, then two fully-connected layers and finally a softmax classifier.

The input for LeNet-5 is a 32×32 grayscale image which passes through the first convolutional layer with 6 feature maps or filters having size 5×5 and a stride of one. The image dimensions changes from 32x32x1 to 28x28x6.

Then the LeNet-5 applies average pooling layer or sub-sampling layer with a filter size 2×2 and a stride of two. The resulting image dimensions will be reduced to 14x14x6.

Next, there is a second convolutional layer with 16 feature maps having size 5×5 and a stride of 1. In this layer, only 10 out of 16 feature maps are connected to 6 feature maps of the previous layer.

The main reason is to break the symmetry in the network and keeps the number of connections within reasonable bounds. That’s why the number of training parameters in these layers are 1516 instead of 2400 and similarly, the number of connections is 151600 instead of 240000.

The fourth layer is again an average pooling layer with filter size 2×2 and a stride of 2. This layer is the same as the second layer except it has 16 feature maps so the output will be reduced to 5x5x16.

The fifth layer is a fully connected convolutional layer with 120 feature maps each of size 1×1. Each of the 120 units is connected to all the 400 nodes (5x5x16) in the fourth layer.

The sixth layer is a fully connected layer with 84 units.

Finally, there is a fully connected softmax output layer ŷ with 10 possible values corresponding to the digits from 0 to 9.

//Code:

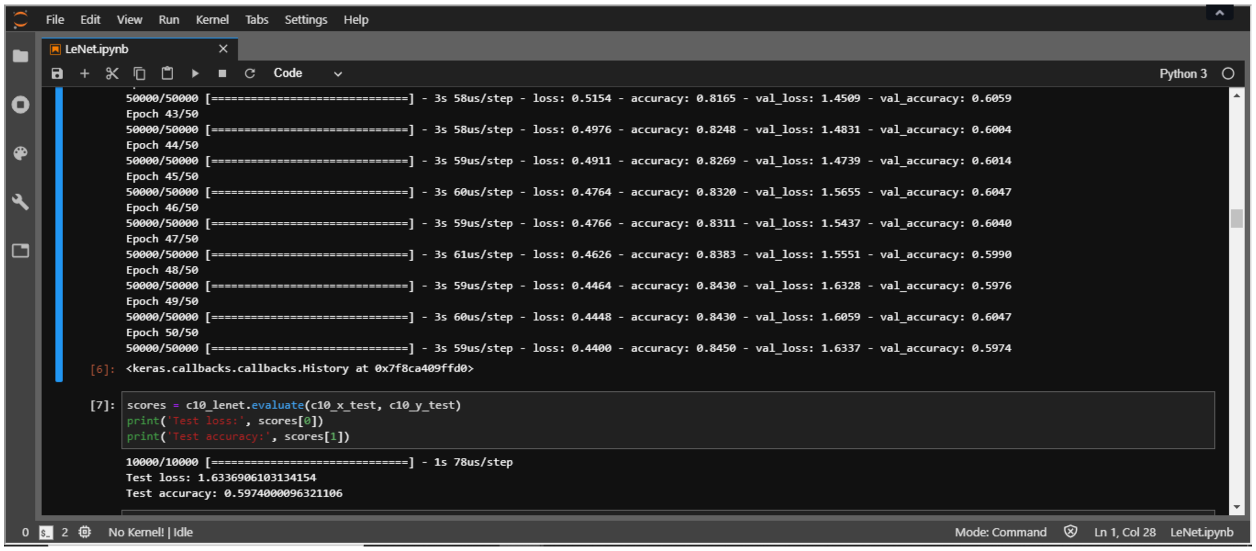
```
def Lenet(input_shape, num_classes):
    lenet = Sequential()
    lenet.add(Conv2D(6, (5, 5), padding='valid', activation = 'relu', kernel_initializer='he_normal', inpu
t_shape=input_shape))
    lenet.add(AveragePooling2D((2, 2), strides=(2, 2)))
    lenet.add(Conv2D(16, (5, 5), padding='valid', activation = 'relu', kernel_initializer='he_normal'))
    lenet.add(AveragePooling2D((2, 2), strides=(2, 2)))
    lenet.add(Flatten())
    lenet.add(Dense(120, activation = 'relu', kernel_initializer='he_normal'))
    lenet.add(Dense(84, activation = 'relu', kernel_initializer='he_normal'))
    lenet.add(Dense(num_classes, activation = 'softmax', kernel_initializer='he_normal'))

    lenet.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    return lenet
```

Ciphar10:

```
# Load data
(c10_x_train_orig, c10_y_train_orig), (c10_x_test_orig, c10_y_test_orig) = cifar10.load_data()

# Pre-processing X
c10_x_train = c10_x_train_orig.astype('float32')
c10_x_test = c10_x_test_orig.astype('float32')
c10_x_train = c10_x_train/255
c10_x_test = c10_x_test/255

# Hyper-parameters
c10_epochs = 50
c10_batch_size = 128
c10_num_classes = 10
c10_input_shape = c10_x_train.shape[1:]

# Pre-processing
c10_y_train = keras.utils.to_categorical(c10_y_train_orig, c10_num_classes)
c10_y_test = keras.utils.to_categorical(c10_y_test_orig, c10_num_classes)

print(c10_x_train.shape, c10_x_test.shape, c10_y_train.shape, c10_y_test.shape)

c10_lenet = Lenet(c10_input_shape, c10_num_classes)
c10_lenet.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 6)         456       
_________________________________________________________________
average_pooling2d_1 (Average (None, 14, 14, 6)         0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 10, 10, 16)        2416      
_________________________________________________________________
average_pooling2d_2 (Average (None, 5, 5, 16)          0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 400)               0         
_________________________________________________________________
dense_1 (Dense)              (None, 120)               48120     
_________________________________________________________________
dense_2 (Dense)              (None, 84)                10164     
_________________________________________________________________
dense_3 (Dense)              (None, 10)                850       
=================================================================
Total params: 62,006
Trainable params: 62,006
Non-trainable params: 0
```

OUTPUT:



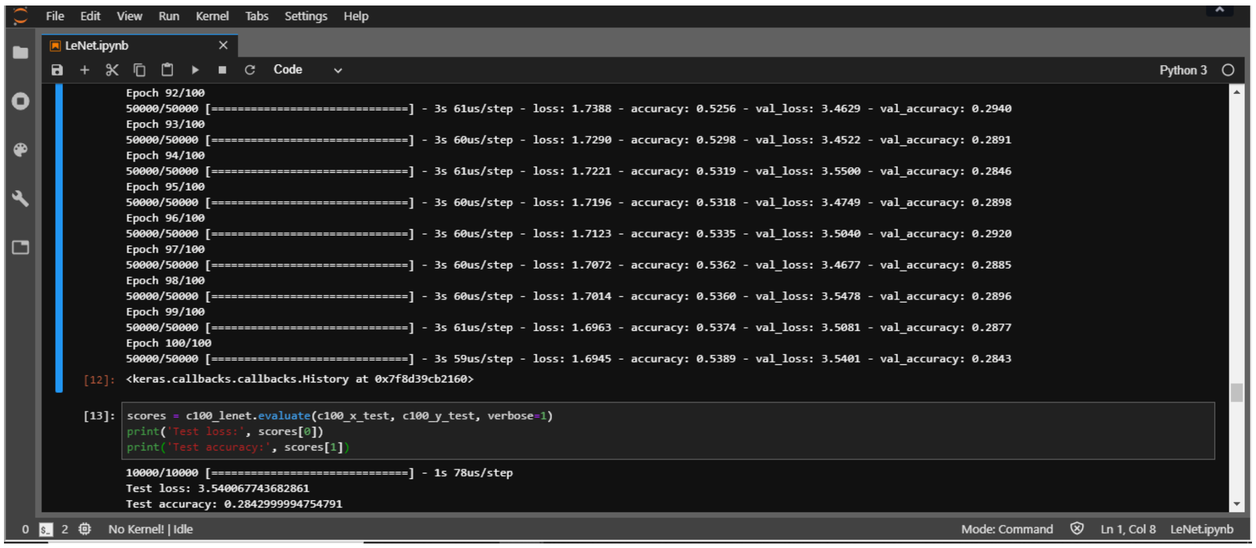
```
50000/50000 [==============================] - 3s 58us/step - loss: 0.5154 - accuracy: 0.8165 - val_loss: 1.4509 - val_accuracy: 0.6059
Epoch 43/50
50000/50000 [==============================] - 3s 58us/step - loss: 0.4976 - accuracy: 0.8248 - val_loss: 1.4831 - val_accuracy: 0.6004
Epoch 44/50
50000/50000 [==============================] - 3s 59us/step - loss: 0.4911 - accuracy: 0.8269 - val_loss: 1.4739 - val_accuracy: 0.6014
Epoch 45/50
50000/50000 [==============================] - 3s 60us/step - loss: 0.4764 - accuracy: 0.8320 - val_loss: 1.5655 - val_accuracy: 0.6047
Epoch 46/50
50000/50000 [==============================] - 3s 59us/step - loss: 0.4766 - accuracy: 0.8311 - val_loss: 1.5437 - val_accuracy: 0.6040
Epoch 47/50
50000/50000 [==============================] - 3s 61us/step - loss: 0.4626 - accuracy: 0.8383 - val_loss: 1.5551 - val_accuracy: 0.5990
Epoch 48/50
50000/50000 [==============================] - 3s 59us/step - loss: 0.4464 - accuracy: 0.8430 - val_loss: 1.6328 - val_accuracy: 0.5976
Epoch 49/50
50000/50000 [==============================] - 3s 60us/step - loss: 0.4448 - accuracy: 0.8430 - val_loss: 1.6059 - val_accuracy: 0.6047
Epoch 50/50
50000/50000 [==============================] - 3s 59us/step - loss: 0.4400 - accuracy: 0.8450 - val_loss: 1.6337 - val_accuracy: 0.5974
[6]: <keras.callbacks.callbacks.History at 0x7f8ca409ffd0>

[7]: scores = c10_lenet.evaluate(c10_x_test, c10_y_test)
     print('Test loss:', scores[0])
     print('Test accuracy:', scores[1])

10000/10000 [==============================] - 1s 78us/step
Test loss: 1.6336906103134154
Test accuracy: 0.5974000096321106
```

Cipha100:

```
# Load data
(c100_x_train_orig, c100_y_train_orig), (c100_x_test_orig, c100_y_test_orig) = cifar100.load_data()

# Pre-processing X
c100_x_train = c100_x_train_orig.astype('float32')
c100_x_test = c100_x_test_orig.astype('float32')
c100_x_train = c100_x_train/255
c100_x_test = c100_x_test/255

# Hyper-parameters
c100_epochs = 100
c100_batch_size = 128
c100_num_classes = 100
c100_input_shape = c100_x_train.shape[1:]

# Pre-processing Y
c100_y_train = keras.utils.to_categorical(c100_y_train_orig, c100_num_classes)
c100_y_test = keras.utils.to_categorical(c100_y_test_orig, c100_num_classes)

print(c100_x_train.shape, c100_x_test.shape, c100_y_train.shape, c100_y_test.shape)


c100_lenet = Lenet(c100_input_shape, c100_num_classes)
c100_lenet.summary()
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
```

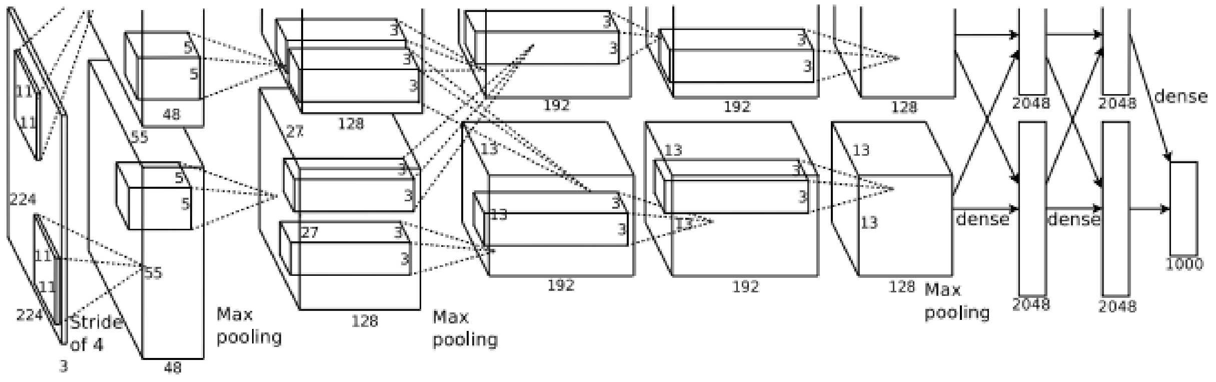
```
=================================================================
conv2d_5 (Conv2D)            (None, 28, 28, 6)         456
_________________________________________________________________
average_pooling2d_5 (Average (None, 14, 14, 6)         0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 10, 10, 16)        2416
_________________________________________________________________
average_pooling2d_6 (Average (None, 5, 5, 16)          0
_________________________________________________________________
flatten_3 (Flatten)          (None, 400)               0
_________________________________________________________________
dense_7 (Dense)              (None, 120)               48120
_________________________________________________________________
dense_8 (Dense)              (None, 84)                10164
_________________________________________________________________
dense_9 (Dense)              (None, 100)               8500
=================================================================
Total params: 69,656
Trainable params: 69,656
Non-trainable params: 0
```

Output:



```
Epoch 92/100
50000/50000 [==============================] - 3s 61us/step - loss: 1.7388 - accuracy: 0.5256 - val_loss: 3.4629 - val_accuracy: 0.2940
Epoch 93/100
50000/50000 [==============================] - 3s 60us/step - loss: 1.7290 - accuracy: 0.5298 - val_loss: 3.4522 - val_accuracy: 0.2891
Epoch 94/100
50000/50000 [==============================] - 3s 61us/step - loss: 1.7221 - accuracy: 0.5319 - val_loss: 3.5500 - val_accuracy: 0.2846
Epoch 95/100
50000/50000 [==============================] - 3s 60us/step - loss: 1.7196 - accuracy: 0.5318 - val_loss: 3.4749 - val_accuracy: 0.2898
Epoch 96/100
50000/50000 [==============================] - 3s 60us/step - loss: 1.7123 - accuracy: 0.5335 - val_loss: 3.5040 - val_accuracy: 0.2920
Epoch 97/100
50000/50000 [==============================] - 3s 60us/step - loss: 1.7072 - accuracy: 0.5362 - val_loss: 3.4677 - val_accuracy: 0.2885
Epoch 98/100
50000/50000 [==============================] - 3s 60us/step - loss: 1.7014 - accuracy: 0.5360 - val_loss: 3.5478 - val_accuracy: 0.2896
Epoch 99/100
50000/50000 [==============================] - 3s 61us/step - loss: 1.6963 - accuracy: 0.5374 - val_loss: 3.5081 - val_accuracy: 0.2877
Epoch 100/100
50000/50000 [==============================] - 3s 59us/step - loss: 1.6945 - accuracy: 0.5389 - val_loss: 3.5401 - val_accuracy: 0.2843
[12]: <keras.callbacks.callbacks.History at 0x7f8d39cb2160>

[13]: scores = c100_lenet.evaluate(c100_x_test, c100_y_test, verbose=1)
      print('Test loss:', scores[0])
      print('Test accuracy:', scores[1])

10000/10000 [==============================] - 1s 78us/step
Test loss: 3.540067743682861
Test accuracy: 0.2842999994754791
```

2.AlexNet:



AlexNet consists of 5 Convolutional Layers and 3 Fully Connected Layers.

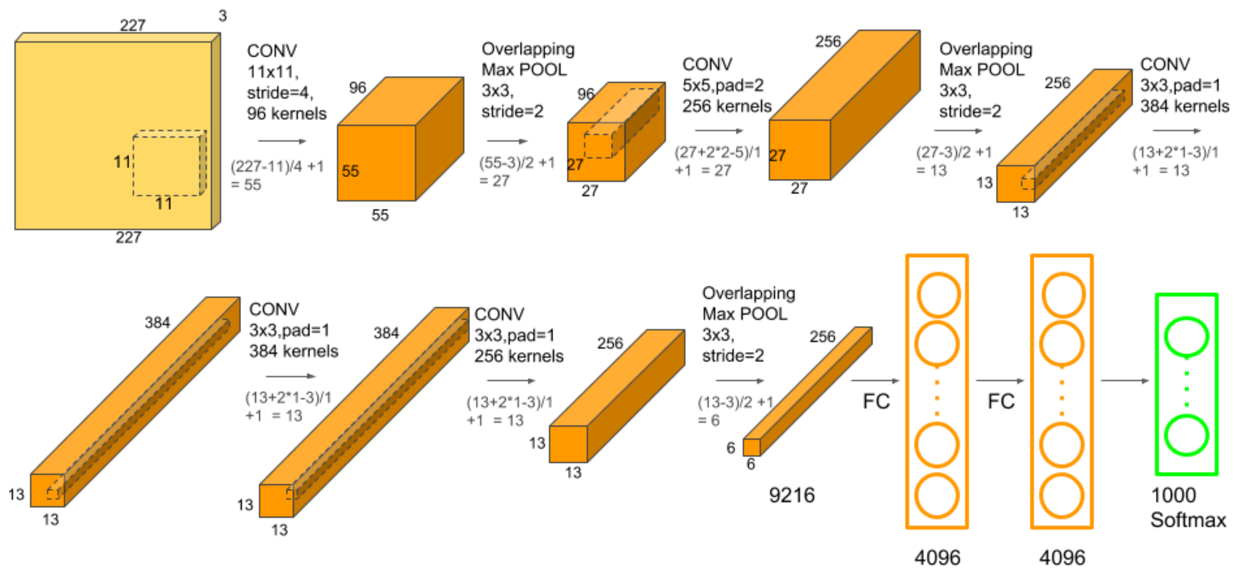
Multiple Convolutional Kernels (a.k.a filters) extract interesting features in an image. In a single convolutional layer, there are usually many kernels of the same size. For example, the first Conv Layer of AlexNet contains 96 kernels of size 11x11x3. Note the width and height of the kernel are usually the same and the depth is the same as the number of channels.

The first two Convolutional layers are followed by the Overlapping Max Pooling layers that we describe next. The third, fourth and fifth convolutional layers are connected directly. The fifth convolutional layer is followed by an Overlapping Max Pooling layer, the output of which goes into a series of two fully connected layers. The second fully connected layer feeds into a softmax classifier with 1000 class labels.

ReLU nonlinearity is applied after all the convolution and fully connected layers. The ReLU nonlinearity of the first and second convolution layers are followed by a local normalization step before doing pooling.

Max Pooling layers are usually used to down sample the width and height of the tensors, keeping the depth same. Overlapping Max Pool layers are similar to the Max Pool layers, except the adjacent windows over which the max is computed overlap each other. The authors used pooling windows of size 3×3 with a stride of 2 between the adjacent windows. This overlapping nature of pooling helped reduce the top-1 error rate by 0.4% and top-5 error rate by 0.3% respectively when compared to using non-overlapping pooling windows of size 2×2 with a stride of 2 that would give same output dimensions.

An important feature of the AlexNet is the use of ReLU(Rectified Linear Unit) Nonlinearity. Tanh or sigmoid activation functions used to be the usual way to train a neural network model. AlexNet showed that using ReLU nonlinearity, deep CNNs could be trained much faster than using the saturating activation functions like tanh or sigmoid. The figure below from the paper shows that using ReLUs(solid curve), AlexNet could achieve a 25% training error rate six times faster than an equivalent network using tanh(dotted curve). This was tested on the CIFAR-10 dataset.



//Code

```
def Alexnet(input_shape, num_classes):
    alexnet = Sequential()
    alexnet.add(Conv2D(filters=96,kernel_size=(3,3),strides=(4,4),input_shape=input_shape, activation='rel
u'))
    alexnet.add(BatchNormalization())
    alexnet.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
    alexnet.add(Conv2D(256,(5,5),padding='same',activation='relu'))
    alexnet.add(BatchNormalization())
    alexnet.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
    alexnet.add(Conv2D(384,(3,3),padding='same',activation='relu'))
    alexnet.add(Conv2D(384,(3,3),padding='same',activation='relu'))
    alexnet.add(Conv2D(256,(3,3),padding='same',activation='relu'))
    alexnet.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
    alexnet.add(Flatten())
    alexnet.add(Dense(4096, activation='relu'))
    alexnet.add(Dropout(0.4))
    alexnet.add(Dense(4096, activation='relu'))
    alexnet.add(Dropout(0.4))
    alexnet.add(Dense(num_classes,activation='softmax'))

    opt = keras.optimizers.Adam(learning_rate=0.0001)
    alexnet.compile(optimizer=opt,loss='categorical_crossentropy',metrics=['accuracy'])
    return alexnet
```

Ciphar10:

```
# Load data
(c10_x_train_orig, c10_y_train_orig), (c10_x_test_orig, c10_y_test_orig) = cifar10.load_data()

# Pre-processing X
c10_x_train = c10_x_train_orig.astype('float32')
c10_x_test = c10_x_test_orig.astype('float32')
c10_x_train = c10_x_train/255
c10_x_test = c10_x_test/255

# Hyper-parameters
c10_epochs = 25
c10_batch_size = 128
c10_num_classes = 10
c10_input_shape = c10_x_train.shape[1:]

# Pre-processing Y
c10_y_train = keras.utils.to_categorical(c10_y_train_orig, c10_num_classes)
```

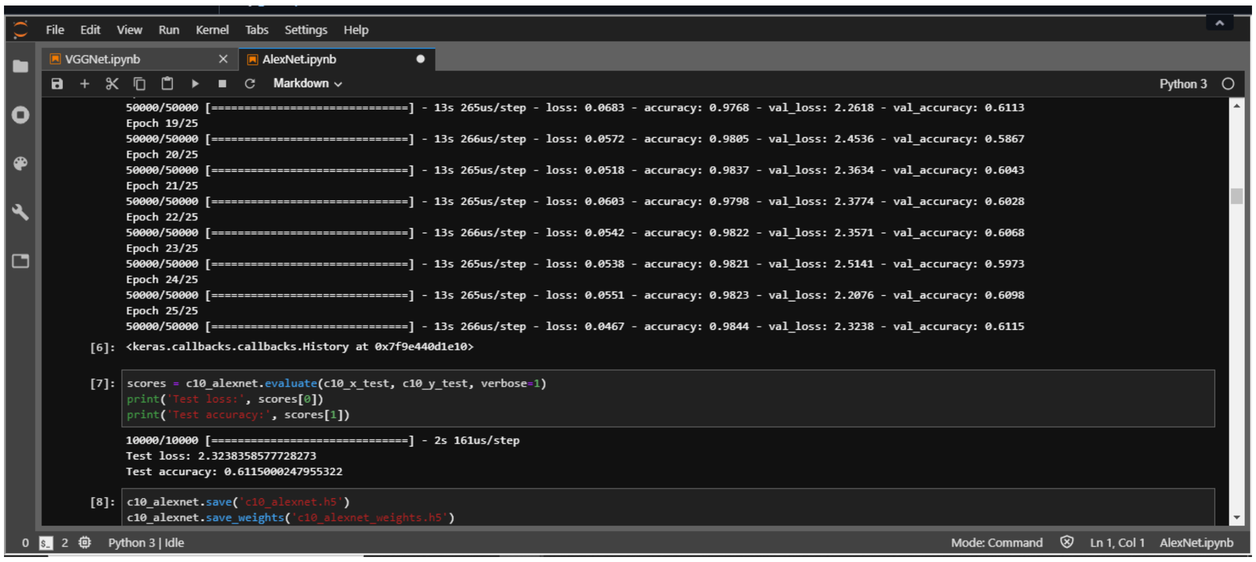
```
c10_y_test = keras.utils.to_categorical(c10_y_test_orig, c10_num_classes)
```

```
print(c10_x_train.shape, c10_x_test.shape, c10_y_train.shape, c10_y_test.shape)
(50000, 32, 32, 3) (10000, 32, 32, 3) (50000, 10) (10000, 10)
```

```
c10_alexnet = Alexnet(c10_input_shape, c10_num_classes)
c10_alexnet.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 8, 8, 96)          2688
_________________________________________________________________
batch_normalization_1 (Batch (None, 8, 8, 96)          384
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 4, 4, 96)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 4, 4, 256)         614656
_________________________________________________________________
batch_normalization_2 (Batch (None, 4, 4, 256)         1024
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 2, 2, 256)         0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 2, 2, 384)         885120
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 2, 2, 384)         1327488
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 2, 2, 256)         884992
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 1, 1, 256)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 4096)              1052672
_________________________________________________________________
dropout_1 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_2 (Dense)              (None, 4096)              16781312
_________________________________________________________________
dropout_2 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                40970
=================================================================
Total params: 21,591,306
Trainable params: 21,590,602
Non-trainable params: 704
_________________________________________________________________
```

OUTPUT:



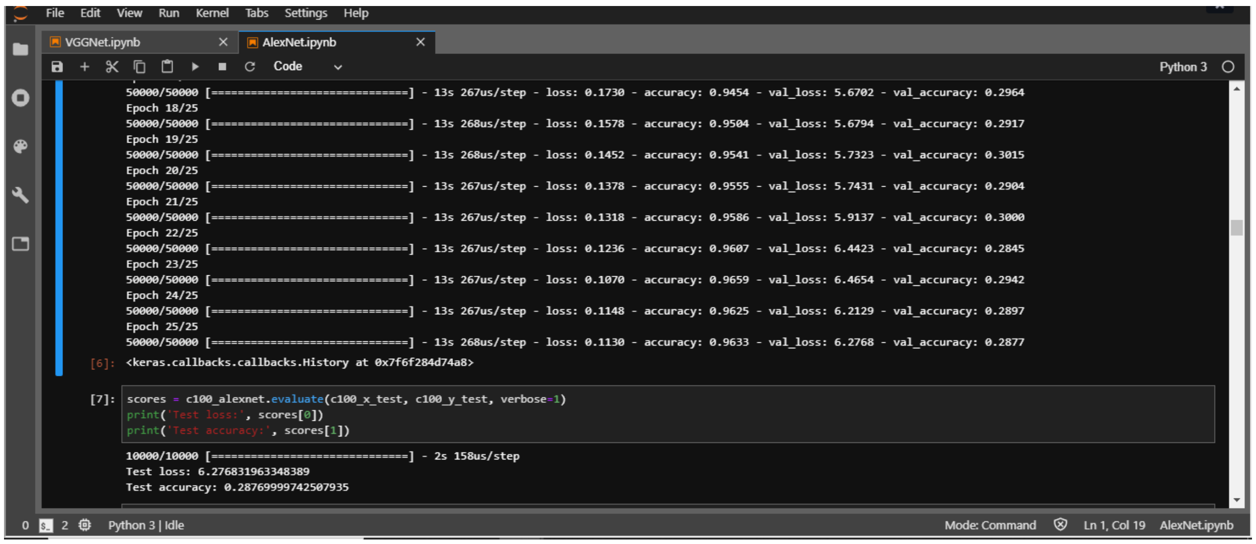
```
         50000/50000 [==============================] - 13s 265us/step - loss: 0.0683 - accuracy: 0.9768 - val_loss: 2.2618 - val_accuracy: 0.6113
         Epoch 19/25
         50000/50000 [==============================] - 13s 266us/step - loss: 0.0572 - accuracy: 0.9805 - val_loss: 2.4536 - val_accuracy: 0.5867
         Epoch 20/25
         50000/50000 [==============================] - 13s 265us/step - loss: 0.0518 - accuracy: 0.9837 - val_loss: 2.3634 - val_accuracy: 0.6043
         Epoch 21/25
         50000/50000 [==============================] - 13s 265us/step - loss: 0.0603 - accuracy: 0.9798 - val_loss: 2.3774 - val_accuracy: 0.6028
         Epoch 22/25
         50000/50000 [==============================] - 13s 266us/step - loss: 0.0542 - accuracy: 0.9822 - val_loss: 2.3571 - val_accuracy: 0.6068
         Epoch 23/25
         50000/50000 [==============================] - 13s 265us/step - loss: 0.0538 - accuracy: 0.9821 - val_loss: 2.5141 - val_accuracy: 0.5973
         Epoch 24/25
         50000/50000 [==============================] - 13s 265us/step - loss: 0.0551 - accuracy: 0.9823 - val_loss: 2.2076 - val_accuracy: 0.6098
         Epoch 25/25
         50000/50000 [==============================] - 13s 266us/step - loss: 0.0467 - accuracy: 0.9844 - val_loss: 2.3238 - val_accuracy: 0.6115
    [6]: <keras.callbacks.callbacks.History at 0x7f9e440d1e10>

    [7]: scores = c10_alexnet.evaluate(c10_x_test, c10_y_test, verbose=1)
         print('Test loss:', scores[0])
         print('Test accuracy:', scores[1])

         10000/10000 [==============================] - 2s 161us/step
         Test loss: 2.3238358577728273
         Test accuracy: 0.6115000247955322

    [8]: c10_alexnet.save('c10_alexnet.h5')
         c10_alexnet.save_weights('c10_alexnet_weights.h5')
```

Ciphar100:

```
# Load data
(c100_x_train_orig, c100_y_train_orig), (c100_x_test_orig, c100_y_test_orig) = cifar100.load_data()

# Pre-processing X
c100_x_train = c100_x_train_orig.astype('float32')
c100_x_test = c100_x_test_orig.astype('float32')
c100_x_train = c100_x_train/255
c100_x_test = c100_x_test/255

# Hyper-parameters
c100_epochs = 25
c100_batch_size = 128
c100_num_classes = 100
c100_input_shape = c100_x_train.shape[1:]

# Pre-processing Y
c100_y_train = keras.utils.to_categorical(c100_y_train_orig, c100_num_classes)
c100_y_test = keras.utils.to_categorical(c100_y_test_orig, c100_num_classes)

print(c100_x_train.shape, c100_x_test.shape, c100_y_train.shape, c100_y_test.shape)
c100_alexnet = Alexnet(c100_input_shape, c100_num_classes)
c100_alexnet.summary()
```

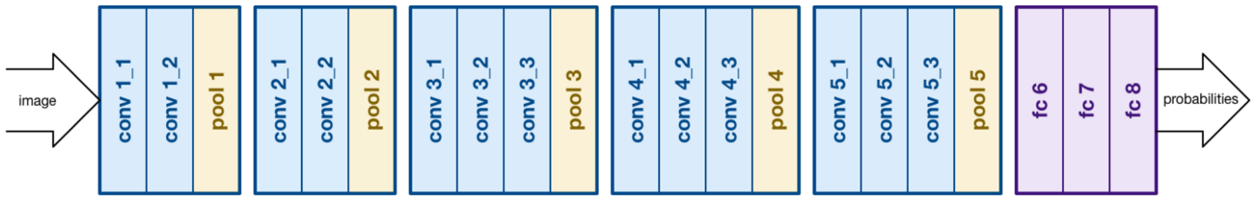
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 8, 8, 96)          2688
_________________________________________________________________
batch_normalization_1 (Batch (None, 8, 8, 96)          384
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 4, 4, 96)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 4, 4, 256)         614656
_________________________________________________________________
batch_normalization_2 (Batch (None, 4, 4, 256)         1024
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 2, 2, 256)         0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 2, 2, 384)         885120
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 2, 2, 384)         1327488
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 2, 2, 256)         884992
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 1, 1, 256)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 4096)              1052672
_________________________________________________________________
dropout_1 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_2 (Dense)              (None, 4096)              16781312
_________________________________________________________________
dropout_2 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_3 (Dense)              (None, 100)               409700
=================================================================
Total params: 21,960,036
Trainable params: 21,959,332
Non-trainable params: 704
```

Output:

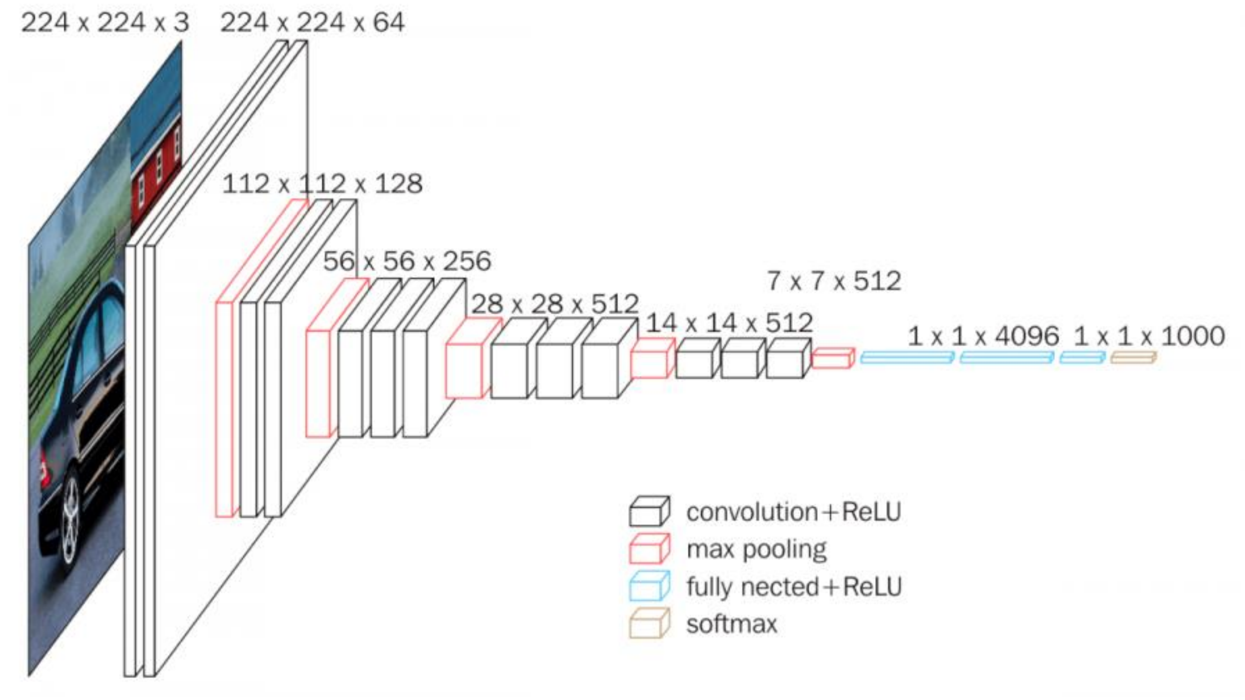


```
50000/50000 [==============================] - 13s 267us/step - loss: 0.1730 - accuracy: 0.9454 - val_loss: 5.6702 - val_accuracy: 0.2964
Epoch 18/25
50000/50000 [==============================] - 13s 268us/step - loss: 0.1578 - accuracy: 0.9504 - val_loss: 5.6794 - val_accuracy: 0.2917
Epoch 19/25
50000/50000 [==============================] - 13s 268us/step - loss: 0.1452 - accuracy: 0.9541 - val_loss: 5.7323 - val_accuracy: 0.3015
Epoch 20/25
50000/50000 [==============================] - 13s 267us/step - loss: 0.1378 - accuracy: 0.9555 - val_loss: 5.7431 - val_accuracy: 0.2904
Epoch 21/25
50000/50000 [==============================] - 13s 267us/step - loss: 0.1318 - accuracy: 0.9586 - val_loss: 5.9137 - val_accuracy: 0.3000
Epoch 22/25
50000/50000 [==============================] - 13s 267us/step - loss: 0.1236 - accuracy: 0.9607 - val_loss: 6.4423 - val_accuracy: 0.2845
Epoch 23/25
50000/50000 [==============================] - 13s 267us/step - loss: 0.1070 - accuracy: 0.9659 - val_loss: 6.4654 - val_accuracy: 0.2942
Epoch 24/25
50000/50000 [==============================] - 13s 267us/step - loss: 0.1148 - accuracy: 0.9625 - val_loss: 6.2129 - val_accuracy: 0.2897
Epoch 25/25
50000/50000 [==============================] - 13s 268us/step - loss: 0.1130 - accuracy: 0.9633 - val_loss: 6.2768 - val_accuracy: 0.2877
[6]: <keras.callbacks.callbacks.History at 0x7f6f284d74a8>

[7]: scores = c100_alexnet.evaluate(c100_x_test, c100_y_test, verbose=1)
     print('Test loss:', scores[0])
     print('Test accuracy:', scores[1])

     10000/10000 [==============================] - 2s 158us/step
     Test loss: 6.276831963348389
     Test accuracy: 0.28769999742507935
```

3.VGGNet:



VGGNet consists of 16 convolutional layers and is very appealing because of its very uniform architecture. Similar to AlexNet, only 3x3 convolutions, but lots of filters. Trained on 4 GPUs for 2–3 weeks. It is currently the most preferred choice in the community for extracting features from images. The weight configuration of the VGGNet is publicly available and has been used in many other applications and challenges as a baseline feature extractor. VGGNet consists of 138 million parameters, which can be a bit challenging to handle.



The input to cov1 layer is of fixed size 224 x 224 RGB image. The image is passed through a stack of convolutional (conv.) layers, where the filters were used with a very small receptive field: 3×3 (which is the smallest size to capture the notion of left/right, up/down, center). In one of the configurations, it also utilizes 1×1 convolution filters, which can be seen as a linear transformation of the input channels (followed by non-linearity). The convolution stride is fixed to 1 pixel; the spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1-pixel for 3×3 conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the conv. layers (not all the conv. layers are followed by max-pooling). Max-pooling is performed over a 2×2 pixel window, with stride 2.

Three Fully-Connected (FC) layers follow a stack of convolutional layers (which has a different depth in different architectures): the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer. The configuration of the fully connected layers is the same in all networks. All hidden layers are equipped with the rectification (ReLU) non-linearity. It is also noted that none of the networks (except for one) contain Local Response Normalisation (LRN), such normalization does not improve the performance on the ILSVRC dataset, but leads to increased memory consumption and computation time.

//Code

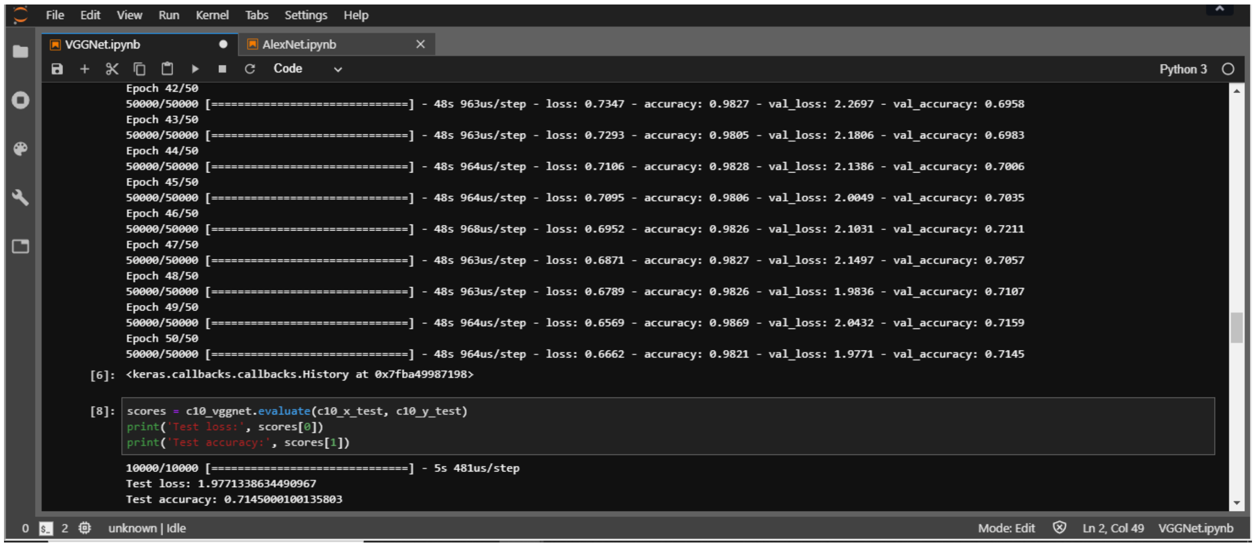
```
def Vggnet(input_shape, num_classes):
    vggnet = Sequential()
    weight_decay = 0.0001
    vggnet.add(Conv2D(64, (3,3), padding="same", activation="relu", kernel_regularizer=keras.regularizers.
l2(weight_decay), kernel_initializer=he_normal(), input_shape=input_shape))
    vggnet.add(BatchNormalization())
    vggnet.add(Conv2D(64, (3,3), padding="same", activation="relu", kernel_regularizer=keras.regularizers.
l2(weight_decay), kernel_initializer=he_normal()))
    vggnet.add(BatchNormalization())
    vggnet.add(MaxPooling2D((2,2),strides=(2,2)))
    vggnet.add(Conv2D(128, (3,3), padding="same", activation="relu", kernel_regularizer=keras.regularizers
.l2(weight_decay), kernel_initializer=he_normal()))
    vggnet.add(BatchNormalization())
    vggnet.add(Conv2D(128, (3,3), padding="same", activation="relu", kernel_regularizer=keras.regularizers
.l2(weight_decay), kernel_initializer=he_normal()))
    vggnet.add(BatchNormalization())
    vggnet.add(MaxPooling2D((2,2),strides=(2,2)))
    vggnet.add(Conv2D(256, (3,3), padding="same", activation="relu", kernel_regularizer=keras.regularizers
.l2(weight_decay), kernel_initializer=he_normal()))
    vggnet.add(BatchNormalization())
    vggnet.add(Conv2D(256, (3,3), padding="same", activation="relu", kernel_regularizer=keras.regularizers
.l2(weight_decay), kernel_initializer=he_normal()))
    vggnet.add(BatchNormalization())
    vggnet.add(Conv2D(256, (3,3), padding="same", activation="relu", kernel_regularizer=keras.regularizers
.l2(weight_decay), kernel_initializer=he_normal()))
    vggnet.add(BatchNormalization())
    vggnet.add(MaxPooling2D((2,2),strides=(2,2)))
    vggnet.add(Conv2D(512, (3,3), padding="same", activation="relu", kernel_regularizer=keras.regularizers
.l2(weight_decay), kernel_initializer=he_normal()))
    vggnet.add(BatchNormalization())
    vggnet.add(Conv2D(512, (3,3), padding="same", activation="relu", kernel_regularizer=keras.regularizers
.l2(weight_decay), kernel_initializer=he_normal()))
    vggnet.add(BatchNormalization())
    vggnet.add(Conv2D(512, (3,3), padding="same", activation="relu", kernel_regularizer=keras.regularizers
.l2(weight_decay), kernel_initializer=he_normal()))
    vggnet.add(BatchNormalization())
    vggnet.add(MaxPooling2D((2,2),strides=(2,2)))
    vggnet.add(Conv2D(512, (3,3), padding="same", activation="relu", kernel_regularizer=keras.regularizers
.l2(weight_decay), kernel_initializer=he_normal()))
    vggnet.add(BatchNormalization())
    vggnet.add(Conv2D(512, (3,3), padding="same", activation="relu", kernel_regularizer=keras.regularizers
.l2(weight_decay), kernel_initializer=he_normal()))
    vggnet.add(BatchNormalization())
    vggnet.add(Conv2D(512, (3,3), padding="same", activation="relu", kernel_regularizer=keras.regularizers
.l2(weight_decay), kernel_initializer=he_normal()))
    vggnet.add(BatchNormalization())
    vggnet.add(MaxPooling2D((2,2),strides=(2,2)))
    vggnet.add(Flatten())
    vggnet.add(Dense(units=4096, activation="relu", kernel_regularizer=keras.regularizers.l2(weight_decay)
, kernel_initializer=he_normal()))
    vggnet.add(BatchNormalization())
    vggnet.add(Dropout(0.5))
    vggnet.add(Dense(units=4096, activation="relu", kernel_regularizer=keras.regularizers.l2(weight_decay)
, kernel_initializer=he_normal()))
    vggnet.add(BatchNormalization())
    vggnet.add(Dropout(0.5))
    vggnet.add(Dense(units=num_classes, activation="softmax", kernel_regularizer=keras.regularizers.l2(wei
ght_decay), kernel_initializer=he_normal()))

    opt = keras.optimizers.Adam(learning_rate=0.0001)
    vggnet.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])
    return vggnet
```

Ciphar10:

```
# Load data
(c10_x_train_orig, c10_y_train_orig), (c10_x_test_orig, c10_y_test_orig) = cifar10.load_data()

# Pre-processing X
c10_x_train = c10_x_train_orig.astype('float32')
c10_x_test = c10_x_test_orig.astype('float32')
c10_x_train = c10_x_train/255
c10_x_test = c10_x_test/255

# Hyper-parameters
c10_epochs = 50
c10_batch_size = 128
c10_num_classes = 10
c10_input_shape = c10_x_train.shape[1:]

# Pre-processing Y
c10_y_train = keras.utils.to_categorical(c10_y_train_orig, c10_num_classes)
c10_y_test = keras.utils.to_categorical(c10_y_test_orig, c10_num_classes)

print(c10_x_train.shape, c10_x_test.shape, c10_y_train.shape, c10_y_test.shape)
(50000, 32, 32, 3) (10000, 32, 32, 3) (50000, 10) (10000, 10)

c10_vggnet = Vggnet(c10_input_shape, c10_num_classes)
c10_vggnet.summary()
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_1 (Conv2D)            (None, 32, 32, 64)        1792      
_________________________________________________________________
batch_normalization_1 (Batch (None, 32, 32, 64)        256       
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 64)        36928     
_________________________________________________________________
batch_normalization_2 (Batch (None, 32, 32, 64)        256       
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 64)        0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 128)       73856     
_________________________________________________________________
batch_normalization_3 (Batch (None, 16, 16, 128)       512       
_________________________________________________________________
```

```
conv2d_4 (Conv2D)            (None, 16, 16, 128)       147584    
_________________________________________________________________
batch_normalization_4 (Batch (None, 16, 16, 128)       512       
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 8, 8, 128)         0         
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 8, 8, 256)         295168    
_________________________________________________________________
batch_normalization_5 (Batch (None, 8, 8, 256)         1024      
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 8, 8, 256)         590080    
_________________________________________________________________
batch_normalization_6 (Batch (None, 8, 8, 256)         1024      
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 8, 8, 256)         590080    
_________________________________________________________________
batch_normalization_7 (Batch (None, 8, 8, 256)         1024      
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 4, 4, 256)         0         
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 4, 4, 512)         1180160   
_________________________________________________________________
batch_normalization_8 (Batch (None, 4, 4, 512)         2048      
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 4, 4, 512)         2359808   
_________________________________________________________________
batch_normalization_9 (Batch (None, 4, 4, 512)         2048      
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 4, 4, 512)         2359808   
_________________________________________________________________
batch_normalization_10 (Batc (None, 4, 4, 512)         2048      
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 2, 2, 512)         0         
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 2, 2, 512)         2359808   
_________________________________________________________________
batch_normalization_11 (Batc (None, 2, 2, 512)         2048      
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 2, 2, 512)         2359808   
_________________________________________________________________
batch_normalization_12 (Batc (None, 2, 2, 512)         2048      
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 2, 2, 512)         2359808   
_________________________________________________________________
batch_normalization_13 (Batc (None, 2, 2, 512)         2048      
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 1, 1, 512)         0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 512)               0         
_________________________________________________________________
dense_1 (Dense)              (None, 4096)              2101248   
_________________________________________________________________
batch_normalization_14 (Batc (None, 4096)              16384     
_________________________________________________________________
dropout_1 (Dropout)          (None, 4096)              0         
_________________________________________________________________
dense_2 (Dense)              (None, 4096)              16781312  
_________________________________________________________________
batch_normalization_15 (Batc (None, 4096)              16384     
_________________________________________________________________
dropout_2 (Dropout)          (None, 4096)              0         
_________________________________________________________________
dense_3 (Dense)              (None, 10)                40970     
=================================================================
Total params: 33,687,882
Trainable params: 33,663,050
Non-trainable params: 24,832
```

OUTPUT:



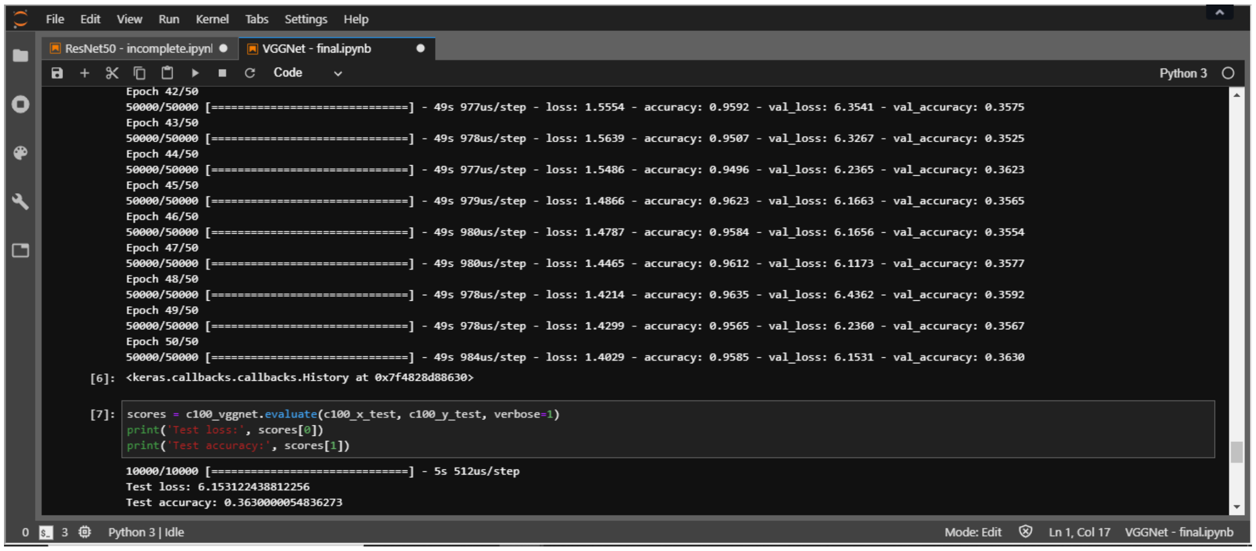
```
Epoch 42/50
50000/50000 [==============================] - 48s 963us/step - loss: 0.7347 - accuracy: 0.9827 - val_loss: 2.2697 - val_accuracy: 0.6958
Epoch 43/50
50000/50000 [==============================] - 48s 963us/step - loss: 0.7293 - accuracy: 0.9805 - val_loss: 2.1806 - val_accuracy: 0.6983
Epoch 44/50
50000/50000 [==============================] - 48s 964us/step - loss: 0.7106 - accuracy: 0.9828 - val_loss: 2.1386 - val_accuracy: 0.7006
Epoch 45/50
50000/50000 [==============================] - 48s 964us/step - loss: 0.7095 - accuracy: 0.9806 - val_loss: 2.0049 - val_accuracy: 0.7035
Epoch 46/50
50000/50000 [==============================] - 48s 968us/step - loss: 0.6952 - accuracy: 0.9826 - val_loss: 2.1031 - val_accuracy: 0.7211
Epoch 47/50
50000/50000 [==============================] - 48s 963us/step - loss: 0.6871 - accuracy: 0.9827 - val_loss: 2.1497 - val_accuracy: 0.7057
Epoch 48/50
50000/50000 [==============================] - 48s 963us/step - loss: 0.6789 - accuracy: 0.9826 - val_loss: 1.9836 - val_accuracy: 0.7107
Epoch 49/50
50000/50000 [==============================] - 48s 964us/step - loss: 0.6569 - accuracy: 0.9869 - val_loss: 2.0432 - val_accuracy: 0.7159
Epoch 50/50
50000/50000 [==============================] - 48s 964us/step - loss: 0.6662 - accuracy: 0.9821 - val_loss: 1.9771 - val_accuracy: 0.7145
[6]: <keras.callbacks.callbacks.History at 0x7fba49987198>

[8]: scores = c10_vggnet.evaluate(c10_x_test, c10_y_test)
     print('Test loss:', scores[0])
     print('Test accuracy:', scores[1])

10000/10000 [==============================] - 5s 481us/step
Test loss: 1.9771338634490967
Test accuracy: 0.7145000100135803
```

Ciphar 100:

```
# Load data
(c100_x_train_orig, c100_y_train_orig), (c100_x_test_orig, c100_y_test_orig) = cifar100.load_data()

# Pre-processing X
c100_x_train = c100_x_train_orig.astype('float32')
c100_x_test = c100_x_test_orig.astype('float32')
c100_x_train = c100_x_train/255
c100_x_test = c100_x_test/255

# Hyper-parameters
c100_epochs = 50
c100_batch_size = 128
c100_num_classes = 100
c100_input_shape = c100_x_train.shape[1:]

# Pre-processing Y
c100_y_train = keras.utils.to_categorical(c100_y_train_orig, c100_num_classes)
c100_y_test = keras.utils.to_categorical(c100_y_test_orig, c100_num_classes)

print(c100_x_train.shape, c100_x_test.shape, c100_y_train.shape, c100_y_test.shape)
c100_vggnet = Vggnet(c100_input_shape, c100_num_classes)
c100_vggnet.summary()
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_27 (Conv2D)           (None, 32, 32, 64)        1792
_________________________________________________________________
batch_normalization_31 (Batc (None, 32, 32, 64)        256
_________________________________________________________________
```

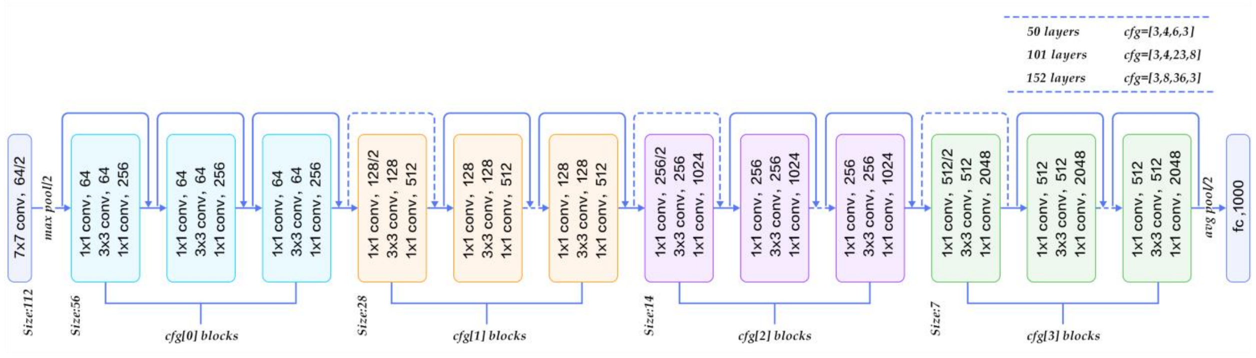
```
conv2d_28 (Conv2D)           (None, 32, 32, 64)        36928
_________________________________________________________________
batch_normalization_32 (Batc (None, 32, 32, 64)        256
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_29 (Conv2D)           (None, 16, 16, 128)       73856
_________________________________________________________________
batch_normalization_33 (Batc (None, 16, 16, 128)       512
_________________________________________________________________
conv2d_30 (Conv2D)           (None, 16, 16, 128)       147584
_________________________________________________________________
batch_normalization_34 (Batc (None, 16, 16, 128)       512
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 8, 8, 128)         0
_________________________________________________________________
conv2d_31 (Conv2D)           (None, 8, 8, 256)         295168
_________________________________________________________________
batch_normalization_35 (Batc (None, 8, 8, 256)         1024
_________________________________________________________________
conv2d_32 (Conv2D)           (None, 8, 8, 256)         590080
_________________________________________________________________
batch_normalization_36 (Batc (None, 8, 8, 256)         1024
_________________________________________________________________
conv2d_33 (Conv2D)           (None, 8, 8, 256)         590080
_________________________________________________________________
batch_normalization_37 (Batc (None, 8, 8, 256)         1024
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 4, 4, 256)         0
_________________________________________________________________
conv2d_34 (Conv2D)           (None, 4, 4, 512)         1180160
_________________________________________________________________
batch_normalization_38 (Batc (None, 4, 4, 512)         2048
_________________________________________________________________
conv2d_35 (Conv2D)           (None, 4, 4, 512)         2359808
_________________________________________________________________
batch_normalization_39 (Batc (None, 4, 4, 512)         2048
_________________________________________________________________
conv2d_36 (Conv2D)           (None, 4, 4, 512)         2359808
_________________________________________________________________
batch_normalization_40 (Batc (None, 4, 4, 512)         2048
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 2, 2, 512)         0
_________________________________________________________________
conv2d_37 (Conv2D)           (None, 2, 2, 512)         2359808
_________________________________________________________________
batch_normalization_41 (Batc (None, 2, 2, 512)         2048
_________________________________________________________________
conv2d_38 (Conv2D)           (None, 2, 2, 512)         2359808
_________________________________________________________________
batch_normalization_42 (Batc (None, 2, 2, 512)         2048
_________________________________________________________________
conv2d_39 (Conv2D)           (None, 2, 2, 512)         2359808
_________________________________________________________________
batch_normalization_43 (Batc (None, 2, 2, 512)         2048
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 1, 1, 512)         0
_________________________________________________________________
flatten_3 (Flatten)          (None, 512)               0
_________________________________________________________________
dense_7 (Dense)              (None, 4096)              2101248
_________________________________________________________________
batch_normalization_44 (Batc (None, 4096)              16384
_________________________________________________________________
dropout_5 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_8 (Dense)              (None, 4096)              16781312
_________________________________________________________________
batch_normalization_45 (Batc (None, 4096)              16384
_________________________________________________________________
dropout_6 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_9 (Dense)              (None, 100)               409700
=================================================================
Total params: 34,056,612
Trainable params: 34,031,780
Non-trainable params: 24,832
```

Output:



```
Epoch 42/50
50000/50000 [==============================] - 49s 977us/step - loss: 1.5554 - accuracy: 0.9592 - val_loss: 6.3541 - val_accuracy: 0.3575
Epoch 43/50
50000/50000 [==============================] - 49s 978us/step - loss: 1.5639 - accuracy: 0.9507 - val_loss: 6.3267 - val_accuracy: 0.3525
Epoch 44/50
50000/50000 [==============================] - 49s 977us/step - loss: 1.5486 - accuracy: 0.9496 - val_loss: 6.2365 - val_accuracy: 0.3623
Epoch 45/50
50000/50000 [==============================] - 49s 979us/step - loss: 1.4866 - accuracy: 0.9623 - val_loss: 6.1663 - val_accuracy: 0.3565
Epoch 46/50
50000/50000 [==============================] - 49s 980us/step - loss: 1.4787 - accuracy: 0.9584 - val_loss: 6.1656 - val_accuracy: 0.3554
Epoch 47/50
50000/50000 [==============================] - 49s 980us/step - loss: 1.4465 - accuracy: 0.9612 - val_loss: 6.1173 - val_accuracy: 0.3577
Epoch 48/50
50000/50000 [==============================] - 49s 978us/step - loss: 1.4214 - accuracy: 0.9635 - val_loss: 6.4362 - val_accuracy: 0.3592
Epoch 49/50
50000/50000 [==============================] - 49s 978us/step - loss: 1.4299 - accuracy: 0.9565 - val_loss: 6.2360 - val_accuracy: 0.3567
Epoch 50/50
50000/50000 [==============================] - 49s 984us/step - loss: 1.4029 - accuracy: 0.9585 - val_loss: 6.1531 - val_accuracy: 0.3630
[6]: <keras.callbacks.callbacks.History at 0x7f4828d88630>

[7]: scores = c100_vggnet.evaluate(c100_x_test, c100_y_test, verbose=1)
     print('Test loss:', scores[0])
     print('Test accuracy:', scores[1])

10000/10000 [==============================] - 5s 512us/step
Test loss: 6.153122438812256
Test accuracy: 0.3630000054836273
```

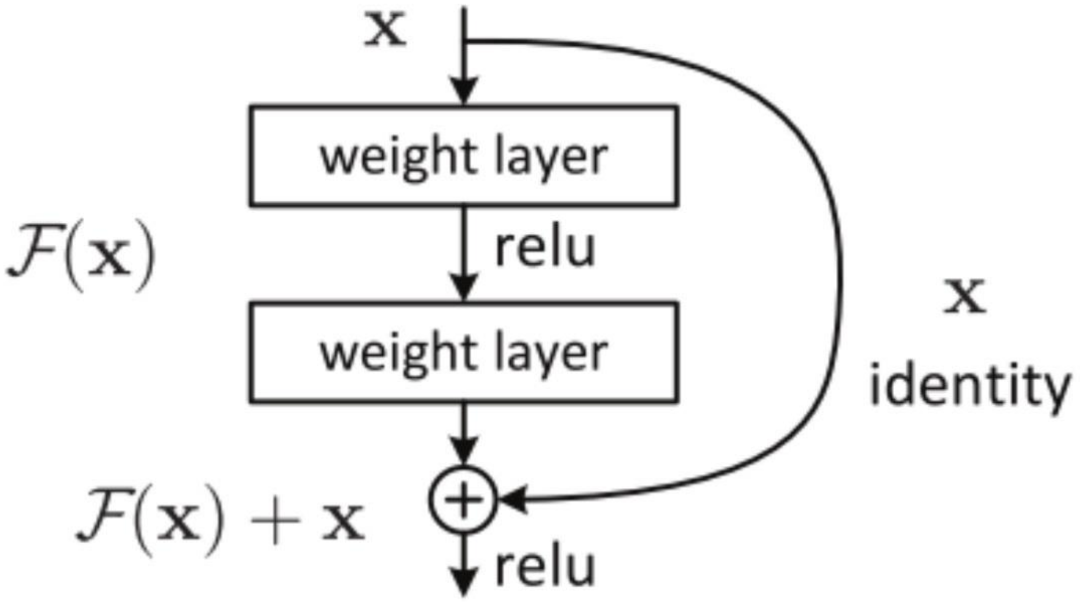
4.ResNet:



ResNet-50 is a deep residual network. The “50” refers to the number of layers it has. It’s a subclass of convolutional neural networks, with ResNet most popularly used for image classification.

The main innovation of ResNet is the skip connection. As you know, without adjustments, deep networks often suffer from vanishing gradients, ie: as the model backpropagates, the gradient gets smaller and smaller. Tiny gradients can make learning intractable.

The skip connection in the diagram below is labeled “identity.” It allows the network to learn the identity function, which allows it pass the the input through the block without passing through the other weight layers!


x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

a residual block

This allows you to stack additional layers and build a deeper network, offsetting the vanishing gradient by allowing your network to skip through layers of it feels they are less relevant in training.

//Code

```
def res_block(X, f, filters, stage, block, increase = False, s = 2):

    # Defining name basis
    conv_name_base = 'res' + str(stage) + block + '_branch'
    bn_name_base = 'bn' + str(stage) + block + '_branch'

    F1, F2, F3 = filters   # Retrieving filters
    X_shortcut = X         # Saving input value
    stride = (1,1)

    if increase:
        stride = (s,s)

        #Shortcut Path
        X_shortcut = Conv2D(F3, (1,1), strides = (s,s), name = conv_name_base + '1', kernel_initializer =
glorot_uniform(seed = 0))(X_shortcut)
        X_shortcut = BatchNormalization(axis = 3, name = bn_name_base + '1')(X_shortcut)

    # First component of main path
    X = Conv2D(filters = F1, kernel_size = (1, 1), strides = stride, padding = 'valid', name = conv_name_b
ase + '2a', kernel_initializer = glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis = 3, name = bn_name_base + '2a')(X)
    X = Activation('relu')(X)

    # Second component of main path (≈3 lines)
    X = Conv2D(filters = F2, kernel_size = (f, f), strides = (1,1), padding = 'same', name = conv_name_bas
e + '2b', kernel_initializer = glorot_uniform(seed = 0))(X)
    X = BatchNormalization(axis = 3, name = bn_name_base + '2b')(X)
    X = Activation('relu')(X)

    # Third component of main path (≈2 lines)
    X = Conv2D(filters  = F3, kernel_size = (1, 1), strides = (1,1), padding = 'valid', name = conv_name_b
ase + '2c', kernel_initializer = glorot_uniform(seed = 0))(X)
    X = BatchNormalization(axis = 3, name = bn_name_base + '2c')(X)

    # Final step: Add shortcut value to main path, and pass it through a RELU activation (≈2 lines)
    X = Add()([X_shortcut, X])
    X = Activation('relu')(X)

    return X
def ResNet50(input_shape, classes):

    X_input = Input(input_shape)

    # Zero-Padding
    X = ZeroPadding2D((3, 3))(X_input)

    # Stage 1
    X = Conv2D(64, (7, 7), strides = (2, 2), name = 'conv1', kernel_initializer = glorot_uniform(seed=0),
padding = 'same')(X)
    X = BatchNormalization(axis = 3, name = 'bn_conv1')(X)
    X = Activation('relu')(X)
    X = MaxPooling2D((3, 3), strides=(2, 2))(X)

    # Stage 2
    X = res_block(X, f = 3, filters = [64, 64, 256], stage = 2, block='a', s = 1, increase = True)
    X = res_block(X, 3, [64, 64, 256], stage=2, block='b')
    X = res_block(X, 3, [64, 64, 256], stage=2, block='c')
    # Stage 3 (≈4 lines)
    X = res_block(X, f = 3, filters = [128,128,512], stage = 3, block = 'a', s = 2, increase = True)
    X = res_block(X, 3, [128,128,512], stage = 3, block = 'b')
    X = res_block(X, 3, [128,128,512], stage = 3, block = 'c')
    X = res_block(X, 3, [128,128,512], stage = 3, block = 'd')

    # Stage 4 (≈6 lines)
    X = res_block(X, f = 3, filters = [256, 256, 1024], stage = 4, block = 'a', s = 2, increase = True)
    X = res_block(X, 3, [256, 256, 1024], stage = 4, block = 'b')
    X = res_block(X, 3, [256, 256, 1024], stage = 4, block = 'c')
    X = res_block(X, 3, [256, 256, 1024], stage = 4, block = 'd')
    X = res_block(X, 3, [256, 256, 1024], stage = 4, block = 'e')
    X = res_block(X, 3, [256, 256, 1024], stage = 4, block = 'f')

    # Stage 5 (≈3 lines)
    X = res_block(X, f = 3, filters = [512, 512, 2048], stage = 5, block = 'a', s = 2, increase = True)
    X = res_block(X, 3, [512, 512, 2048], stage = 5, block = 'b')
    X = res_block(X, 3, [512, 512, 2048], stage = 5, block = 'c')

    # AVGPOOL (≈1 line). Use "X = AveragePooling2D(...)(X)"
    X = AveragePooling2D((2,2))(X)

    # output layer
    X = Flatten()(X)
    X = Dense(classes, activation='softmax', name='output', kernel_initializer = glorot_uniform(seed=0))(X
)

    # Create model
    resnet = Model(inputs = X_input, outputs = X, name='ResNet50')
    opt = keras.optimizers.Adam(learning_rate=0.0001)
    resnet.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

    return resnet
```

Ciphar10:

```
(c10_x_train_orig, c10_y_train_orig), (c10_x_test_orig, c10_y_test_orig) = cifar10.load_data()

# Pre-processing X
c10_x_train = c10_x_train_orig.astype('float32')
c10_x_test = c10_x_test_orig.astype('float32')
c10_x_train = c10_x_train/255
c10_x_test = c10_x_test/255

# Hyper-parameters
```

```
c10_epochs = 50
c10_batch_size = 128
c10_num_classes = 10
c10_input_shape = c10_x_train.shape[1:]

# Pre-processing Y
c10_y_train = keras.utils.to_categorical(c10_y_train_orig, c10_num_classes)
c10_y_test = keras.utils.to_categorical(c10_y_test_orig, c10_num_classes)


print(c10_x_train.shape, c10_x_test.shape, c10_y_train.shape, c10_y_test.shape)
(50000, 32, 32, 3) (10000, 32, 32, 3) (50000, 10) (10000, 10)

c10_resnet = ResNet50(c10_input_shape, c10_num_classes)
c10_resnet.summary()
```

```
Model: "ResNet50"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            (None, 32, 32, 3)    0
__________________________________________________________________________________________________
zero_padding2d_1 (ZeroPadding2D (None, 38, 38, 3)    0           input_1[0][0]
__________________________________________________________________________________________________
conv1 (Conv2D)                  (None, 19, 19, 64)   9472        zero_padding2d_1[0][0]
__________________________________________________________________________________________________
bn_conv1 (BatchNormalization)   (None, 19, 19, 64)   256         conv1[0][0]
__________________________________________________________________________________________________
activation_1 (Activation)       (None, 19, 19, 64)   0           bn_conv1[0][0]
__________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 9, 9, 64)     0           activation_1[0][0]
__________________________________________________________________________________________________
res2a_branch2a (Conv2D)         (None, 9, 9, 64)     4160        max_pooling2d_1[0][0]
__________________________________________________________________________________________________
bn2a_branch2a (BatchNormalizati (None, 9, 9, 64)     256         res2a_branch2a[0][0]
__________________________________________________________________________________________________
activation_2 (Activation)       (None, 9, 9, 64)     0           bn2a_branch2a[0][0]
__________________________________________________________________________________________________
res2a_branch2b (Conv2D)         (None, 9, 9, 64)     36928       activation_2[0][0]
__________________________________________________________________________________________________
bn2a_branch2b (BatchNormalizati (None, 9, 9, 64)     256         res2a_branch2b[0][0]
__________________________________________________________________________________________________
activation_3 (Activation)       (None, 9, 9, 64)     0           bn2a_branch2b[0][0]
__________________________________________________________________________________________________
res2a_branch1 (Conv2D)          (None, 9, 9, 256)    16640       max_pooling2d_1[0][0]
__________________________________________________________________________________________________
res2a_branch2c (Conv2D)         (None, 9, 9, 256)    16640       activation_3[0][0]
__________________________________________________________________________________________________
bn2a_branch1 (BatchNormalizatio (None, 9, 9, 256)    1024        res2a_branch1[0][0]
__________________________________________________________________________________________________
bn2a_branch2c (BatchNormalizati (None, 9, 9, 256)    1024        res2a_branch2c[0][0]
__________________________________________________________________________________________________
add_1 (Add)                     (None, 9, 9, 256)    0           bn2a_branch1[0][0]
                                                                 bn2a_branch2c[0][0]
__________________________________________________________________________________________________
activation_4 (Activation)       (None, 9, 9, 256)    0           add_1[0][0]
__________________________________________________________________________________________________
res2b_branch2a (Conv2D)         (None, 9, 9, 64)     16448       activation_4[0][0]
__________________________________________________________________________________________________
bn2b_branch2a (BatchNormalizati (None, 9, 9, 64)     256         res2b_branch2a[0][0]
__________________________________________________________________________________________________
activation_5 (Activation)       (None, 9, 9, 64)     0           bn2b_branch2a[0][0]
```

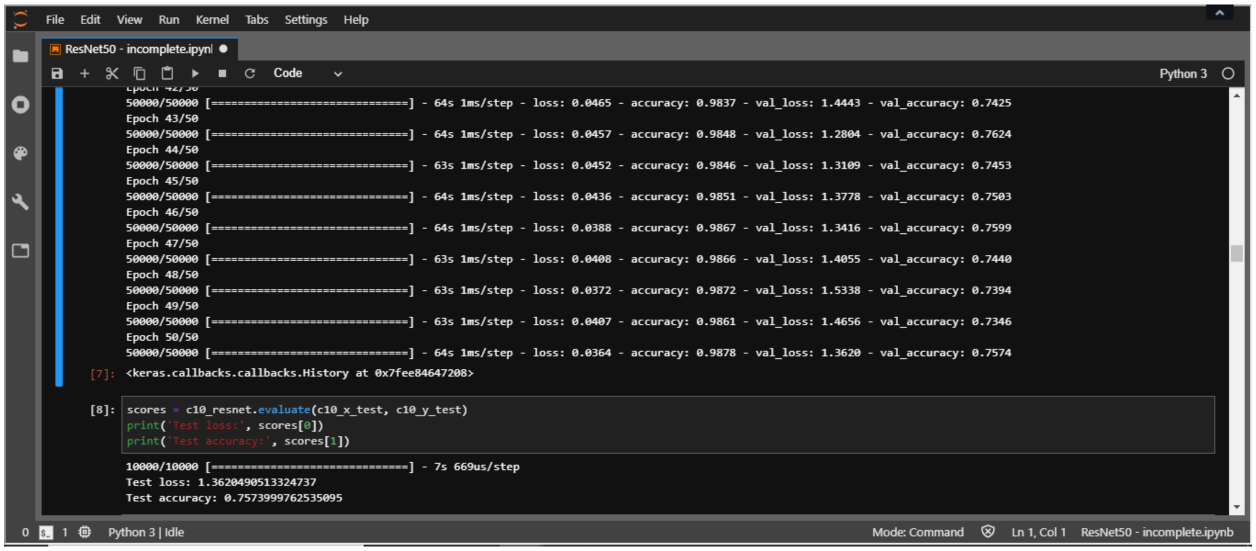
```
__________________________________________________________________________________________________
res2b_branch2b (Conv2D)         (None, 9, 9, 64)     36928       activation_5[0][0]
__________________________________________________________________________________________________
bn2b_branch2b (BatchNormalizati (None, 9, 9, 64)     256         res2b_branch2b[0][0]
__________________________________________________________________________________________________
activation_6 (Activation)       (None, 9, 9, 64)     0           bn2b_branch2b[0][0]
__________________________________________________________________________________________________
res2b_branch2c (Conv2D)         (None, 9, 9, 256)    16640       activation_6[0][0]
__________________________________________________________________________________________________
bn2b_branch2c (BatchNormalizati (None, 9, 9, 256)    1024        res2b_branch2c[0][0]
__________________________________________________________________________________________________
add_2 (Add)                     (None, 9, 9, 256)    0           activation_4[0][0]
                                                                 bn2b_branch2c[0][0]
__________________________________________________________________________________________________
activation_7 (Activation)       (None, 9, 9, 256)    0           add_2[0][0]
__________________________________________________________________________________________________
res2c_branch2a (Conv2D)         (None, 9, 9, 64)     16448       activation_7[0][0]
__________________________________________________________________________________________________
bn2c_branch2a (BatchNormalizati (None, 9, 9, 64)     256         res2c_branch2a[0][0]
__________________________________________________________________________________________________
activation_8 (Activation)       (None, 9, 9, 64)     0           bn2c_branch2a[0][0]
__________________________________________________________________________________________________
res2c_branch2b (Conv2D)         (None, 9, 9, 64)     36928       activation_8[0][0]
__________________________________________________________________________________________________
bn2c_branch2b (BatchNormalizati (None, 9, 9, 64)     256         res2c_branch2b[0][0]
__________________________________________________________________________________________________
activation_9 (Activation)       (None, 9, 9, 64)     0           bn2c_branch2b[0][0]
__________________________________________________________________________________________________
res2c_branch2c (Conv2D)         (None, 9, 9, 256)    16640       activation_9[0][0]
__________________________________________________________________________________________________
bn2c_branch2c (BatchNormalizati (None, 9, 9, 256)    1024        res2c_branch2c[0][0]
__________________________________________________________________________________________________
add_3 (Add)                     (None, 9, 9, 256)    0           activation_7[0][0]
                                                                 bn2c_branch2c[0][0]
__________________________________________________________________________________________________
activation_10 (Activation)      (None, 9, 9, 256)    0           add_3[0][0]
__________________________________________________________________________________________________
res3a_branch2a (Conv2D)         (None, 5, 5, 128)    32896       activation_10[0][0]
__________________________________________________________________________________________________
bn3a_branch2a (BatchNormalizati (None, 5, 5, 128)    512         res3a_branch2a[0][0]
__________________________________________________________________________________________________
activation_11 (Activation)      (None, 5, 5, 128)    0           bn3a_branch2a[0][0]
__________________________________________________________________________________________________
res3a_branch2b (Conv2D)         (None, 5, 5, 128)    147584      activation_11[0][0]
__________________________________________________________________________________________________
bn3a_branch2b (BatchNormalizati (None, 5, 5, 128)    512         res3a_branch2b[0][0]
__________________________________________________________________________________________________
activation_12 (Activation)      (None, 5, 5, 128)    0           bn3a_branch2b[0][0]
__________________________________________________________________________________________________
res3a_branch1 (Conv2D)          (None, 5, 5, 512)    131584      activation_10[0][0]
__________________________________________________________________________________________________
res3a_branch2c (Conv2D)         (None, 5, 5, 512)    66048       activation_12[0][0]
__________________________________________________________________________________________________
bn3a_branch1 (BatchNormalizatio (None, 5, 5, 512)    2048        res3a_branch1[0][0]
__________________________________________________________________________________________________
bn3a_branch2c (BatchNormalizati (None, 5, 5, 512)    2048        res3a_branch2c[0][0]
__________________________________________________________________________________________________
add_4 (Add)                     (None, 5, 5, 512)    0           bn3a_branch1[0][0]
                                                                 bn3a_branch2c[0][0]
__________________________________________________________________________________________________
activation_13 (Activation)      (None, 5, 5, 512)    0           add_4[0][0]
__________________________________________________________________________________________________
res3b_branch2a (Conv2D)         (None, 5, 5, 128)    65664       activation_13[0][0]
__________________________________________________________________________________________________
bn3b_branch2a (BatchNormalizati (None, 5, 5, 128)    512         res3b_branch2a[0][0]
__________________________________________________________________________________________________
activation_14 (Activation)      (None, 5, 5, 128)    0           bn3b_branch2a[0][0]
__________________________________________________________________________________________________
res3b_branch2b (Conv2D)         (None, 5, 5, 128)    147584      activation_14[0][0]
```

```
________________________________________________________________________________________________________
bn3b_branch2b (BatchNormalizati (None, 5, 5, 128)    512         res3b_branch2b[0][0]
________________________________________________________________________________________________________
activation_15 (Activation)      (None, 5, 5, 128)    0           bn3b_branch2b[0][0]
________________________________________________________________________________________________________
res3b_branch2c (Conv2D)         (None, 5, 5, 512)    66048       activation_15[0][0]
________________________________________________________________________________________________________
bn3b_branch2c (BatchNormalizati (None, 5, 5, 512)    2048        res3b_branch2c[0][0]
________________________________________________________________________________________________________
add_5 (Add)                     (None, 5, 5, 512)    0           activation_13[0][0]
                                                                 bn3b_branch2c[0][0]
________________________________________________________________________________________________________
activation_16 (Activation)      (None, 5, 5, 512)    0           add_5[0][0]
________________________________________________________________________________________________________
res3c_branch2a (Conv2D)         (None, 5, 5, 128)    65664       activation_16[0][0]
________________________________________________________________________________________________________
bn3c_branch2a (BatchNormalizati (None, 5, 5, 128)    512         res3c_branch2a[0][0]
________________________________________________________________________________________________________
activation_17 (Activation)      (None, 5, 5, 128)    0           bn3c_branch2a[0][0]
________________________________________________________________________________________________________
res3c_branch2b (Conv2D)         (None, 5, 5, 128)    147584      activation_17[0][0]
________________________________________________________________________________________________________
bn3c_branch2b (BatchNormalizati (None, 5, 5, 128)    512         res3c_branch2b[0][0]
________________________________________________________________________________________________________
activation_18 (Activation)      (None, 5, 5, 128)    0           bn3c_branch2b[0][0]
________________________________________________________________________________________________________
res3c_branch2c (Conv2D)         (None, 5, 5, 512)    66048       activation_18[0][0]
________________________________________________________________________________________________________
bn3c_branch2c (BatchNormalizati (None, 5, 5, 512)    2048        res3c_branch2c[0][0]
________________________________________________________________________________________________________
add_6 (Add)                     (None, 5, 5, 512)    0           activation_16[0][0]
                                                                 bn3c_branch2c[0][0]
________________________________________________________________________________________________________
activation_19 (Activation)      (None, 5, 5, 512)    0           add_6[0][0]
________________________________________________________________________________________________________
res3d_branch2a (Conv2D)         (None, 5, 5, 128)    65664       activation_19[0][0]
________________________________________________________________________________________________________
bn3d_branch2a (BatchNormalizati (None, 5, 5, 128)    512         res3d_branch2a[0][0]
________________________________________________________________________________________________________
activation_20 (Activation)      (None, 5, 5, 128)    0           bn3d_branch2a[0][0]
________________________________________________________________________________________________________
res3d_branch2b (Conv2D)         (None, 5, 5, 128)    147584      activation_20[0][0]
________________________________________________________________________________________________________
bn3d_branch2b (BatchNormalizati (None, 5, 5, 128)    512         res3d_branch2b[0][0]
________________________________________________________________________________________________________
activation_21 (Activation)      (None, 5, 5, 128)    0           bn3d_branch2b[0][0]
________________________________________________________________________________________________________
res3d_branch2c (Conv2D)         (None, 5, 5, 512)    66048       activation_21[0][0]
________________________________________________________________________________________________________
bn3d_branch2c (BatchNormalizati (None, 5, 5, 512)    2048        res3d_branch2c[0][0]
________________________________________________________________________________________________________
add_7 (Add)                     (None, 5, 5, 512)    0           activation_19[0][0]
                                                                 bn3d_branch2c[0][0]
________________________________________________________________________________________________________
activation_22 (Activation)      (None, 5, 5, 512)    0           add_7[0][0]
________________________________________________________________________________________________________
res4a_branch2a (Conv2D)         (None, 3, 3, 256)    131328      activation_22[0][0]
________________________________________________________________________________________________________
bn4a_branch2a (BatchNormalizati (None, 3, 3, 256)    1024        res4a_branch2a[0][0]
________________________________________________________________________________________________________
activation_23 (Activation)      (None, 3, 3, 256)    0           bn4a_branch2a[0][0]
________________________________________________________________________________________________________
res4a_branch2b (Conv2D)         (None, 3, 3, 256)    590080      activation_23[0][0]
________________________________________________________________________________________________________
bn4a_branch2b (BatchNormalizati (None, 3, 3, 256)    1024        res4a_branch2b[0][0]
________________________________________________________________________________________________________
activation_24 (Activation)      (None, 3, 3, 256)    0           bn4a_branch2b[0][0]
________________________________________________________________________________________________________
res4a_branch1 (Conv2D)          (None, 3, 3, 1024)   525312      activation_22[0][0]
```

```
__________________________________________________________________________________________________
res4a_branch2c (Conv2D)         (None, 3, 3, 1024)   263168      activation_24[0][0]
__________________________________________________________________________________________________
bn4a_branch1 (BatchNormalizatio (None, 3, 3, 1024)   4096        res4a_branch1[0][0]
__________________________________________________________________________________________________
bn4a_branch2c (BatchNormalizati (None, 3, 3, 1024)   4096        res4a_branch2c[0][0]
__________________________________________________________________________________________________
add_8 (Add)                     (None, 3, 3, 1024)   0           bn4a_branch1[0][0]
                                                                 bn4a_branch2c[0][0]
__________________________________________________________________________________________________
activation_25 (Activation)      (None, 3, 3, 1024)   0           add_8[0][0]
__________________________________________________________________________________________________
res4b_branch2a (Conv2D)         (None, 3, 3, 256)    262400      activation_25[0][0]
__________________________________________________________________________________________________
bn4b_branch2a (BatchNormalizati (None, 3, 3, 256)    1024        res4b_branch2a[0][0]
__________________________________________________________________________________________________
activation_26 (Activation)      (None, 3, 3, 256)    0           bn4b_branch2a[0][0]
__________________________________________________________________________________________________
res4b_branch2b (Conv2D)         (None, 3, 3, 256)    590080      activation_26[0][0]
__________________________________________________________________________________________________
bn4b_branch2b (BatchNormalizati (None, 3, 3, 256)    1024        res4b_branch2b[0][0]
__________________________________________________________________________________________________
activation_27 (Activation)      (None, 3, 3, 256)    0           bn4b_branch2b[0][0]
__________________________________________________________________________________________________
res4b_branch2c (Conv2D)         (None, 3, 3, 1024)   263168      activation_27[0][0]
__________________________________________________________________________________________________
bn4b_branch2c (BatchNormalizati (None, 3, 3, 1024)   4096        res4b_branch2c[0][0]
__________________________________________________________________________________________________
add_9 (Add)                     (None, 3, 3, 1024)   0           activation_25[0][0]
                                                                 bn4b_branch2c[0][0]
__________________________________________________________________________________________________
activation_28 (Activation)      (None, 3, 3, 1024)   0           add_9[0][0]
__________________________________________________________________________________________________
res4c_branch2a (Conv2D)         (None, 3, 3, 256)    262400      activation_28[0][0]
__________________________________________________________________________________________________
bn4c_branch2a (BatchNormalizati (None, 3, 3, 256)    1024        res4c_branch2a[0][0]
__________________________________________________________________________________________________
activation_29 (Activation)      (None, 3, 3, 256)    0           bn4c_branch2a[0][0]
__________________________________________________________________________________________________
res4c_branch2b (Conv2D)         (None, 3, 3, 256)    590080      activation_29[0][0]
__________________________________________________________________________________________________
bn4c_branch2b (BatchNormalizati (None, 3, 3, 256)    1024        res4c_branch2b[0][0]
__________________________________________________________________________________________________
activation_30 (Activation)      (None, 3, 3, 256)    0           bn4c_branch2b[0][0]
__________________________________________________________________________________________________
res4c_branch2c (Conv2D)         (None, 3, 3, 1024)   263168      activation_30[0][0]
__________________________________________________________________________________________________
bn4c_branch2c (BatchNormalizati (None, 3, 3, 1024)   4096        res4c_branch2c[0][0]
__________________________________________________________________________________________________
add_10 (Add)                    (None, 3, 3, 1024)   0           activation_28[0][0]
                                                                 bn4c_branch2c[0][0]
__________________________________________________________________________________________________
activation_31 (Activation)      (None, 3, 3, 1024)   0           add_10[0][0]
__________________________________________________________________________________________________
res4d_branch2a (Conv2D)         (None, 3, 3, 256)    262400      activation_31[0][0]
__________________________________________________________________________________________________
bn4d_branch2a (BatchNormalizati (None, 3, 3, 256)    1024        res4d_branch2a[0][0]
__________________________________________________________________________________________________
activation_32 (Activation)      (None, 3, 3, 256)    0           bn4d_branch2a[0][0]
__________________________________________________________________________________________________
res4d_branch2b (Conv2D)         (None, 3, 3, 256)    590080      activation_32[0][0]
__________________________________________________________________________________________________
bn4d_branch2b (BatchNormalizati (None, 3, 3, 256)    1024        res4d_branch2b[0][0]
__________________________________________________________________________________________________
activation_33 (Activation)      (None, 3, 3, 256)    0           bn4d_branch2b[0][0]
__________________________________________________________________________________________________
res4d_branch2c (Conv2D)         (None, 3, 3, 1024)   263168      activation_33[0][0]
__________________________________________________________________________________________________
bn4d_branch2c (BatchNormalizati (None, 3, 3, 1024)   4096        res4d_branch2c[0][0]
________________________________________________________________________________________________________
add_11 (Add)                    (None, 3, 3, 1024)   0           activation_31[0][0]
                                                                 bn4d_branch2c[0][0]
________________________________________________________________________________________________________
activation_34 (Activation)      (None, 3, 3, 1024)   0           add_11[0][0]
________________________________________________________________________________________________________
res4e_branch2a (Conv2D)         (None, 3, 3, 256)    262400      activation_34[0][0]
________________________________________________________________________________________________________
bn4e_branch2a (BatchNormalizati (None, 3, 3, 256)    1024        res4e_branch2a[0][0]
________________________________________________________________________________________________________
activation_35 (Activation)      (None, 3, 3, 256)    0           bn4e_branch2a[0][0]
________________________________________________________________________________________________________
res4e_branch2b (Conv2D)         (None, 3, 3, 256)    590080      activation_35[0][0]
________________________________________________________________________________________________________
bn4e_branch2b (BatchNormalizati (None, 3, 3, 256)    1024        res4e_branch2b[0][0]
________________________________________________________________________________________________________
activation_36 (Activation)      (None, 3, 3, 256)    0           bn4e_branch2b[0][0]
________________________________________________________________________________________________________
res4e_branch2c (Conv2D)         (None, 3, 3, 1024)   263168      activation_36[0][0]
________________________________________________________________________________________________________
bn4e_branch2c (BatchNormalizati (None, 3, 3, 1024)   4096        res4e_branch2c[0][0]
________________________________________________________________________________________________________
add_12 (Add)                    (None, 3, 3, 1024)   0           activation_34[0][0]
                                                                 bn4e_branch2c[0][0]
________________________________________________________________________________________________________
activation_37 (Activation)      (None, 3, 3, 1024)   0           add_12[0][0]
________________________________________________________________________________________________________
res4f_branch2a (Conv2D)         (None, 3, 3, 256)    262400      activation_37[0][0]
________________________________________________________________________________________________________
bn4f_branch2a (BatchNormalizati (None, 3, 3, 256)    1024        res4f_branch2a[0][0]
________________________________________________________________________________________________________
activation_38 (Activation)      (None, 3, 3, 256)    0           bn4f_branch2a[0][0]
________________________________________________________________________________________________________
res4f_branch2b (Conv2D)         (None, 3, 3, 256)    590080      activation_38[0][0]
________________________________________________________________________________________________________
bn4f_branch2b (BatchNormalizati (None, 3, 3, 256)    1024        res4f_branch2b[0][0]
________________________________________________________________________________________________________
activation_39 (Activation)      (None, 3, 3, 256)    0           bn4f_branch2b[0][0]
________________________________________________________________________________________________________
res4f_branch2c (Conv2D)         (None, 3, 3, 1024)   263168      activation_39[0][0]
________________________________________________________________________________________________________
bn4f_branch2c (BatchNormalizati (None, 3, 3, 1024)   4096        res4f_branch2c[0][0]
________________________________________________________________________________________________________
add_13 (Add)                    (None, 3, 3, 1024)   0           activation_37[0][0]
                                                                 bn4f_branch2c[0][0]
________________________________________________________________________________________________________
activation_40 (Activation)      (None, 3, 3, 1024)   0           add_13[0][0]
________________________________________________________________________________________________________
res5a_branch2a (Conv2D)         (None, 2, 2, 512)    524800      activation_40[0][0]
________________________________________________________________________________________________________
bn5a_branch2a (BatchNormalizati (None, 2, 2, 512)    2048        res5a_branch2a[0][0]
________________________________________________________________________________________________________
activation_41 (Activation)      (None, 2, 2, 512)    0           bn5a_branch2a[0][0]
________________________________________________________________________________________________________
res5a_branch2b (Conv2D)         (None, 2, 2, 512)    2359808     activation_41[0][0]
________________________________________________________________________________________________________
bn5a_branch2b (BatchNormalizati (None, 2, 2, 512)    2048        res5a_branch2b[0][0]
________________________________________________________________________________________________________
activation_42 (Activation)      (None, 2, 2, 512)    0           bn5a_branch2b[0][0]
________________________________________________________________________________________________________
res5a_branch1 (Conv2D)          (None, 2, 2, 2048)   2099200     activation_40[0][0]
________________________________________________________________________________________________________
res5a_branch2c (Conv2D)         (None, 2, 2, 2048)   1050624     activation_42[0][0]
________________________________________________________________________________________________________
bn5a_branch1 (BatchNormalizatio (None, 2, 2, 2048)   8192        res5a_branch1[0][0]
________________________________________________________________________________________________________
bn5a_branch2c (BatchNormalizati (None, 2, 2, 2048)   8192        res5a_branch2c[0][0]
________________________________________________________________________________________________________
add_14 (Add)                    (None, 2, 2, 2048)   0           bn5a_branch1[0][0]
```

```
                                                                 bn5a_branch2c[0][0]
__________________________________________________________________________________________________
activation_43 (Activation)      (None, 2, 2, 2048)   0           add_14[0][0]
__________________________________________________________________________________________________
res5b_branch2a (Conv2D)         (None, 2, 2, 512)    1049088     activation_43[0][0]
__________________________________________________________________________________________________
bn5b_branch2a (BatchNormalizati (None, 2, 2, 512)    2048        res5b_branch2a[0][0]
__________________________________________________________________________________________________
activation_44 (Activation)      (None, 2, 2, 512)    0           bn5b_branch2a[0][0]
__________________________________________________________________________________________________
res5b_branch2b (Conv2D)         (None, 2, 2, 512)    2359808     activation_44[0][0]
__________________________________________________________________________________________________
bn5b_branch2b (BatchNormalizati (None, 2, 2, 512)    2048        res5b_branch2b[0][0]
__________________________________________________________________________________________________
activation_45 (Activation)      (None, 2, 2, 512)    0           bn5b_branch2b[0][0]
__________________________________________________________________________________________________
res5b_branch2c (Conv2D)         (None, 2, 2, 2048)   1050624     activation_45[0][0]
__________________________________________________________________________________________________
bn5b_branch2c (BatchNormalizati (None, 2, 2, 2048)   8192        res5b_branch2c[0][0]
__________________________________________________________________________________________________
add_15 (Add)                    (None, 2, 2, 2048)   0           activation_43[0][0]
                                                                 bn5b_branch2c[0][0]
__________________________________________________________________________________________________
activation_46 (Activation)      (None, 2, 2, 2048)   0           add_15[0][0]
__________________________________________________________________________________________________
res5c_branch2a (Conv2D)         (None, 2, 2, 512)    1049088     activation_46[0][0]
__________________________________________________________________________________________________
bn5c_branch2a (BatchNormalizati (None, 2, 2, 512)    2048        res5c_branch2a[0][0]
__________________________________________________________________________________________________
activation_47 (Activation)      (None, 2, 2, 512)    0           bn5c_branch2a[0][0]
__________________________________________________________________________________________________
res5c_branch2b (Conv2D)         (None, 2, 2, 512)    2359808     activation_47[0][0]
__________________________________________________________________________________________________
bn5c_branch2b (BatchNormalizati (None, 2, 2, 512)    2048        res5c_branch2b[0][0]
__________________________________________________________________________________________________
activation_48 (Activation)      (None, 2, 2, 512)    0           bn5c_branch2b[0][0]
__________________________________________________________________________________________________
res5c_branch2c (Conv2D)         (None, 2, 2, 2048)   1050624     activation_48[0][0]
__________________________________________________________________________________________________
bn5c_branch2c (BatchNormalizati (None, 2, 2, 2048)   8192        res5c_branch2c[0][0]
__________________________________________________________________________________________________
add_16 (Add)                    (None, 2, 2, 2048)   0           activation_46[0][0]
                                                                 bn5c_branch2c[0][0]
__________________________________________________________________________________________________
activation_49 (Activation)      (None, 2, 2, 2048)   0           add_16[0][0]
__________________________________________________________________________________________________
average_pooling2d_1 (AveragePoo (None, 1, 1, 2048)   0           activation_49[0][0]
__________________________________________________________________________________________________
flatten_1 (Flatten)             (None, 2048)         0           average_pooling2d_1[0][0]
__________________________________________________________________________________________________
output (Dense)                  (None, 10)           20490       flatten_1[0][0]
==================================================================================================
Total params: 23,608,202
Trainable params: 23,555,082
Non-trainable params: 53,120
__________________________________________________________________________________________________
```

OUTPUT:


File Edit View Run Kernel Tabs Settings Help
ResNet50 - incomplete.ipynl
Code
Python 3
0 1 Python 3 | Idle
Mode: Command
Ln 1, Col 1
ResNet50 - incomplete.ipynb

```
50000/50000 [==============================] - 64s 1ms/step - loss: 0.0465 - accuracy: 0.9837 - val_loss: 1.4443 - val_accuracy: 0.7425
Epoch 43/50
50000/50000 [==============================] - 64s 1ms/step - loss: 0.0457 - accuracy: 0.9848 - val_loss: 1.2804 - val_accuracy: 0.7624
Epoch 44/50
50000/50000 [==============================] - 63s 1ms/step - loss: 0.0452 - accuracy: 0.9846 - val_loss: 1.3109 - val_accuracy: 0.7453
Epoch 45/50
50000/50000 [==============================] - 64s 1ms/step - loss: 0.0436 - accuracy: 0.9851 - val_loss: 1.3778 - val_accuracy: 0.7503
Epoch 46/50
50000/50000 [==============================] - 64s 1ms/step - loss: 0.0388 - accuracy: 0.9867 - val_loss: 1.3416 - val_accuracy: 0.7599
Epoch 47/50
50000/50000 [==============================] - 63s 1ms/step - loss: 0.0408 - accuracy: 0.9866 - val_loss: 1.4055 - val_accuracy: 0.7440
Epoch 48/50
50000/50000 [==============================] - 63s 1ms/step - loss: 0.0372 - accuracy: 0.9872 - val_loss: 1.5338 - val_accuracy: 0.7394
Epoch 49/50
50000/50000 [==============================] - 63s 1ms/step - loss: 0.0407 - accuracy: 0.9861 - val_loss: 1.4656 - val_accuracy: 0.7346
Epoch 50/50
50000/50000 [==============================] - 64s 1ms/step - loss: 0.0364 - accuracy: 0.9878 - val_loss: 1.3620 - val_accuracy: 0.7574
[7]: <keras.callbacks.callbacks.History at 0x7fee84647208>

[8]: scores = c10_resnet.evaluate(c10_x_test, c10_y_test)
     print('Test loss:', scores[0])
     print('Test accuracy:', scores[1])

10000/10000 [==============================] - 7s 669us/step
Test loss: 1.3620490513324737
Test accuracy: 0.7573999762535095
```

Ciphar100:

```
# Load data
(c100_x_train_orig, c100_y_train_orig), (c100_x_test_orig, c100_y_test_orig) = cifar100.load_data()

# Pre-processing X
c100_x_train = c100_x_train_orig.astype('float32')
c100_x_test = c100_x_test_orig.astype('float32')
c100_x_train = c100_x_train/255
c100_x_test = c100_x_test/255

# Hyper-parameters
c100_epochs = 50
c100_batch_size = 128
c100_num_classes = 100
c100_input_shape = c100_x_train.shape[1:]

# Pre-processing Y
c100_y_train = keras.utils.to_categorical(c100_y_train_orig, c100_num_classes)
c100_y_test = keras.utils.to_categorical(c100_y_test_orig, c100_num_classes)

print(c100_x_train.shape, c100_x_test.shape, c100_y_train.shape, c100_y_test.shape)
(50000, 32, 32, 3) (10000, 32, 32, 3) (50000, 100) (10000, 100)

c100_resnet = ResNet50(c100_input_shape, c100_num_classes)
c100_resnet.summary()
Model: "ResNet50"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            (None, 32, 32, 3)    0
__________________________________________________________________________________________________
zero_padding2d_1 (ZeroPadding2D (None, 38, 38, 3)    0           input_1[0][0]
__________________________________________________________________________________________________
```

```
conv1 (Conv2D)                  (None, 19, 19, 64)   9472        zero_padding2d_1[0][0]
__________________________________________________________________________________________________
bn_conv1 (BatchNormalization)   (None, 19, 19, 64)   256         conv1[0][0]
__________________________________________________________________________________________________
activation_1 (Activation)       (None, 19, 19, 64)   0           bn_conv1[0][0]
__________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 9, 9, 64)     0           activation_1[0][0]
__________________________________________________________________________________________________
res2a_branch2a (Conv2D)         (None, 9, 9, 64)     4160        max_pooling2d_1[0][0]
__________________________________________________________________________________________________
bn2a_branch2a (BatchNormalizati (None, 9, 9, 64)     256         res2a_branch2a[0][0]
__________________________________________________________________________________________________
activation_2 (Activation)       (None, 9, 9, 64)     0           bn2a_branch2a[0][0]
__________________________________________________________________________________________________
res2a_branch2b (Conv2D)         (None, 9, 9, 64)     36928       activation_2[0][0]
__________________________________________________________________________________________________
bn2a_branch2b (BatchNormalizati (None, 9, 9, 64)     256         res2a_branch2b[0][0]
__________________________________________________________________________________________________
activation_3 (Activation)       (None, 9, 9, 64)     0           bn2a_branch2b[0][0]
__________________________________________________________________________________________________
res2a_branch1 (Conv2D)          (None, 9, 9, 256)    16640       max_pooling2d_1[0][0]
__________________________________________________________________________________________________
res2a_branch2c (Conv2D)         (None, 9, 9, 256)    16640       activation_3[0][0]
__________________________________________________________________________________________________
bn2a_branch1 (BatchNormalizatio (None, 9, 9, 256)    1024        res2a_branch1[0][0]
__________________________________________________________________________________________________
bn2a_branch2c (BatchNormalizati (None, 9, 9, 256)    1024        res2a_branch2c[0][0]
__________________________________________________________________________________________________
add_1 (Add)                     (None, 9, 9, 256)    0           bn2a_branch1[0][0]
                                                                 bn2a_branch2c[0][0]
__________________________________________________________________________________________________
activation_4 (Activation)       (None, 9, 9, 256)    0           add_1[0][0]
__________________________________________________________________________________________________
res2b_branch2a (Conv2D)         (None, 9, 9, 64)     16448       activation_4[0][0]
__________________________________________________________________________________________________
bn2b_branch2a (BatchNormalizati (None, 9, 9, 64)     256         res2b_branch2a[0][0]
__________________________________________________________________________________________________
activation_5 (Activation)       (None, 9, 9, 64)     0           bn2b_branch2a[0][0]
__________________________________________________________________________________________________
res2b_branch2b (Conv2D)         (None, 9, 9, 64)     36928       activation_5[0][0]
__________________________________________________________________________________________________
bn2b_branch2b (BatchNormalizati (None, 9, 9, 64)     256         res2b_branch2b[0][0]
__________________________________________________________________________________________________
activation_6 (Activation)       (None, 9, 9, 64)     0           bn2b_branch2b[0][0]
__________________________________________________________________________________________________
res2b_branch2c (Conv2D)         (None, 9, 9, 256)    16640       activation_6[0][0]
__________________________________________________________________________________________________
bn2b_branch2c (BatchNormalizati (None, 9, 9, 256)    1024        res2b_branch2c[0][0]
__________________________________________________________________________________________________
add_2 (Add)                     (None, 9, 9, 256)    0           activation_4[0][0]
                                                                 bn2b_branch2c[0][0]
__________________________________________________________________________________________________
activation_7 (Activation)       (None, 9, 9, 256)    0           add_2[0][0]
__________________________________________________________________________________________________
res2c_branch2a (Conv2D)         (None, 9, 9, 64)     16448       activation_7[0][0]
__________________________________________________________________________________________________
bn2c_branch2a (BatchNormalizati (None, 9, 9, 64)     256         res2c_branch2a[0][0]
__________________________________________________________________________________________________
activation_8 (Activation)       (None, 9, 9, 64)     0           bn2c_branch2a[0][0]
__________________________________________________________________________________________________
res2c_branch2b (Conv2D)         (None, 9, 9, 64)     36928       activation_8[0][0]
__________________________________________________________________________________________________
bn2c_branch2b (BatchNormalizati (None, 9, 9, 64)     256         res2c_branch2b[0][0]
__________________________________________________________________________________________________
activation_9 (Activation)       (None, 9, 9, 64)     0           bn2c_branch2b[0][0]
__________________________________________________________________________________________________
res2c_branch2c (Conv2D)         (None, 9, 9, 256)    16640       activation_9[0][0]
__________________________________________________________________________________________________
bn2c_branch2c (BatchNormalizati (None, 9, 9, 256)    1024        res2c_branch2c[0][0]
__________________________________________________________________________________________________
add_3 (Add)                     (None, 9, 9, 256)    0           activation_7[0][0]
                                                                 bn2c_branch2c[0][0]
__________________________________________________________________________________________________
activation_10 (Activation)      (None, 9, 9, 256)    0           add_3[0][0]
__________________________________________________________________________________________________
res3a_branch2a (Conv2D)         (None, 5, 5, 128)    32896       activation_10[0][0]
__________________________________________________________________________________________________
bn3a_branch2a (BatchNormalizati (None, 5, 5, 128)    512         res3a_branch2a[0][0]
__________________________________________________________________________________________________
activation_11 (Activation)      (None, 5, 5, 128)    0           bn3a_branch2a[0][0]
__________________________________________________________________________________________________
res3a_branch2b (Conv2D)         (None, 5, 5, 128)    147584      activation_11[0][0]
__________________________________________________________________________________________________
bn3a_branch2b (BatchNormalizati (None, 5, 5, 128)    512         res3a_branch2b[0][0]
__________________________________________________________________________________________________
activation_12 (Activation)      (None, 5, 5, 128)    0           bn3a_branch2b[0][0]
__________________________________________________________________________________________________
res3a_branch1 (Conv2D)          (None, 5, 5, 512)    131584      activation_10[0][0]
__________________________________________________________________________________________________
res3a_branch2c (Conv2D)         (None, 5, 5, 512)    66048       activation_12[0][0]
__________________________________________________________________________________________________
bn3a_branch1 (BatchNormalizatio (None, 5, 5, 512)    2048        res3a_branch1[0][0]
__________________________________________________________________________________________________
bn3a_branch2c (BatchNormalizati (None, 5, 5, 512)    2048        res3a_branch2c[0][0]
__________________________________________________________________________________________________
add_4 (Add)                     (None, 5, 5, 512)    0           bn3a_branch1[0][0]
                                                                 bn3a_branch2c[0][0]
__________________________________________________________________________________________________
activation_13 (Activation)      (None, 5, 5, 512)    0           add_4[0][0]
__________________________________________________________________________________________________
res3b_branch2a (Conv2D)         (None, 5, 5, 128)    65664       activation_13[0][0]
__________________________________________________________________________________________________
bn3b_branch2a (BatchNormalizati (None, 5, 5, 128)    512         res3b_branch2a[0][0]
__________________________________________________________________________________________________
activation_14 (Activation)      (None, 5, 5, 128)    0           bn3b_branch2a[0][0]
__________________________________________________________________________________________________
res3b_branch2b (Conv2D)         (None, 5, 5, 128)    147584      activation_14[0][0]
__________________________________________________________________________________________________
bn3b_branch2b (BatchNormalizati (None, 5, 5, 128)    512         res3b_branch2b[0][0]
__________________________________________________________________________________________________
activation_15 (Activation)      (None, 5, 5, 128)    0           bn3b_branch2b[0][0]
__________________________________________________________________________________________________
res3b_branch2c (Conv2D)         (None, 5, 5, 512)    66048       activation_15[0][0]
__________________________________________________________________________________________________
bn3b_branch2c (BatchNormalizati (None, 5, 5, 512)    2048        res3b_branch2c[0][0]
__________________________________________________________________________________________________
add_5 (Add)                     (None, 5, 5, 512)    0           activation_13[0][0]
                                                                 bn3b_branch2c[0][0]
__________________________________________________________________________________________________
activation_16 (Activation)      (None, 5, 5, 512)    0           add_5[0][0]
__________________________________________________________________________________________________
res3c_branch2a (Conv2D)         (None, 5, 5, 128)    65664       activation_16[0][0]
__________________________________________________________________________________________________
bn3c_branch2a (BatchNormalizati (None, 5, 5, 128)    512         res3c_branch2a[0][0]
__________________________________________________________________________________________________
activation_17 (Activation)      (None, 5, 5, 128)    0           bn3c_branch2a[0][0]
__________________________________________________________________________________________________
res3c_branch2b (Conv2D)         (None, 5, 5, 128)    147584      activation_17[0][0]
__________________________________________________________________________________________________
bn3c_branch2b (BatchNormalizati (None, 5, 5, 128)    512         res3c_branch2b[0][0]
__________________________________________________________________________________________________
activation_18 (Activation)      (None, 5, 5, 128)    0           bn3c_branch2b[0][0]
__________________________________________________________________________________________________
res3c_branch2c (Conv2D)         (None, 5, 5, 512)    66048       activation_18[0][0]
__________________________________________________________________________________________________
bn3c_branch2c (BatchNormalizati (None, 5, 5, 512)    2048        res3c_branch2c[0][0]
__________________________________________________________________________________________________
add_6 (Add)                     (None, 5, 5, 512)    0           activation_16[0][0]
```

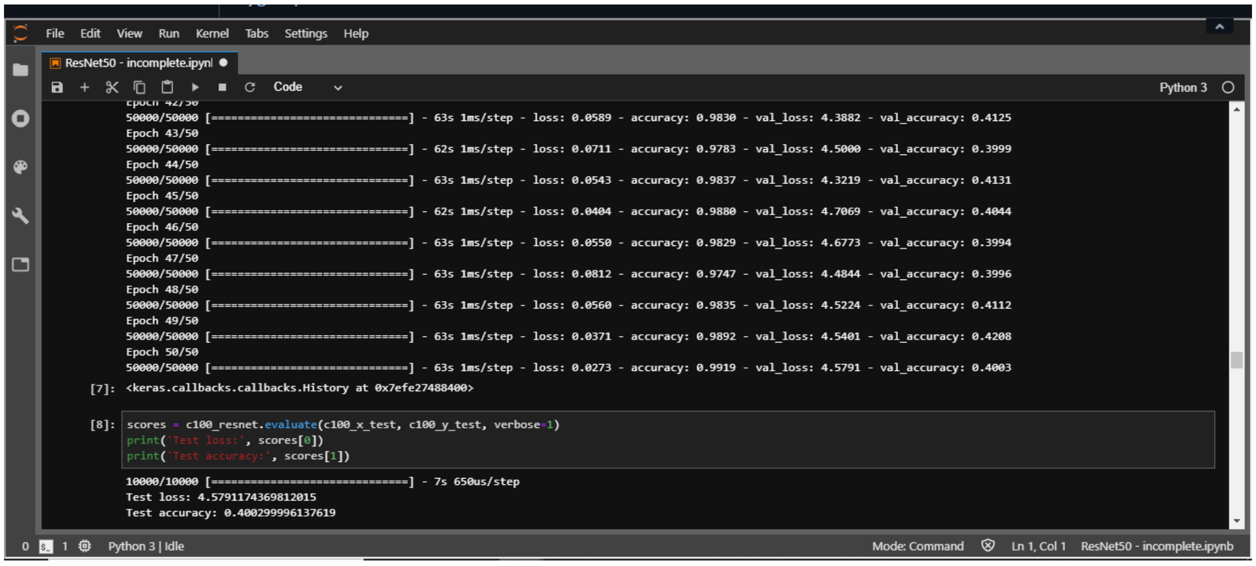
```
                                                                  bn3c_branch2c[0][0]
__________________________________________________________________________________________________
activation_19 (Activation)      (None, 5, 5, 512)    0            add_6[0][0]
__________________________________________________________________________________________________
res3d_branch2a (Conv2D)         (None, 5, 5, 128)    65664        activation_19[0][0]
__________________________________________________________________________________________________
bn3d_branch2a (BatchNormalizati (None, 5, 5, 128)    512          res3d_branch2a[0][0]
__________________________________________________________________________________________________
activation_20 (Activation)      (None, 5, 5, 128)    0            bn3d_branch2a[0][0]
__________________________________________________________________________________________________
res3d_branch2b (Conv2D)         (None, 5, 5, 128)    147584       activation_20[0][0]
__________________________________________________________________________________________________
bn3d_branch2b (BatchNormalizati (None, 5, 5, 128)    512          res3d_branch2b[0][0]
__________________________________________________________________________________________________
activation_21 (Activation)      (None, 5, 5, 128)    0            bn3d_branch2b[0][0]
__________________________________________________________________________________________________
res3d_branch2c (Conv2D)         (None, 5, 5, 512)    66048        activation_21[0][0]
__________________________________________________________________________________________________
bn3d_branch2c (BatchNormalizati (None, 5, 5, 512)    2048         res3d_branch2c[0][0]
__________________________________________________________________________________________________
add_7 (Add)                     (None, 5, 5, 512)    0            activation_19[0][0]
                                                                  bn3d_branch2c[0][0]
__________________________________________________________________________________________________
activation_22 (Activation)      (None, 5, 5, 512)    0            add_7[0][0]
__________________________________________________________________________________________________
res4a_branch2a (Conv2D)         (None, 3, 3, 256)    131328       activation_22[0][0]
__________________________________________________________________________________________________
bn4a_branch2a (BatchNormalizati (None, 3, 3, 256)    1024         res4a_branch2a[0][0]
__________________________________________________________________________________________________
activation_23 (Activation)      (None, 3, 3, 256)    0            bn4a_branch2a[0][0]
__________________________________________________________________________________________________
res4a_branch2b (Conv2D)         (None, 3, 3, 256)    590080       activation_23[0][0]
__________________________________________________________________________________________________
bn4a_branch2b (BatchNormalizati (None, 3, 3, 256)    1024         res4a_branch2b[0][0]
__________________________________________________________________________________________________
activation_24 (Activation)      (None, 3, 3, 256)    0            bn4a_branch2b[0][0]
__________________________________________________________________________________________________
res4a_branch1 (Conv2D)          (None, 3, 3, 1024)   525312       activation_22[0][0]
__________________________________________________________________________________________________
res4a_branch2c (Conv2D)         (None, 3, 3, 1024)   263168       activation_24[0][0]
__________________________________________________________________________________________________
bn4a_branch1 (BatchNormalizatio (None, 3, 3, 1024)   4096         res4a_branch1[0][0]
__________________________________________________________________________________________________
bn4a_branch2c (BatchNormalizati (None, 3, 3, 1024)   4096         res4a_branch2c[0][0]
__________________________________________________________________________________________________
add_8 (Add)                     (None, 3, 3, 1024)   0            bn4a_branch1[0][0]
                                                                  bn4a_branch2c[0][0]
__________________________________________________________________________________________________
activation_25 (Activation)      (None, 3, 3, 1024)   0            add_8[0][0]
__________________________________________________________________________________________________
res4b_branch2a (Conv2D)         (None, 3, 3, 256)    262400       activation_25[0][0]
__________________________________________________________________________________________________
bn4b_branch2a (BatchNormalizati (None, 3, 3, 256)    1024         res4b_branch2a[0][0]
__________________________________________________________________________________________________
activation_26 (Activation)      (None, 3, 3, 256)    0            bn4b_branch2a[0][0]
__________________________________________________________________________________________________
res4b_branch2b (Conv2D)         (None, 3, 3, 256)    590080       activation_26[0][0]
__________________________________________________________________________________________________
bn4b_branch2b (BatchNormalizati (None, 3, 3, 256)    1024         res4b_branch2b[0][0]
__________________________________________________________________________________________________
activation_27 (Activation)      (None, 3, 3, 256)    0            bn4b_branch2b[0][0]
__________________________________________________________________________________________________
res4b_branch2c (Conv2D)         (None, 3, 3, 1024)   263168       activation_27[0][0]
__________________________________________________________________________________________________
bn4b_branch2c (BatchNormalizati (None, 3, 3, 1024)   4096         res4b_branch2c[0][0]
__________________________________________________________________________________________________
add_9 (Add)                     (None, 3, 3, 1024)   0            activation_25[0][0]
                                                                  bn4b_branch2c[0][0]
__________________________________________________________________________________________________
```

```
activation_28 (Activation)      (None, 3, 3, 1024)   0           add_9[0][0]
__________________________________________________________________________________________________
res4c_branch2a (Conv2D)         (None, 3, 3, 256)    262400      activation_28[0][0]
__________________________________________________________________________________________________
bn4c_branch2a (BatchNormalizati (None, 3, 3, 256)    1024        res4c_branch2a[0][0]
__________________________________________________________________________________________________
activation_29 (Activation)      (None, 3, 3, 256)    0           bn4c_branch2a[0][0]
__________________________________________________________________________________________________
res4c_branch2b (Conv2D)         (None, 3, 3, 256)    590080      activation_29[0][0]
__________________________________________________________________________________________________
bn4c_branch2b (BatchNormalizati (None, 3, 3, 256)    1024        res4c_branch2b[0][0]
__________________________________________________________________________________________________
activation_30 (Activation)      (None, 3, 3, 256)    0           bn4c_branch2b[0][0]
__________________________________________________________________________________________________
res4c_branch2c (Conv2D)         (None, 3, 3, 1024)   263168      activation_30[0][0]
__________________________________________________________________________________________________
bn4c_branch2c (BatchNormalizati (None, 3, 3, 1024)   4096        res4c_branch2c[0][0]
__________________________________________________________________________________________________
add_10 (Add)                    (None, 3, 3, 1024)   0           activation_28[0][0]
                                                                 bn4c_branch2c[0][0]
__________________________________________________________________________________________________
activation_31 (Activation)      (None, 3, 3, 1024)   0           add_10[0][0]
__________________________________________________________________________________________________
res4d_branch2a (Conv2D)         (None, 3, 3, 256)    262400      activation_31[0][0]
__________________________________________________________________________________________________
bn4d_branch2a (BatchNormalizati (None, 3, 3, 256)    1024        res4d_branch2a[0][0]
__________________________________________________________________________________________________
activation_32 (Activation)      (None, 3, 3, 256)    0           bn4d_branch2a[0][0]
__________________________________________________________________________________________________
res4d_branch2b (Conv2D)         (None, 3, 3, 256)    590080      activation_32[0][0]
__________________________________________________________________________________________________
bn4d_branch2b (BatchNormalizati (None, 3, 3, 256)    1024        res4d_branch2b[0][0]
__________________________________________________________________________________________________
activation_33 (Activation)      (None, 3, 3, 256)    0           bn4d_branch2b[0][0]
__________________________________________________________________________________________________
res4d_branch2c (Conv2D)         (None, 3, 3, 1024)   263168      activation_33[0][0]
__________________________________________________________________________________________________
bn4d_branch2c (BatchNormalizati (None, 3, 3, 1024)   4096        res4d_branch2c[0][0]
__________________________________________________________________________________________________
add_11 (Add)                    (None, 3, 3, 1024)   0           activation_31[0][0]
                                                                 bn4d_branch2c[0][0]
__________________________________________________________________________________________________
activation_34 (Activation)      (None, 3, 3, 1024)   0           add_11[0][0]
__________________________________________________________________________________________________
res4e_branch2a (Conv2D)         (None, 3, 3, 256)    262400      activation_34[0][0]
__________________________________________________________________________________________________
bn4e_branch2a (BatchNormalizati (None, 3, 3, 256)    1024        res4e_branch2a[0][0]
__________________________________________________________________________________________________
activation_35 (Activation)      (None, 3, 3, 256)    0           bn4e_branch2a[0][0]
__________________________________________________________________________________________________
res4e_branch2b (Conv2D)         (None, 3, 3, 256)    590080      activation_35[0][0]
__________________________________________________________________________________________________
bn4e_branch2b (BatchNormalizati (None, 3, 3, 256)    1024        res4e_branch2b[0][0]
__________________________________________________________________________________________________
activation_36 (Activation)      (None, 3, 3, 256)    0           bn4e_branch2b[0][0]
__________________________________________________________________________________________________
res4e_branch2c (Conv2D)         (None, 3, 3, 1024)   263168      activation_36[0][0]
__________________________________________________________________________________________________
bn4e_branch2c (BatchNormalizati (None, 3, 3, 1024)   4096        res4e_branch2c[0][0]
__________________________________________________________________________________________________
add_12 (Add)                    (None, 3, 3, 1024)   0           activation_34[0][0]
                                                                 bn4e_branch2c[0][0]
__________________________________________________________________________________________________
activation_37 (Activation)      (None, 3, 3, 1024)   0           add_12[0][0]
__________________________________________________________________________________________________
res4f_branch2a (Conv2D)         (None, 3, 3, 256)    262400      activation_37[0][0]
__________________________________________________________________________________________________
bn4f_branch2a (BatchNormalizati (None, 3, 3, 256)    1024        res4f_branch2a[0][0]
__________________________________________________________________________________________________
```

```
activation_38 (Activation)      (None, 3, 3, 256)    0           bn4f_branch2a[0][0]
__________________________________________________________________________________________________
res4f_branch2b (Conv2D)         (None, 3, 3, 256)    590080      activation_38[0][0]
__________________________________________________________________________________________________
bn4f_branch2b (BatchNormalizati (None, 3, 3, 256)    1024        res4f_branch2b[0][0]
__________________________________________________________________________________________________
activation_39 (Activation)      (None, 3, 3, 256)    0           bn4f_branch2b[0][0]
__________________________________________________________________________________________________
res4f_branch2c (Conv2D)         (None, 3, 3, 1024)   263168      activation_39[0][0]
__________________________________________________________________________________________________
bn4f_branch2c (BatchNormalizati (None, 3, 3, 1024)   4096        res4f_branch2c[0][0]
__________________________________________________________________________________________________
add_13 (Add)                    (None, 3, 3, 1024)   0           activation_37[0][0]
                                                                 bn4f_branch2c[0][0]
__________________________________________________________________________________________________
activation_40 (Activation)      (None, 3, 3, 1024)   0           add_13[0][0]
__________________________________________________________________________________________________
res5a_branch2a (Conv2D)         (None, 2, 2, 512)    524800      activation_40[0][0]
__________________________________________________________________________________________________
bn5a_branch2a (BatchNormalizati (None, 2, 2, 512)    2048        res5a_branch2a[0][0]
__________________________________________________________________________________________________
activation_41 (Activation)      (None, 2, 2, 512)    0           bn5a_branch2a[0][0]
__________________________________________________________________________________________________
res5a_branch2b (Conv2D)         (None, 2, 2, 512)    2359808     activation_41[0][0]
__________________________________________________________________________________________________
bn5a_branch2b (BatchNormalizati (None, 2, 2, 512)    2048        res5a_branch2b[0][0]
__________________________________________________________________________________________________
activation_42 (Activation)      (None, 2, 2, 512)    0           bn5a_branch2b[0][0]
__________________________________________________________________________________________________
res5a_branch1 (Conv2D)          (None, 2, 2, 2048)   2099200     activation_40[0][0]
__________________________________________________________________________________________________
res5a_branch2c (Conv2D)         (None, 2, 2, 2048)   1050624     activation_42[0][0]
__________________________________________________________________________________________________
bn5a_branch1 (BatchNormalizatio (None, 2, 2, 2048)   8192        res5a_branch1[0][0]
__________________________________________________________________________________________________
bn5a_branch2c (BatchNormalizati (None, 2, 2, 2048)   8192        res5a_branch2c[0][0]
__________________________________________________________________________________________________
add_14 (Add)                    (None, 2, 2, 2048)   0           bn5a_branch1[0][0]
                                                                 bn5a_branch2c[0][0]
__________________________________________________________________________________________________
activation_43 (Activation)      (None, 2, 2, 2048)   0           add_14[0][0]
__________________________________________________________________________________________________
res5b_branch2a (Conv2D)         (None, 2, 2, 512)    1049088     activation_43[0][0]
__________________________________________________________________________________________________
bn5b_branch2a (BatchNormalizati (None, 2, 2, 512)    2048        res5b_branch2a[0][0]
__________________________________________________________________________________________________
activation_44 (Activation)      (None, 2, 2, 512)    0           bn5b_branch2a[0][0]
__________________________________________________________________________________________________
res5b_branch2b (Conv2D)         (None, 2, 2, 512)    2359808     activation_44[0][0]
__________________________________________________________________________________________________
bn5b_branch2b (BatchNormalizati (None, 2, 2, 512)    2048        res5b_branch2b[0][0]
__________________________________________________________________________________________________
activation_45 (Activation)      (None, 2, 2, 512)    0           bn5b_branch2b[0][0]
__________________________________________________________________________________________________
res5b_branch2c (Conv2D)         (None, 2, 2, 2048)   1050624     activation_45[0][0]
__________________________________________________________________________________________________
bn5b_branch2c (BatchNormalizati (None, 2, 2, 2048)   8192        res5b_branch2c[0][0]
__________________________________________________________________________________________________
add_15 (Add)                    (None, 2, 2, 2048)   0           activation_43[0][0]
                                                                 bn5b_branch2c[0][0]
__________________________________________________________________________________________________
activation_46 (Activation)      (None, 2, 2, 2048)   0           add_15[0][0]
__________________________________________________________________________________________________
res5c_branch2a (Conv2D)         (None, 2, 2, 512)    1049088     activation_46[0][0]
__________________________________________________________________________________________________
bn5c_branch2a (BatchNormalizati (None, 2, 2, 512)    2048        res5c_branch2a[0][0]
__________________________________________________________________________________________________
activation_47 (Activation)      (None, 2, 2, 512)    0           bn5c_branch2a[0][0]
__________________________________________________________________________________________________
```

```
res5c_branch2b (Conv2D)         (None, 2, 2, 512)    2359808     activation_47[0][0]
__________________________________________________________________________________________________
bn5c_branch2b (BatchNormalizati (None, 2, 2, 512)    2048        res5c_branch2b[0][0]
__________________________________________________________________________________________________
activation_48 (Activation)      (None, 2, 2, 512)    0           bn5c_branch2b[0][0]
__________________________________________________________________________________________________
res5c_branch2c (Conv2D)         (None, 2, 2, 2048)   1050624     activation_48[0][0]
__________________________________________________________________________________________________
bn5c_branch2c (BatchNormalizati (None, 2, 2, 2048)   8192        res5c_branch2c[0][0]
__________________________________________________________________________________________________
add_16 (Add)                    (None, 2, 2, 2048)   0           activation_46[0][0]
                                                                 bn5c_branch2c[0][0]
__________________________________________________________________________________________________
activation_49 (Activation)      (None, 2, 2, 2048)   0           add_16[0][0]
__________________________________________________________________________________________________
average_pooling2d_1 (AveragePoo (None, 1, 1, 2048)   0           activation_49[0][0]
__________________________________________________________________________________________________
flatten_1 (Flatten)             (None, 2048)         0           average_pooling2d_1[0][0]
__________________________________________________________________________________________________
output (Dense)                  (None, 100)          204900      flatten_1[0][0]
==================================================================================================
Total params: 23,792,612
Trainable params: 23,739,492
Non-trainable params: 53,120
```

Output:



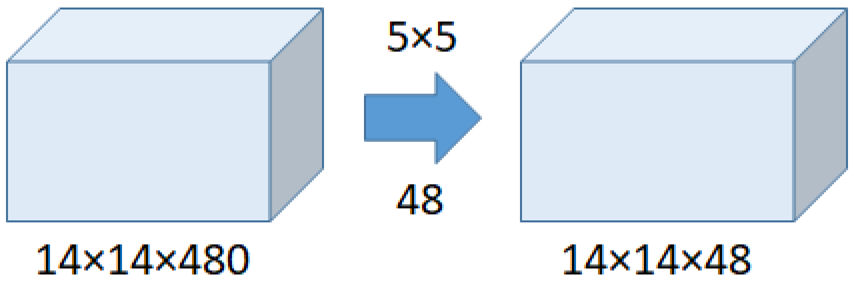
```
Epoch 42/50
50000/50000 [==============================] - 63s 1ms/step - loss: 0.0589 - accuracy: 0.9830 - val_loss: 4.3882 - val_accuracy: 0.4125
Epoch 43/50
50000/50000 [==============================] - 62s 1ms/step - loss: 0.0711 - accuracy: 0.9783 - val_loss: 4.5000 - val_accuracy: 0.3999
Epoch 44/50
50000/50000 [==============================] - 63s 1ms/step - loss: 0.0543 - accuracy: 0.9837 - val_loss: 4.3219 - val_accuracy: 0.4131
Epoch 45/50
50000/50000 [==============================] - 62s 1ms/step - loss: 0.0404 - accuracy: 0.9880 - val_loss: 4.7069 - val_accuracy: 0.4044
Epoch 46/50
50000/50000 [==============================] - 63s 1ms/step - loss: 0.0550 - accuracy: 0.9829 - val_loss: 4.6773 - val_accuracy: 0.3994
Epoch 47/50
50000/50000 [==============================] - 63s 1ms/step - loss: 0.0812 - accuracy: 0.9747 - val_loss: 4.4844 - val_accuracy: 0.3996
Epoch 48/50
50000/50000 [==============================] - 63s 1ms/step - loss: 0.0560 - accuracy: 0.9835 - val_loss: 4.5224 - val_accuracy: 0.4112
Epoch 49/50
50000/50000 [==============================] - 63s 1ms/step - loss: 0.0371 - accuracy: 0.9892 - val_loss: 4.5401 - val_accuracy: 0.4208
Epoch 50/50
50000/50000 [==============================] - 63s 1ms/step - loss: 0.0273 - accuracy: 0.9919 - val_loss: 4.5791 - val_accuracy: 0.4003
[7]: <keras.callbacks.callbacks.History at 0x7efe27488400>

[8]: scores = c100_resnet.evaluate(c100_x_test, c100_y_test, verbose=1)
     print('Test loss:', scores[0])
     print('Test accuracy:', scores[1])

10000/10000 [==============================] - 7s 650us/step
Test loss: 4.5791174369812015
Test accuracy: 0.4002999961376619
```

File Edit View Run Kernel Tabs Settings Help
ResNet50 - incomplete.ipynb
Code
Python 3
0 1 Python 3 | Idle
Mode: Command
Ln 1, Col 1
ResNet50 - incomplete.ipynb

5.GoogLeNET:

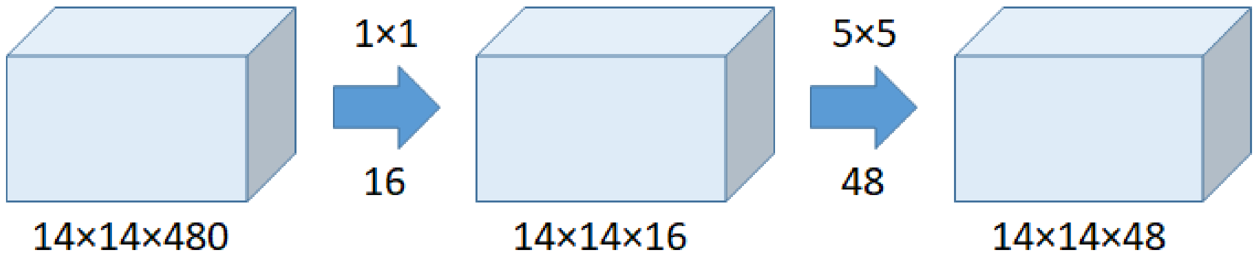
The 1×1 Convolution

The 1×1 convolution is introduced by NIN [6]. 1×1 convolution is used with ReLU. Thus, originally, NIN uses it for introducing more non-linearity to increase the representational power of the network since authors in NIN believe data is in non-linearity form. In GoogLeNet, 1×1 convolution is used as a dimension reduction module to reduce the computation. By reducing the computation bottleneck, depth and width can be increased.



Without the Use of 1×1 Convolution

Number of operations = (14×14×48)×(5×5×480) = 112.9M



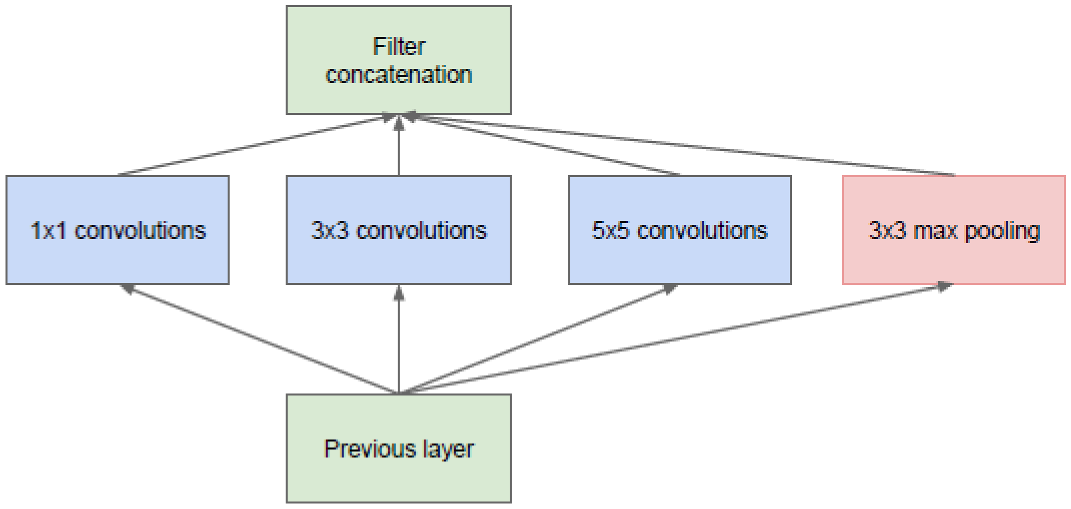
With the Use of 1×1 Convolution

Number of operations for 1×1 = (14×14×16)×(1×1×480) = 1.5M
Number of operations for 5×5 = (14×14×48)×(5×5×16) = 3.8M
Total number of operations = 1.5M + 3.8M = 5.3M

1×1 convolution can help to reduce model size which can also somehow help to reduce the overfitting problem!!

Inception Module

The inception module (naive version, without 1×1 convolution) is as below:



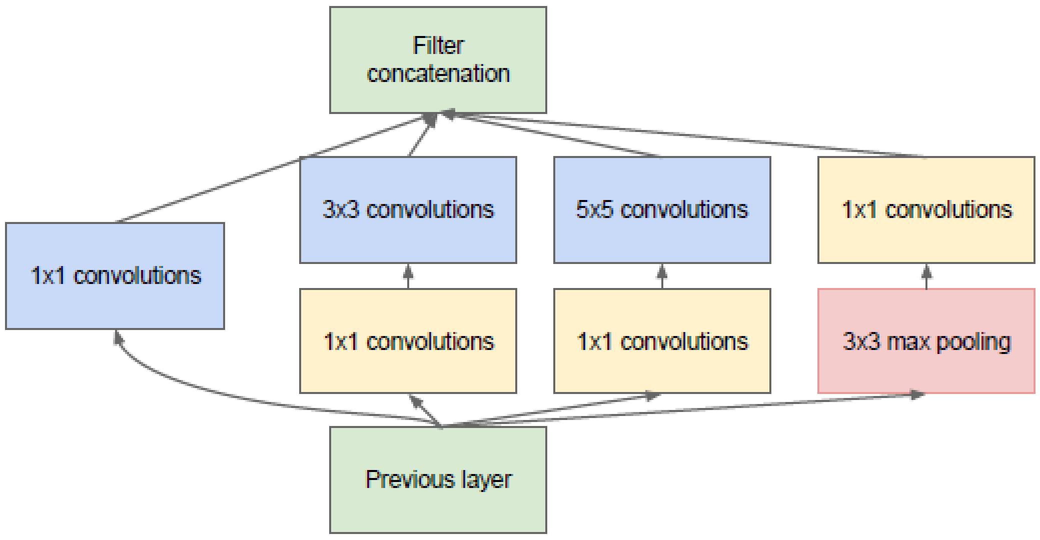
(a) Inception module, naïve version

Inception Module (Without 1×1 Convolution)

Previously, such as AlexNet, and VGGNet, conv size is fixed for each layer.

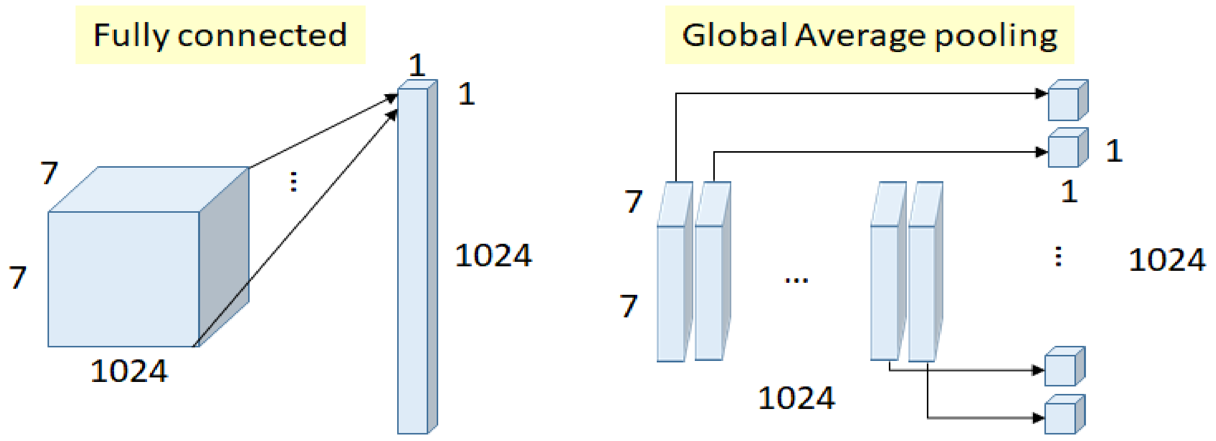
Now, 1×1 conv, 3×3 conv, 5×5 conv, and 3×3 max pooling are done altogether for the previous input, and stack together again at output. When image's coming in, different sizes of convolutions as well as max pooling are tried. Then different kinds of features are extracted.

After that, all feature maps at different paths are concatenated together as the input of the next module.



1×1 convolution is inserted into the inception module for dimension reduction!

Global Average Pooling



Fully Connected Layer VS Global Average Pooling

Number of weights (connections) above = 7×7×1024×1024 = 51.3M

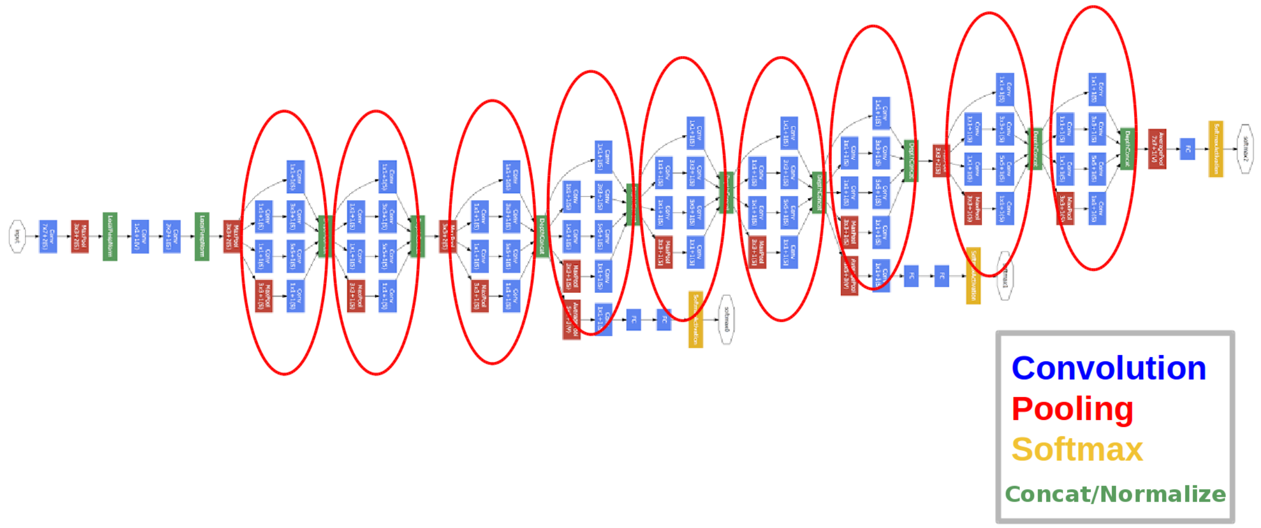
In GoogLeNet, global average pooling is used nearly at the end of network by averaging each feature map from 7×7 to 1×1, as in the figure above.

Number of weights = 0

And authors found that a move from FC layers to average pooling improved the top-1 accuracy by about 0.6%.

This is less prone to overfitting.

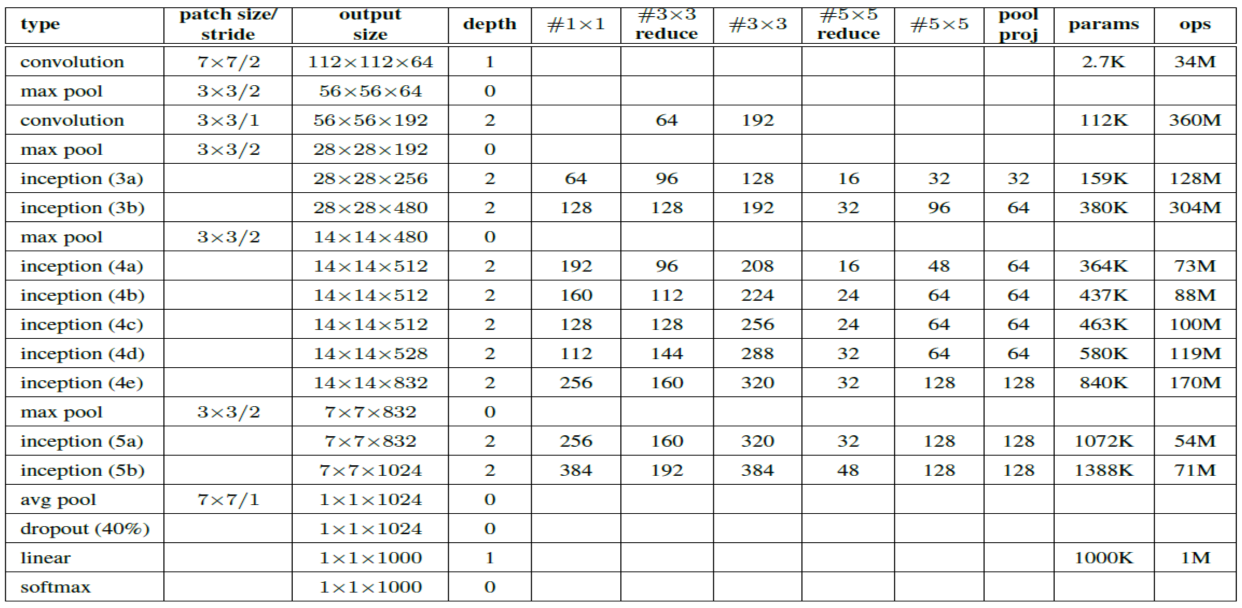
Overall Architecture



GoogLeNet Network (From Left to Right)

There are 22 layers in total

It is already a very deep model compared with previous AlexNet, ZFNet and VGGNet. (But not so deep compared with ResNet invented afterwards.) And we can see that there are numerous inception modules connected together to go deeper.



type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Details about Parameters of Each Layer in GoogLeNet Network (From Top to Bottom)

Auxiliary Classifiers for Training

As we can see there are some intermediate softmax branches at the middle, they are used for training only. These branches are auxiliary classifiers which consist of:

5×5 Average Pooling (Stride 3)
1×1 Conv (128 filters)
1024 FC
1000 FC
Softmax

//CODE

```
def inception_module(X, f1, f2_in, f2_out, f3_in, f3_out, f4_out, stage, block):

    name_sb = str(stage) + str(block)

    # 1x1 conv
    conv1 = Conv2D(f1, (1,1), padding='same', activation='relu', name = 'conv_tower1_' + name_sb)(X)
    # 3x3 conv
    conv3 = Conv2D(f2_in, (1,1), padding='same', activation='relu', name = 'conv_tower2_1_' + name_sb)(X)
    conv3 = Conv2D(f2_out, (3,3), padding='same', activation='relu', name = 'conv_tower2_2' + name_sb)(con
v3)
    # 5x5 conv
    conv5 = Conv2D(f3_in, (1,1), padding='same', activation='relu', name = 'conv_tower3_1_' + name_sb)(X)
    conv5 = Conv2D(f3_out, (5,5), padding='same', activation='relu', name = 'conv_tower3_2_' + name_sb)(co
nv5)
    # 3x3 max pooling
    pool = MaxPooling2D((3,3), strides=(1,1), padding='same', name = 'mpool_tower4_' + name_sb)(X)
    pool = Conv2D(f4_out, (1,1), padding='same', activation='relu', name = 'conv_tower4_' + name_sb)(pool)

    # concatenate filters
    layer_out = concatenate([conv1, conv3, conv5, pool], axis=-1)

    return layer_out

def auxiliary(x, classes, name=None):
    layer = AveragePooling2D(pool_size=(5,5), strides=3, padding='valid')(x)
    layer = Conv2D(filters=128, kernel_size=(1,1), strides=1, padding='same', activation='relu')(layer)
    layer = Flatten()(layer)
    layer = Dense(units=256, activation='relu')(layer)
    layer = Dropout(0.4)(layer)
    layer = Dense(units=classes, activation='softmax', name=name)(layer)

    return layer


def googlenet(input_shape, classes):

    X_input = Input(input_shape)

    # Zero-Padding
    X = ZeroPadding2D((3, 3))(X_input)

    # Stage 1
    X = Conv2D(32, (3, 3), strides = (2, 2), name = 'conv_pre_inception_1', kernel_initializer = glorot_un
iform(seed=0))(X)
    X = BatchNormalization(axis = 3, name = 'bn_pre_inception_1')(X)
    X = Activation('relu')(X)
    X = MaxPooling2D((2, 2), strides=(1, 1), name = 'mpool_pre_inception_1')(X)

    # Stage 2
    X = Conv2D(192, (3, 3), strides = (1, 1), name = 'conv_pre_inception_2', kernel_initializer = glorot_u
niform(seed=0))(X)
    X = BatchNormalization(axis = 3, name = 'bn_pre_inception_2')(X)
    X = Activation('relu')(X)
    X = Conv2D(192, (3, 3), strides = (1, 1), name = 'conv_pre_inception_3', kernel_initializer = glorot_u
niform(seed=0))(X)
    X = BatchNormalization(axis = 3, name = 'bn_pre_inception_3')(X)
    X = Activation('relu')(X)
    X = MaxPooling2D((2, 2), strides=(1, 1), name = 'mpool_pre_inception_2')(X)

    # Stage 3
    X = inception_module(X, 64, 96, 128, 16, 32, 32, 3, 'a')
    X = inception_module(X, 128, 128, 192, 32, 96, 64, 3, 'b')
    X = MaxPooling2D((2, 2), strides=(1, 1), name = 'mpool_inception_3')(X)

    # Stage 4
    X = inception_module(X, 192, 96, 208, 16, 48, 64, 4, 'a')
    aux1  = auxiliary(X, classes, name='aux1')
    X = inception_module(X, 160, 112, 224, 24, 64, 64, 4, 'b')
    X = inception_module(X, 128, 128, 256, 24, 24, 64, 4, 'c')
    X = inception_module(X, 112, 144, 288, 32, 64, 64, 4, 'd')
    aux2  = auxiliary(X, classes, name='aux2')
    X = inception_module(X, 256, 120, 320, 32, 128, 128, 4, 'e')
    X = MaxPooling2D((2, 2), strides=(1, 1), name = 'mpool_inception_4')(X)

    # Stage 5
    X = inception_module(X, 256, 160, 320, 32, 128, 128, 5, 'a')
    X = inception_module(X, 384, 192, 384, 48, 128, 128, 5, 'b')
    X = AveragePooling2D((7, 7), strides=(1, 1), name = 'apool_inception_5')(X)

    # Stage 6
    X = Dropout(0.4)(X)
    X = Flatten()(X)
    X = Dense(1000, activation='relu', kernel_initializer = glorot_uniform(seed=0))(X)
    main = Dense(classes, activation='softmax', name='main', kernel_initializer = glorot_uniform(seed=0))(
X)

    aux1 = keras.layers.Lambda(lambda x: x * 0.3)(aux1)
    aux2 = keras.layers.Lambda(lambda x: x * 0.3)(aux2)
    totalloss = keras.layers.Add()([aux1, aux2, main])

    inceptionv1 = Model(inputs = X_input, outputs = [totalloss], name='InceptionV1')
    opt = keras.optimizers.Adam(learning_rate=0.0001)
    inceptionv1.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])

    return inceptionv1
```

Ciphar10:

```
(c10_x_train_orig, c10_y_train_orig), (c10_x_test_orig, c10_y_test_orig) = cifar10.load_data()

# Pre-processing X
c10_x_train = c10_x_train_orig.astype('float32')
c10_x_test = c10_x_test_orig.astype('float32')
c10_x_train = c10_x_train/255
c10_x_test = c10_x_test/255

# Hyper-parameters
c10_epochs = 50
c10_batch_size = 128
c10_num_classes = 10
c10_input_shape = c10_x_train.shape[1:]

# Pre-processing Y
c10_y_train = keras.utils.to_categorical(c10_y_train_orig, c10_num_classes)
c10_y_test = keras.utils.to_categorical(c10_y_test_orig, c10_num_classes)
```

```
print(c10_x_train.shape, c10_x_test.shape, c10_y_train.shape, c10_y_test.shape)
(50000, 32, 32, 3) (10000, 32, 32, 3) (50000, 10) (10000, 10)
```

```
c10_googlenet = googlenet(c10_input_shape, c10_num_classes)
c10_googlenet.summary()
```

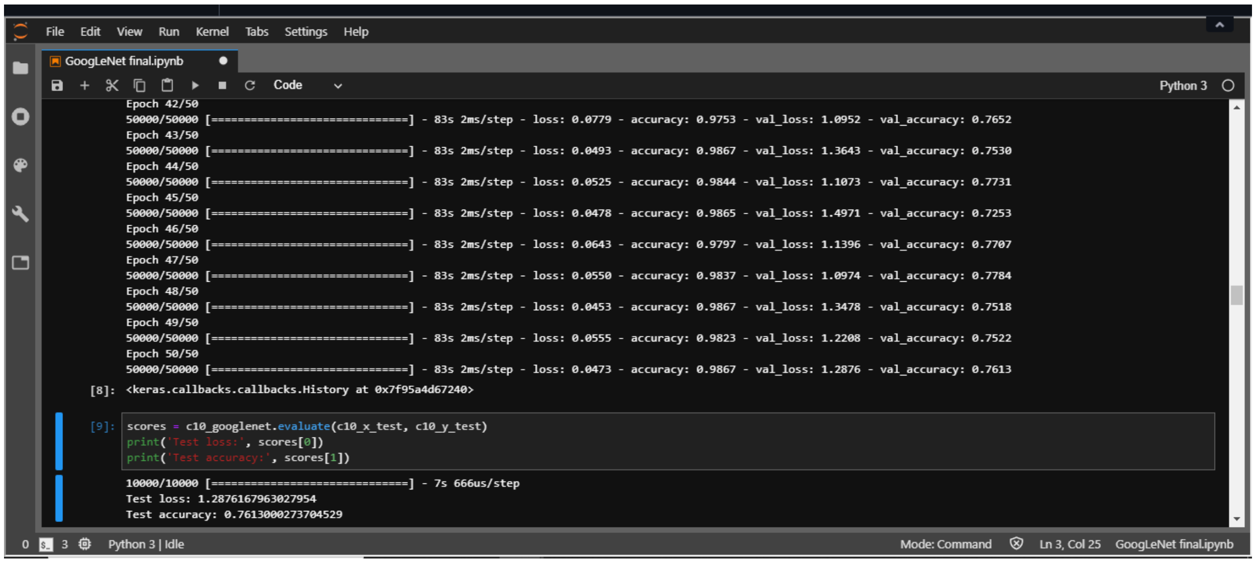
```
Model: "InceptionV1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            (None, 32, 32, 3)    0
__________________________________________________________________________________________________
zero_padding2d_1 (ZeroPadding2D (None, 38, 38, 3)    0           input_1[0][0]
__________________________________________________________________________________________________
conv_pre_inception_1 (Conv2D)   (None, 18, 18, 32)   896         zero_padding2d_1[0][0]
__________________________________________________________________________________________________
bn_pre_inception_1 (BatchNormal (None, 18, 18, 32)   128         conv_pre_inception_1[0][0]
__________________________________________________________________________________________________
activation_1 (Activation)       (None, 18, 18, 32)   0           bn_pre_inception_1[0][0]
__________________________________________________________________________________________________
mpool_pre_inception_1 (MaxPooli (None, 17, 17, 32)   0           activation_1[0][0]
__________________________________________________________________________________________________
conv_pre_inception_2 (Conv2D)   (None, 15, 15, 192)  55488       mpool_pre_inception_1[0][0]
__________________________________________________________________________________________________
bn_pre_inception_2 (BatchNormal (None, 15, 15, 192)  768         conv_pre_inception_2[0][0]
__________________________________________________________________________________________________
activation_2 (Activation)       (None, 15, 15, 192)  0           bn_pre_inception_2[0][0]
__________________________________________________________________________________________________
conv_pre_inception_3 (Conv2D)   (None, 13, 13, 192)  331968      activation_2[0][0]
__________________________________________________________________________________________________
bn_pre_inception_3 (BatchNormal (None, 13, 13, 192)  768         conv_pre_inception_3[0][0]
__________________________________________________________________________________________________
activation_3 (Activation)       (None, 13, 13, 192)  0           bn_pre_inception_3[0][0]
__________________________________________________________________________________________________
mpool_pre_inception_2 (MaxPooli (None, 12, 12, 192)  0           activation_3[0][0]
__________________________________________________________________________________________________
conv_tower2_1_3a (Conv2D)       (None, 12, 12, 96)   18528       mpool_pre_inception_2[0][0]
__________________________________________________________________________________________________
conv_tower3_1_3a (Conv2D)       (None, 12, 12, 16)   3088        mpool_pre_inception_2[0][0]
__________________________________________________________________________________________________
mpool_tower4_3a (MaxPooling2D)  (None, 12, 12, 192)  0           mpool_pre_inception_2[0][0]
__________________________________________________________________________________________________
conv_tower1_3a (Conv2D)         (None, 12, 12, 64)   12352       mpool_pre_inception_2[0][0]
__________________________________________________________________________________________________
conv_tower2_23a (Conv2D)        (None, 12, 12, 128)  110720      conv_tower2_1_3a[0][0]
__________________________________________________________________________________________________
conv_tower3_2_3a (Conv2D)       (None, 12, 12, 32)   12832       conv_tower3_1_3a[0][0]
__________________________________________________________________________________________________
conv_tower4_3a (Conv2D)         (None, 12, 12, 32)   6176        mpool_tower4_3a[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 12, 12, 256)  0           conv_tower1_3a[0][0]
                                                                 conv_tower2_23a[0][0]
                                                                 conv_tower3_2_3a[0][0]
                                                                 conv_tower4_3a[0][0]
__________________________________________________________________________________________________
conv_tower2_1_3b (Conv2D)       (None, 12, 12, 128)  32896       concatenate_1[0][0]
__________________________________________________________________________________________________
conv_tower3_1_3b (Conv2D)       (None, 12, 12, 32)   8224        concatenate_1[0][0]
__________________________________________________________________________________________________
mpool_tower4_3b (MaxPooling2D)  (None, 12, 12, 256)  0           concatenate_1[0][0]
__________________________________________________________________________________________________
conv_tower1_3b (Conv2D)         (None, 12, 12, 128)  32896       concatenate_1[0][0]
__________________________________________________________________________________________________
conv_tower2_23b (Conv2D)        (None, 12, 12, 192)  221376      conv_tower2_1_3b[0][0]
```

```
__________________________________________________________________________________________________
conv_tower3_2_3b (Conv2D)       (None, 12, 12, 96)   76896       conv_tower3_1_3b[0][0]
__________________________________________________________________________________________________
conv_tower4_3b (Conv2D)         (None, 12, 12, 64)   16448       mpool_tower4_3b[0][0]
__________________________________________________________________________________________________
concatenate_2 (Concatenate)     (None, 12, 12, 480)  0           conv_tower1_3b[0][0]
                                                                 conv_tower2_23b[0][0]
                                                                 conv_tower3_2_3b[0][0]
                                                                 conv_tower4_3b[0][0]
__________________________________________________________________________________________________
mpool_inception_3 (MaxPooling2D (None, 11, 11, 480)  0           concatenate_2[0][0]
__________________________________________________________________________________________________
conv_tower2_1_4a (Conv2D)       (None, 11, 11, 96)   46176       mpool_inception_3[0][0]
__________________________________________________________________________________________________
conv_tower3_1_4a (Conv2D)       (None, 11, 11, 16)   7696        mpool_inception_3[0][0]
__________________________________________________________________________________________________
mpool_tower4_4a (MaxPooling2D)  (None, 11, 11, 480)  0           mpool_inception_3[0][0]
__________________________________________________________________________________________________
conv_tower1_4a (Conv2D)         (None, 11, 11, 192)  92352       mpool_inception_3[0][0]
__________________________________________________________________________________________________
conv_tower2_24a (Conv2D)        (None, 11, 11, 208)  179920      conv_tower2_1_4a[0][0]
__________________________________________________________________________________________________
conv_tower3_2_4a (Conv2D)       (None, 11, 11, 48)   19248       conv_tower3_1_4a[0][0]
__________________________________________________________________________________________________
conv_tower4_4a (Conv2D)         (None, 11, 11, 64)   30784       mpool_tower4_4a[0][0]
__________________________________________________________________________________________________
concatenate_3 (Concatenate)     (None, 11, 11, 512)  0           conv_tower1_4a[0][0]
                                                                 conv_tower2_24a[0][0]
                                                                 conv_tower3_2_4a[0][0]
                                                                 conv_tower4_4a[0][0]
__________________________________________________________________________________________________
conv_tower2_1_4b (Conv2D)       (None, 11, 11, 112)  57456       concatenate_3[0][0]
__________________________________________________________________________________________________
conv_tower3_1_4b (Conv2D)       (None, 11, 11, 24)   12312       concatenate_3[0][0]
__________________________________________________________________________________________________
mpool_tower4_4b (MaxPooling2D)  (None, 11, 11, 512)  0           concatenate_3[0][0]
__________________________________________________________________________________________________
conv_tower1_4b (Conv2D)         (None, 11, 11, 160)  82080       concatenate_3[0][0]
__________________________________________________________________________________________________
conv_tower2_24b (Conv2D)        (None, 11, 11, 224)  226016      conv_tower2_1_4b[0][0]
__________________________________________________________________________________________________
conv_tower3_2_4b (Conv2D)       (None, 11, 11, 64)   38464       conv_tower3_1_4b[0][0]
__________________________________________________________________________________________________
conv_tower4_4b (Conv2D)         (None, 11, 11, 64)   32832       mpool_tower4_4b[0][0]
__________________________________________________________________________________________________
concatenate_4 (Concatenate)     (None, 11, 11, 512)  0           conv_tower1_4b[0][0]
                                                                 conv_tower2_24b[0][0]
                                                                 conv_tower3_2_4b[0][0]
                                                                 conv_tower4_4b[0][0]
__________________________________________________________________________________________________
conv_tower2_1_4c (Conv2D)       (None, 11, 11, 128)  65664       concatenate_4[0][0]
__________________________________________________________________________________________________
conv_tower3_1_4c (Conv2D)       (None, 11, 11, 24)   12312       concatenate_4[0][0]
__________________________________________________________________________________________________
mpool_tower4_4c (MaxPooling2D)  (None, 11, 11, 512)  0           concatenate_4[0][0]
__________________________________________________________________________________________________
conv_tower1_4c (Conv2D)         (None, 11, 11, 128)  65664       concatenate_4[0][0]
__________________________________________________________________________________________________
conv_tower2_24c (Conv2D)        (None, 11, 11, 256)  295168      conv_tower2_1_4c[0][0]
__________________________________________________________________________________________________
conv_tower3_2_4c (Conv2D)       (None, 11, 11, 24)   14424       conv_tower3_1_4c[0][0]
__________________________________________________________________________________________________
conv_tower4_4c (Conv2D)         (None, 11, 11, 64)   32832       mpool_tower4_4c[0][0]
__________________________________________________________________________________________________
concatenate_5 (Concatenate)     (None, 11, 11, 472)  0           conv_tower1_4c[0][0]
                                                                 conv_tower2_24c[0][0]
                                                                 conv_tower3_2_4c[0][0]
                                                                 conv_tower4_4c[0][0]
__________________________________________________________________________________________________
```

```
conv_tower2_1_4d (Conv2D)       (None, 11, 11, 144)  68112       concatenate_5[0][0]
__________________________________________________________________________________________________
conv_tower3_1_4d (Conv2D)       (None, 11, 11, 32)   15136       concatenate_5[0][0]
__________________________________________________________________________________________________
mpool_tower4_4d (MaxPooling2D)  (None, 11, 11, 472)  0           concatenate_5[0][0]
__________________________________________________________________________________________________
conv_tower1_4d (Conv2D)         (None, 11, 11, 112)  52976       concatenate_5[0][0]
__________________________________________________________________________________________________
conv_tower2_24d (Conv2D)        (None, 11, 11, 288)  373536      conv_tower2_1_4d[0][0]
__________________________________________________________________________________________________
conv_tower3_2_4d (Conv2D)       (None, 11, 11, 64)   51264       conv_tower3_1_4d[0][0]
__________________________________________________________________________________________________
conv_tower4_4d (Conv2D)         (None, 11, 11, 64)   30272       mpool_tower4_4d[0][0]
__________________________________________________________________________________________________
concatenate_6 (Concatenate)     (None, 11, 11, 528)  0           conv_tower1_4d[0][0]
                                                                 conv_tower2_24d[0][0]
                                                                 conv_tower3_2_4d[0][0]
                                                                 conv_tower4_4d[0][0]
__________________________________________________________________________________________________
conv_tower2_1_4e (Conv2D)       (None, 11, 11, 120)  63480       concatenate_6[0][0]
__________________________________________________________________________________________________
conv_tower3_1_4e (Conv2D)       (None, 11, 11, 32)   16928       concatenate_6[0][0]
__________________________________________________________________________________________________
mpool_tower4_4e (MaxPooling2D)  (None, 11, 11, 528)  0           concatenate_6[0][0]
__________________________________________________________________________________________________
conv_tower1_4e (Conv2D)         (None, 11, 11, 256)  135424      concatenate_6[0][0]
__________________________________________________________________________________________________
conv_tower2_24e (Conv2D)        (None, 11, 11, 320)  345920      conv_tower2_1_4e[0][0]
__________________________________________________________________________________________________
conv_tower3_2_4e (Conv2D)       (None, 11, 11, 128)  102528      conv_tower3_1_4e[0][0]
__________________________________________________________________________________________________
conv_tower4_4e (Conv2D)         (None, 11, 11, 128)  67712       mpool_tower4_4e[0][0]
__________________________________________________________________________________________________
concatenate_7 (Concatenate)     (None, 11, 11, 832)  0           conv_tower1_4e[0][0]
                                                                 conv_tower2_24e[0][0]
                                                                 conv_tower3_2_4e[0][0]
                                                                 conv_tower4_4e[0][0]
__________________________________________________________________________________________________
mpool_inception_4 (MaxPooling2D (None, 10, 10, 832)  0           concatenate_7[0][0]
__________________________________________________________________________________________________
conv_tower2_1_5a (Conv2D)       (None, 10, 10, 160)  133280      mpool_inception_4[0][0]
__________________________________________________________________________________________________
conv_tower3_1_5a (Conv2D)       (None, 10, 10, 32)   26656       mpool_inception_4[0][0]
__________________________________________________________________________________________________
mpool_tower4_5a (MaxPooling2D)  (None, 10, 10, 832)  0           mpool_inception_4[0][0]
__________________________________________________________________________________________________
conv_tower1_5a (Conv2D)         (None, 10, 10, 256)  213248      mpool_inception_4[0][0]
__________________________________________________________________________________________________
conv_tower2_25a (Conv2D)        (None, 10, 10, 320)  461120      conv_tower2_1_5a[0][0]
__________________________________________________________________________________________________
conv_tower3_2_5a (Conv2D)       (None, 10, 10, 128)  102528      conv_tower3_1_5a[0][0]
__________________________________________________________________________________________________
conv_tower4_5a (Conv2D)         (None, 10, 10, 128)  106624      mpool_tower4_5a[0][0]
__________________________________________________________________________________________________
concatenate_8 (Concatenate)     (None, 10, 10, 832)  0           conv_tower1_5a[0][0]
                                                                 conv_tower2_25a[0][0]
                                                                 conv_tower3_2_5a[0][0]
                                                                 conv_tower4_5a[0][0]
__________________________________________________________________________________________________
conv_tower2_1_5b (Conv2D)       (None, 10, 10, 192)  159936      concatenate_8[0][0]
__________________________________________________________________________________________________
conv_tower3_1_5b (Conv2D)       (None, 10, 10, 48)   39984       concatenate_8[0][0]
__________________________________________________________________________________________________
mpool_tower4_5b (MaxPooling2D)  (None, 10, 10, 832)  0           concatenate_8[0][0]
__________________________________________________________________________________________________
average_pooling2d_1 (AveragePoo (None, 3, 3, 512)    0           concatenate_3[0][0]
__________________________________________________________________________________________________
average_pooling2d_2 (AveragePoo (None, 3, 3, 528)    0           concatenate_6[0][0]
__________________________________________________________________________________________________
```

```
conv_tower1_5b (Conv2D)         (None, 10, 10, 384)  319872      concatenate_8[0][0]
__________________________________________________________________________________________________
conv_tower2_25b (Conv2D)        (None, 10, 10, 384)  663936      conv_tower2_1_5b[0][0]
__________________________________________________________________________________________________
conv_tower3_2_5b (Conv2D)       (None, 10, 10, 128)  153728      conv_tower3_1_5b[0][0]
__________________________________________________________________________________________________
conv_tower4_5b (Conv2D)         (None, 10, 10, 128)  106624      mpool_tower4_5b[0][0]
__________________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 3, 3, 128)    65664       average_pooling2d_1[0][0]
__________________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, 3, 3, 128)    67712       average_pooling2d_2[0][0]
__________________________________________________________________________________________________
concatenate_9 (Concatenate)     (None, 10, 10, 1024) 0           conv_tower1_5b[0][0]
                                                                 conv_tower2_25b[0][0]
                                                                 conv_tower3_2_5b[0][0]
                                                                 conv_tower4_5b[0][0]
__________________________________________________________________________________________________
flatten_1 (Flatten)             (None, 1152)         0           conv2d_1[0][0]
__________________________________________________________________________________________________
flatten_2 (Flatten)             (None, 1152)         0           conv2d_2[0][0]
__________________________________________________________________________________________________
apool_inception_5 (AveragePooli (None, 4, 4, 1024)   0           concatenate_9[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 256)          295168      flatten_1[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 256)          295168      flatten_2[0][0]
__________________________________________________________________________________________________
dropout_3 (Dropout)             (None, 4, 4, 1024)   0           apool_inception_5[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 256)          0           dense_1[0][0]
__________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 256)          0           dense_2[0][0]
__________________________________________________________________________________________________
flatten_3 (Flatten)             (None, 16384)        0           dropout_3[0][0]
__________________________________________________________________________________________________
aux1 (Dense)                    (None, 10)           2570        dropout_1[0][0]
__________________________________________________________________________________________________
aux2 (Dense)                    (None, 10)           2570        dropout_2[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 1000)         16385000    flatten_3[0][0]
__________________________________________________________________________________________________
lambda_1 (Lambda)               (None, 10)           0           aux1[0][0]
__________________________________________________________________________________________________
lambda_2 (Lambda)               (None, 10)           0           aux2[0][0]
__________________________________________________________________________________________________
main (Dense)                    (None, 10)           10010       dense_3[0][0]
__________________________________________________________________________________________________
add_1 (Add)                     (None, 10)           0           lambda_1[0][0]
                                                                 lambda_2[0][0]
                                                                 main[0][0]
==================================================================================================
Total params: 23,188,534
Trainable params: 23,187,702
Non-trainable params: 832
```

Output:



```
Epoch 42/50
50000/50000 [==============================] - 83s 2ms/step - loss: 0.0779 - accuracy: 0.9753 - val_loss: 1.0952 - val_accuracy: 0.7652
Epoch 43/50
50000/50000 [==============================] - 83s 2ms/step - loss: 0.0493 - accuracy: 0.9867 - val_loss: 1.3643 - val_accuracy: 0.7530
Epoch 44/50
50000/50000 [==============================] - 83s 2ms/step - loss: 0.0525 - accuracy: 0.9844 - val_loss: 1.1073 - val_accuracy: 0.7731
Epoch 45/50
50000/50000 [==============================] - 83s 2ms/step - loss: 0.0478 - accuracy: 0.9865 - val_loss: 1.4971 - val_accuracy: 0.7253
Epoch 46/50
50000/50000 [==============================] - 83s 2ms/step - loss: 0.0643 - accuracy: 0.9797 - val_loss: 1.1396 - val_accuracy: 0.7707
Epoch 47/50
50000/50000 [==============================] - 83s 2ms/step - loss: 0.0550 - accuracy: 0.9837 - val_loss: 1.0974 - val_accuracy: 0.7784
Epoch 48/50
50000/50000 [==============================] - 83s 2ms/step - loss: 0.0453 - accuracy: 0.9867 - val_loss: 1.3478 - val_accuracy: 0.7518
Epoch 49/50
50000/50000 [==============================] - 83s 2ms/step - loss: 0.0555 - accuracy: 0.9823 - val_loss: 1.2208 - val_accuracy: 0.7522
Epoch 50/50
50000/50000 [==============================] - 83s 2ms/step - loss: 0.0473 - accuracy: 0.9867 - val_loss: 1.2876 - val_accuracy: 0.7613
[8]: <keras.callbacks.callbacks.History at 0x7f95a4d67240>

[9]: scores = c10_googlenet.evaluate(c10_x_test, c10_y_test)
     print('Test loss:', scores[0])
     print('Test accuracy:', scores[1])

10000/10000 [==============================] - 7s 666us/step
Test loss: 1.2876167963027954
Test accuracy: 0.7613000273704529
```

Ciphar100:

```
# Load data
(c100_x_train_orig, c100_y_train_orig), (c100_x_test_orig, c100_y_test_orig) = cifar100.load_data()

# Pre-processing X
c100_x_train = c100_x_train_orig.astype('float32')
c100_x_test = c100_x_test_orig.astype('float32')
c100_x_train = c100_x_train/255
c100_x_test = c100_x_test/255

# Hyper-parameters
c100_epochs = 50
c100_batch_size = 128
c100_num_classes = 100
c100_input_shape = c100_x_train.shape[1:]

# Pre-processing Y
c100_y_train = keras.utils.to_categorical(c100_y_train_orig, c100_num_classes)
c100_y_test = keras.utils.to_categorical(c100_y_test_orig, c100_num_classes)

print(c100_x_train.shape, c100_x_test.shape, c100_y_train.shape, c100_y_test.shape)
(50000, 32, 32, 3) (10000, 32, 32, 3) (50000, 100) (10000, 100)

c100_googlenet = googlenet(c100_input_shape, c100_num_classes)
c100_googlenet.summary()
```

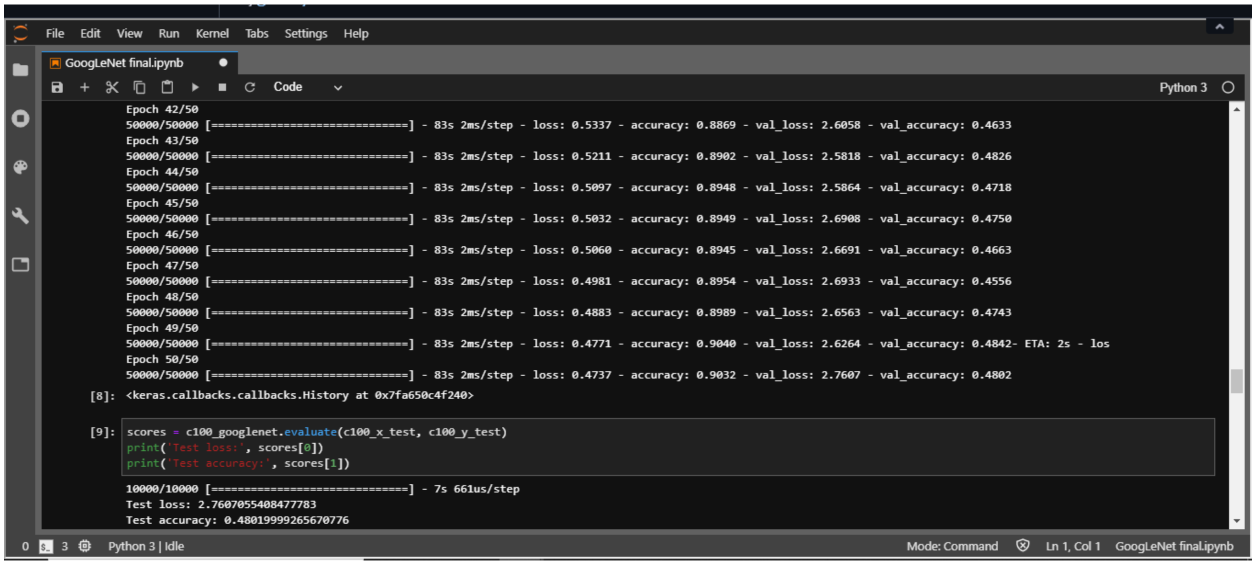
```
Model: "InceptionV1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_2 (InputLayer)            (None, 32, 32, 3)    0
__________________________________________________________________________________________________
```

```
zero_padding2d_2 (ZeroPadding2D (None, 38, 38, 3)    0           input_2[0][0]
__________________________________________________________________________________________________
conv_pre_inception_1 (Conv2D)   (None, 18, 18, 32)   896         zero_padding2d_2[0][0]
__________________________________________________________________________________________________
bn_pre_inception_1 (BatchNormal (None, 18, 18, 32)   128         conv_pre_inception_1[0][0]
__________________________________________________________________________________________________
activation_4 (Activation)       (None, 18, 18, 32)   0           bn_pre_inception_1[0][0]
__________________________________________________________________________________________________
mpool_pre_inception_1 (MaxPooli (None, 17, 17, 32)   0           activation_4[0][0]
__________________________________________________________________________________________________
conv_pre_inception_2 (Conv2D)   (None, 15, 15, 192)  55488       mpool_pre_inception_1[0][0]
__________________________________________________________________________________________________
bn_pre_inception_2 (BatchNormal (None, 15, 15, 192)  768         conv_pre_inception_2[0][0]
__________________________________________________________________________________________________
activation_5 (Activation)       (None, 15, 15, 192)  0           bn_pre_inception_2[0][0]
__________________________________________________________________________________________________
conv_pre_inception_3 (Conv2D)   (None, 13, 13, 192)  331968      activation_5[0][0]
__________________________________________________________________________________________________
bn_pre_inception_3 (BatchNormal (None, 13, 13, 192)  768         conv_pre_inception_3[0][0]
__________________________________________________________________________________________________
activation_6 (Activation)       (None, 13, 13, 192)  0           bn_pre_inception_3[0][0]
__________________________________________________________________________________________________
mpool_pre_inception_2 (MaxPooli (None, 12, 12, 192)  0           activation_6[0][0]
__________________________________________________________________________________________________
conv_tower2_1_3a (Conv2D)       (None, 12, 12, 96)   18528       mpool_pre_inception_2[0][0]
__________________________________________________________________________________________________
conv_tower3_1_3a (Conv2D)       (None, 12, 12, 16)   3088        mpool_pre_inception_2[0][0]
__________________________________________________________________________________________________
mpool_tower4_3a (MaxPooling2D)  (None, 12, 12, 192)  0           mpool_pre_inception_2[0][0]
__________________________________________________________________________________________________
conv_tower1_3a (Conv2D)         (None, 12, 12, 64)   12352       mpool_pre_inception_2[0][0]
__________________________________________________________________________________________________
conv_tower2_23a (Conv2D)        (None, 12, 12, 128)  110720      conv_tower2_1_3a[0][0]
__________________________________________________________________________________________________
conv_tower3_2_3a (Conv2D)       (None, 12, 12, 32)   12832       conv_tower3_1_3a[0][0]
__________________________________________________________________________________________________
conv_tower4_3a (Conv2D)         (None, 12, 12, 32)   6176        mpool_tower4_3a[0][0]
__________________________________________________________________________________________________
concatenate_10 (Concatenate)    (None, 12, 12, 256)  0           conv_tower1_3a[0][0]
                                                                 conv_tower2_23a[0][0]
                                                                 conv_tower3_2_3a[0][0]
                                                                 conv_tower4_3a[0][0]
__________________________________________________________________________________________________
conv_tower2_1_3b (Conv2D)       (None, 12, 12, 128)  32896       concatenate_10[0][0]
__________________________________________________________________________________________________
conv_tower3_1_3b (Conv2D)       (None, 12, 12, 32)   8224        concatenate_10[0][0]
__________________________________________________________________________________________________
mpool_tower4_3b (MaxPooling2D)  (None, 12, 12, 256)  0           concatenate_10[0][0]
__________________________________________________________________________________________________
conv_tower1_3b (Conv2D)         (None, 12, 12, 128)  32896       concatenate_10[0][0]
__________________________________________________________________________________________________
conv_tower2_23b (Conv2D)        (None, 12, 12, 192)  221376      conv_tower2_1_3b[0][0]
__________________________________________________________________________________________________
conv_tower3_2_3b (Conv2D)       (None, 12, 12, 96)   76896       conv_tower3_1_3b[0][0]
__________________________________________________________________________________________________
conv_tower4_3b (Conv2D)         (None, 12, 12, 64)   16448       mpool_tower4_3b[0][0]
__________________________________________________________________________________________________
concatenate_11 (Concatenate)    (None, 12, 12, 480)  0           conv_tower1_3b[0][0]
                                                                 conv_tower2_23b[0][0]
                                                                 conv_tower3_2_3b[0][0]
                                                                 conv_tower4_3b[0][0]
__________________________________________________________________________________________________
mpool_inception_3 (MaxPooling2D (None, 11, 11, 480)  0           concatenate_11[0][0]
__________________________________________________________________________________________________
conv_tower2_1_4a (Conv2D)       (None, 11, 11, 96)   46176       mpool_inception_3[0][0]
__________________________________________________________________________________________________
conv_tower3_1_4a (Conv2D)       (None, 11, 11, 16)   7696        mpool_inception_3[0][0]
__________________________________________________________________________________________________
mpool_tower4_4a (MaxPooling2D)  (None, 11, 11, 480)  0           mpool_inception_3[0][0]
```

```
________________________________________________________________________________________________
conv_tower1_4a (Conv2D)         (None, 11, 11, 192)  92352       mpool_inception_3[0][0]
________________________________________________________________________________________________
conv_tower2_24a (Conv2D)        (None, 11, 11, 208)  179920      conv_tower2_1_4a[0][0]
________________________________________________________________________________________________
conv_tower3_2_4a (Conv2D)       (None, 11, 11, 48)   19248       conv_tower3_1_4a[0][0]
________________________________________________________________________________________________
conv_tower4_4a (Conv2D)         (None, 11, 11, 64)   30784       mpool_tower4_4a[0][0]
________________________________________________________________________________________________
concatenate_12 (Concatenate)    (None, 11, 11, 512)  0           conv_tower1_4a[0][0]
                                                                 conv_tower2_24a[0][0]
                                                                 conv_tower3_2_4a[0][0]
                                                                 conv_tower4_4a[0][0]
________________________________________________________________________________________________
conv_tower2_1_4b (Conv2D)       (None, 11, 11, 112)  57456       concatenate_12[0][0]
________________________________________________________________________________________________
conv_tower3_1_4b (Conv2D)       (None, 11, 11, 24)   12312       concatenate_12[0][0]
________________________________________________________________________________________________
mpool_tower4_4b (MaxPooling2D)  (None, 11, 11, 512)  0           concatenate_12[0][0]
________________________________________________________________________________________________
conv_tower1_4b (Conv2D)         (None, 11, 11, 160)  82080       concatenate_12[0][0]
________________________________________________________________________________________________
conv_tower2_24b (Conv2D)        (None, 11, 11, 224)  226016      conv_tower2_1_4b[0][0]
________________________________________________________________________________________________
conv_tower3_2_4b (Conv2D)       (None, 11, 11, 64)   38464       conv_tower3_1_4b[0][0]
________________________________________________________________________________________________
conv_tower4_4b (Conv2D)         (None, 11, 11, 64)   32832       mpool_tower4_4b[0][0]
________________________________________________________________________________________________
concatenate_13 (Concatenate)    (None, 11, 11, 512)  0           conv_tower1_4b[0][0]
                                                                 conv_tower2_24b[0][0]
                                                                 conv_tower3_2_4b[0][0]
                                                                 conv_tower4_4b[0][0]
________________________________________________________________________________________________
conv_tower2_1_4c (Conv2D)       (None, 11, 11, 128)  65664       concatenate_13[0][0]
________________________________________________________________________________________________
conv_tower3_1_4c (Conv2D)       (None, 11, 11, 24)   12312       concatenate_13[0][0]
________________________________________________________________________________________________
mpool_tower4_4c (MaxPooling2D)  (None, 11, 11, 512)  0           concatenate_13[0][0]
________________________________________________________________________________________________
conv_tower1_4c (Conv2D)         (None, 11, 11, 128)  65664       concatenate_13[0][0]
________________________________________________________________________________________________
conv_tower2_24c (Conv2D)        (None, 11, 11, 256)  295168      conv_tower2_1_4c[0][0]
________________________________________________________________________________________________
conv_tower3_2_4c (Conv2D)       (None, 11, 11, 24)   14424       conv_tower3_1_4c[0][0]
________________________________________________________________________________________________
conv_tower4_4c (Conv2D)         (None, 11, 11, 64)   32832       mpool_tower4_4c[0][0]
________________________________________________________________________________________________
concatenate_14 (Concatenate)    (None, 11, 11, 472)  0           conv_tower1_4c[0][0]
                                                                 conv_tower2_24c[0][0]
                                                                 conv_tower3_2_4c[0][0]
                                                                 conv_tower4_4c[0][0]
________________________________________________________________________________________________
conv_tower2_1_4d (Conv2D)       (None, 11, 11, 144)  68112       concatenate_14[0][0]
________________________________________________________________________________________________
conv_tower3_1_4d (Conv2D)       (None, 11, 11, 32)   15136       concatenate_14[0][0]
________________________________________________________________________________________________
mpool_tower4_4d (MaxPooling2D)  (None, 11, 11, 472)  0           concatenate_14[0][0]
________________________________________________________________________________________________
conv_tower1_4d (Conv2D)         (None, 11, 11, 112)  52976       concatenate_14[0][0]
________________________________________________________________________________________________
conv_tower2_24d (Conv2D)        (None, 11, 11, 288)  373536      conv_tower2_1_4d[0][0]
________________________________________________________________________________________________
conv_tower3_2_4d (Conv2D)       (None, 11, 11, 64)   51264       conv_tower3_1_4d[0][0]
________________________________________________________________________________________________
conv_tower4_4d (Conv2D)         (None, 11, 11, 64)   30272       mpool_tower4_4d[0][0]
________________________________________________________________________________________________
concatenate_15 (Concatenate)    (None, 11, 11, 528)  0           conv_tower1_4d[0][0]
                                                                 conv_tower2_24d[0][0]
                                                                 conv_tower3_2_4d[0][0]
                                                                 conv_tower4_4d[0][0]
__________________________________________________________________________________________________
conv_tower2_1_4e (Conv2D)       (None, 11, 11, 120)  63480       concatenate_15[0][0]
__________________________________________________________________________________________________
conv_tower3_1_4e (Conv2D)       (None, 11, 11, 32)   16928       concatenate_15[0][0]
__________________________________________________________________________________________________
mpool_tower4_4e (MaxPooling2D)  (None, 11, 11, 528)  0           concatenate_15[0][0]
__________________________________________________________________________________________________
conv_tower1_4e (Conv2D)         (None, 11, 11, 256)  135424      concatenate_15[0][0]
__________________________________________________________________________________________________
conv_tower2_24e (Conv2D)        (None, 11, 11, 320)  345920      conv_tower2_1_4e[0][0]
__________________________________________________________________________________________________
conv_tower3_2_4e (Conv2D)       (None, 11, 11, 128)  102528      conv_tower3_1_4e[0][0]
__________________________________________________________________________________________________
conv_tower4_4e (Conv2D)         (None, 11, 11, 128)  67712       mpool_tower4_4e[0][0]
__________________________________________________________________________________________________
concatenate_16 (Concatenate)    (None, 11, 11, 832)  0           conv_tower1_4e[0][0]
                                                                 conv_tower2_24e[0][0]
                                                                 conv_tower3_2_4e[0][0]
                                                                 conv_tower4_4e[0][0]
__________________________________________________________________________________________________
mpool_inception_4 (MaxPooling2D (None, 10, 10, 832)  0           concatenate_16[0][0]
__________________________________________________________________________________________________
conv_tower2_1_5a (Conv2D)       (None, 10, 10, 160)  133280      mpool_inception_4[0][0]
__________________________________________________________________________________________________
conv_tower3_1_5a (Conv2D)       (None, 10, 10, 32)   26656       mpool_inception_4[0][0]
__________________________________________________________________________________________________
mpool_tower4_5a (MaxPooling2D)  (None, 10, 10, 832)  0           mpool_inception_4[0][0]
__________________________________________________________________________________________________
conv_tower1_5a (Conv2D)         (None, 10, 10, 256)  213248      mpool_inception_4[0][0]
__________________________________________________________________________________________________
conv_tower2_25a (Conv2D)        (None, 10, 10, 320)  461120      conv_tower2_1_5a[0][0]
__________________________________________________________________________________________________
conv_tower3_2_5a (Conv2D)       (None, 10, 10, 128)  102528      conv_tower3_1_5a[0][0]
__________________________________________________________________________________________________
conv_tower4_5a (Conv2D)         (None, 10, 10, 128)  106624      mpool_tower4_5a[0][0]
__________________________________________________________________________________________________
concatenate_17 (Concatenate)    (None, 10, 10, 832)  0           conv_tower1_5a[0][0]
                                                                 conv_tower2_25a[0][0]
                                                                 conv_tower3_2_5a[0][0]
                                                                 conv_tower4_5a[0][0]
__________________________________________________________________________________________________
conv_tower2_1_5b (Conv2D)       (None, 10, 10, 192)  159936      concatenate_17[0][0]
__________________________________________________________________________________________________
conv_tower3_1_5b (Conv2D)       (None, 10, 10, 48)   39984       concatenate_17[0][0]
__________________________________________________________________________________________________
mpool_tower4_5b (MaxPooling2D)  (None, 10, 10, 832)  0           concatenate_17[0][0]
__________________________________________________________________________________________________
average_pooling2d_3 (AveragePoo (None, 3, 3, 512)    0           concatenate_12[0][0]
__________________________________________________________________________________________________
average_pooling2d_4 (AveragePoo (None, 3, 3, 528)    0           concatenate_15[0][0]
__________________________________________________________________________________________________
conv_tower1_5b (Conv2D)         (None, 10, 10, 384)  319872      concatenate_17[0][0]
__________________________________________________________________________________________________
conv_tower2_25b (Conv2D)        (None, 10, 10, 384)  663936      conv_tower2_1_5b[0][0]
__________________________________________________________________________________________________
conv_tower3_2_5b (Conv2D)       (None, 10, 10, 128)  153728      conv_tower3_1_5b[0][0]
__________________________________________________________________________________________________
conv_tower4_5b (Conv2D)         (None, 10, 10, 128)  106624      mpool_tower4_5b[0][0]
__________________________________________________________________________________________________
conv2d_3 (Conv2D)               (None, 3, 3, 128)    65664       average_pooling2d_3[0][0]
__________________________________________________________________________________________________
conv2d_4 (Conv2D)               (None, 3, 3, 128)    67712       average_pooling2d_4[0][0]
__________________________________________________________________________________________________
concatenate_18 (Concatenate)    (None, 10, 10, 1024) 0           conv_tower1_5b[0][0]
                                                                 conv_tower2_25b[0][0]
                                                                 conv_tower3_2_5b[0][0]
                                                                 conv_tower4_5b[0][0]
__________________________________________________________________________________________________
```

```
flatten_4 (Flatten)             (None, 1152)         0           conv2d_3[0][0]
__________________________________________________________________________________________________
flatten_5 (Flatten)             (None, 1152)         0           conv2d_4[0][0]
__________________________________________________________________________________________________
apool_inception_5 (AveragePooli (None, 4, 4, 1024)   0           concatenate_18[0][0]
__________________________________________________________________________________________________
dense_4 (Dense)                 (None, 256)          295168      flatten_4[0][0]
__________________________________________________________________________________________________
dense_5 (Dense)                 (None, 256)          295168      flatten_5[0][0]
__________________________________________________________________________________________________
dropout_6 (Dropout)             (None, 4, 4, 1024)   0           apool_inception_5[0][0]
__________________________________________________________________________________________________
dropout_4 (Dropout)             (None, 256)          0           dense_4[0][0]
__________________________________________________________________________________________________
dropout_5 (Dropout)             (None, 256)          0           dense_5[0][0]
__________________________________________________________________________________________________
flatten_6 (Flatten)             (None, 16384)        0           dropout_6[0][0]
__________________________________________________________________________________________________
aux1 (Dense)                    (None, 100)          25700       dropout_4[0][0]
__________________________________________________________________________________________________
aux2 (Dense)                    (None, 100)          25700       dropout_5[0][0]
__________________________________________________________________________________________________
dense_6 (Dense)                 (None, 1000)         16385000    flatten_6[0][0]
__________________________________________________________________________________________________
lambda_3 (Lambda)               (None, 100)          0           aux1[0][0]
__________________________________________________________________________________________________
lambda_4 (Lambda)               (None, 100)          0           aux2[0][0]
__________________________________________________________________________________________________
main (Dense)                    (None, 100)          100100      dense_6[0][0]
__________________________________________________________________________________________________
add_2 (Add)                     (None, 100)          0           lambda_3[0][0]
                                                                 lambda_4[0][0]
                                                                 main[0][0]
==================================================================================================
Total params: 23,324,884
Trainable params: 23,324,052
Non-trainable params: 832
```

Output:

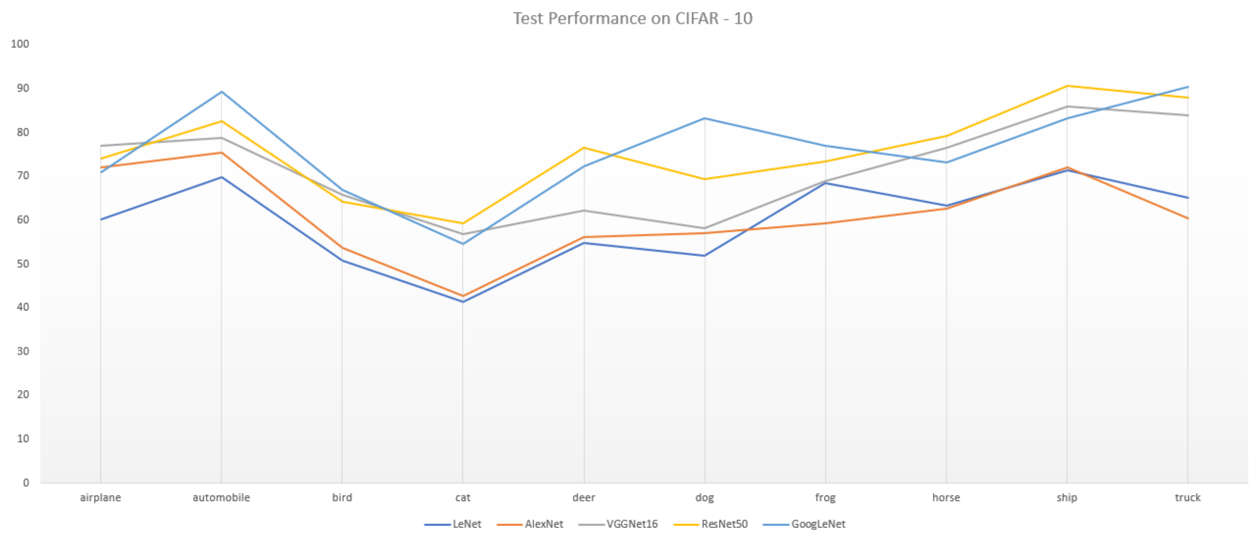


```
Epoch 42/50
50000/50000 [==============================] - 83s 2ms/step - loss: 0.5337 - accuracy: 0.8869 - val_loss: 2.6058 - val_accuracy: 0.4633
Epoch 43/50
50000/50000 [==============================] - 83s 2ms/step - loss: 0.5211 - accuracy: 0.8902 - val_loss: 2.5818 - val_accuracy: 0.4826
Epoch 44/50
50000/50000 [==============================] - 83s 2ms/step - loss: 0.5097 - accuracy: 0.8948 - val_loss: 2.5864 - val_accuracy: 0.4718
Epoch 45/50
50000/50000 [==============================] - 83s 2ms/step - loss: 0.5032 - accuracy: 0.8949 - val_loss: 2.6908 - val_accuracy: 0.4750
Epoch 46/50
50000/50000 [==============================] - 83s 2ms/step - loss: 0.5060 - accuracy: 0.8945 - val_loss: 2.6691 - val_accuracy: 0.4663
Epoch 47/50
50000/50000 [==============================] - 83s 2ms/step - loss: 0.4981 - accuracy: 0.8954 - val_loss: 2.6933 - val_accuracy: 0.4556
Epoch 48/50
50000/50000 [==============================] - 83s 2ms/step - loss: 0.4883 - accuracy: 0.8989 - val_loss: 2.6563 - val_accuracy: 0.4743
Epoch 49/50
50000/50000 [==============================] - 83s 2ms/step - loss: 0.4771 - accuracy: 0.9040 - val_loss: 2.6264 - val_accuracy: 0.4842- ETA: 2s - los
Epoch 50/50
50000/50000 [==============================] - 83s 2ms/step - loss: 0.4737 - accuracy: 0.9032 - val_loss: 2.7607 - val_accuracy: 0.4802
[8]: <keras.callbacks.callbacks.History at 0x7fa650c4f240>

[9]: scores = c100_googlenet.evaluate(c100_x_test, c100_y_test)
     print('Test loss:', scores[0])
     print('Test accuracy:', scores[1])

10000/10000 [==============================] - 7s 661us/step
Test loss: 2.7607055408477783
Test accuracy: 0.48019999265670776
```

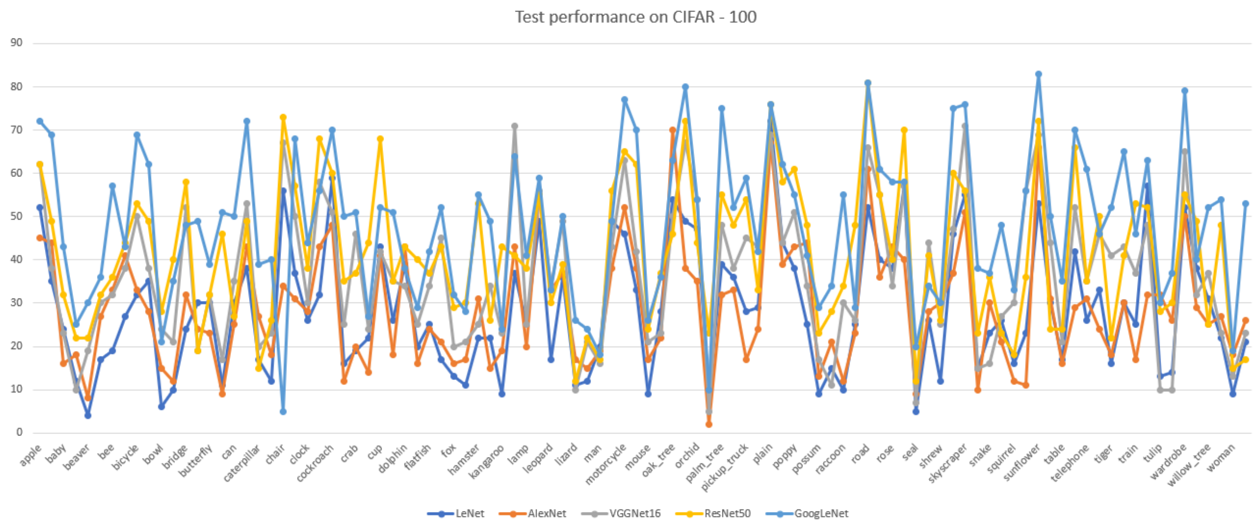
5. Conclusion

5.1 Results:

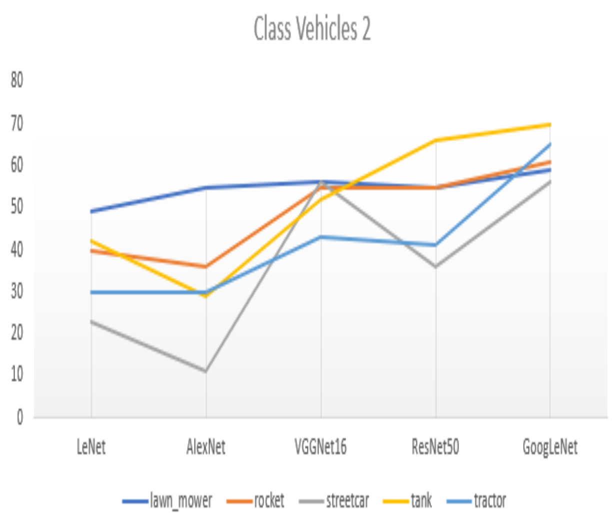
Graph for comparative accuracy of CNN’s on Ciphar10 dataset

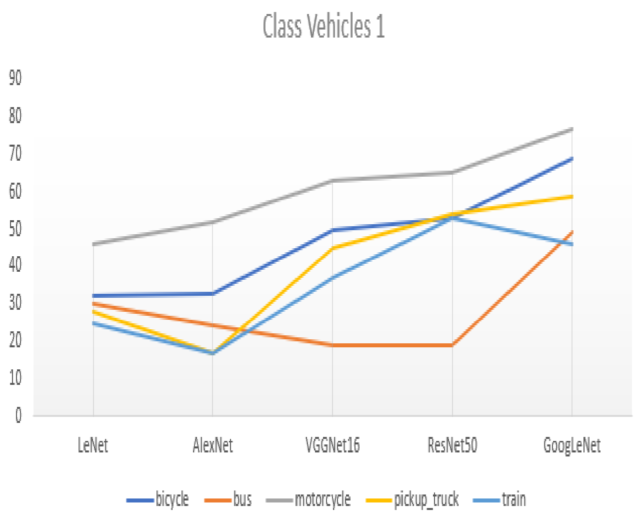


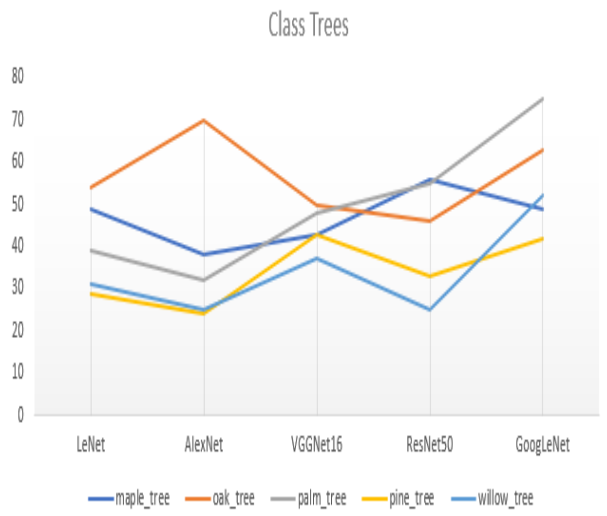
Graph for comparative accuracy of CNN’s on Ciphar100 dataset

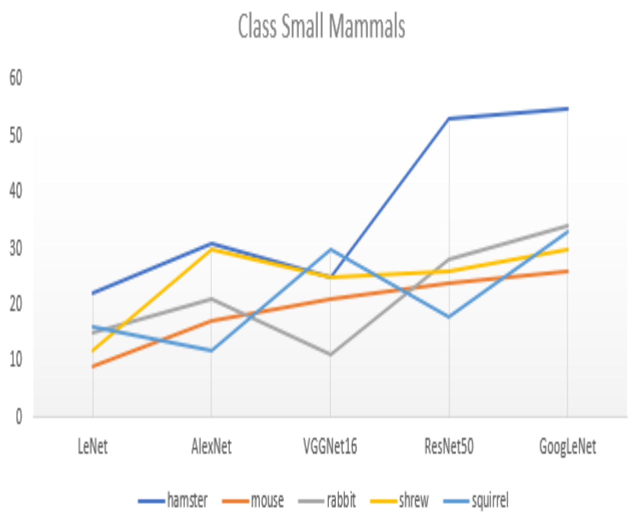


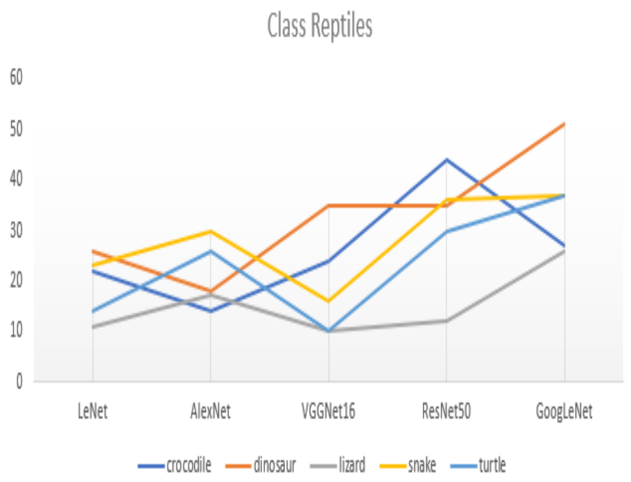
Following are the class wise accuracy of different neural networks

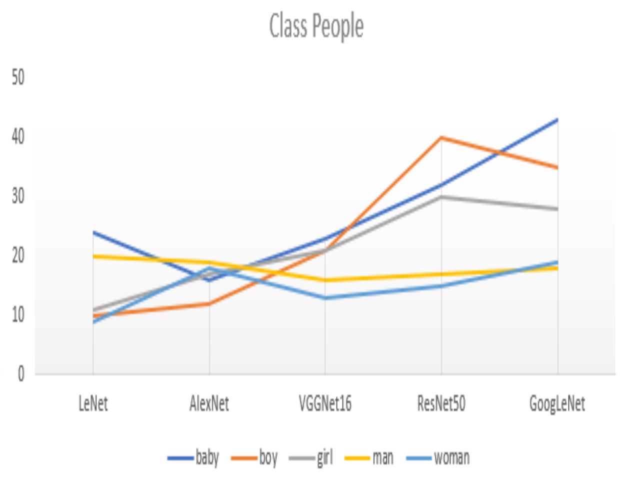


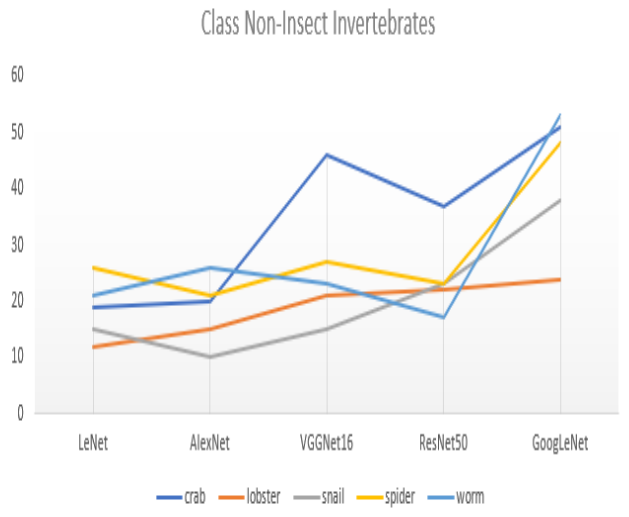


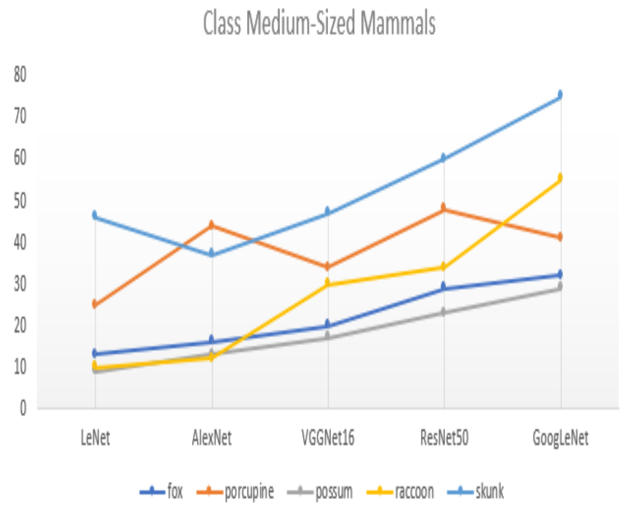


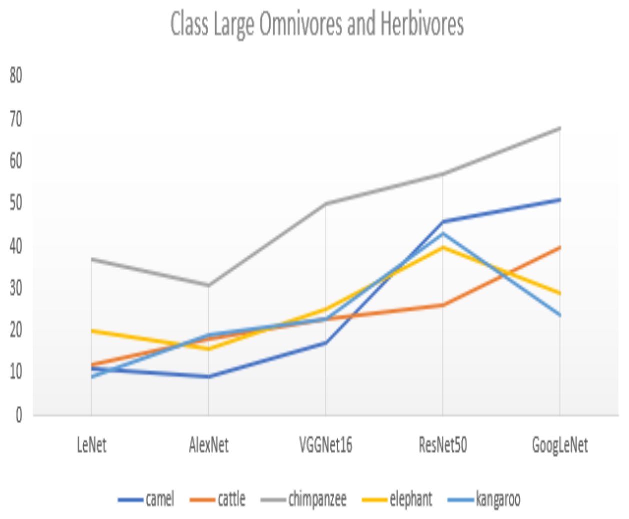


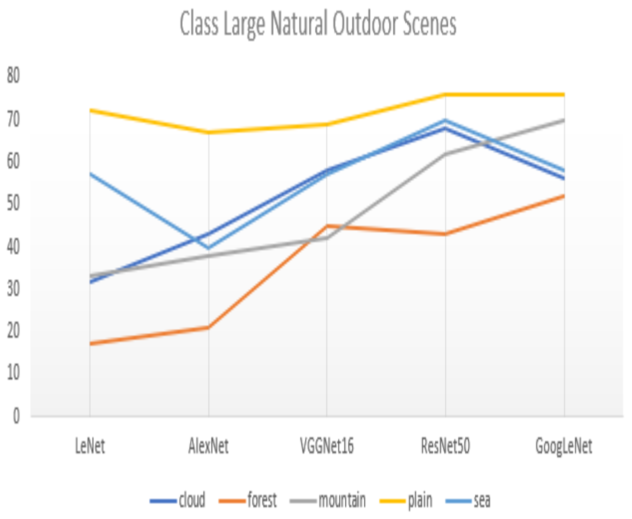


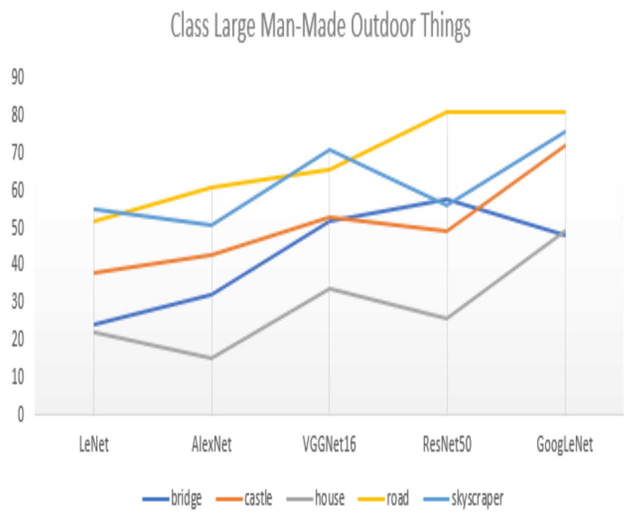


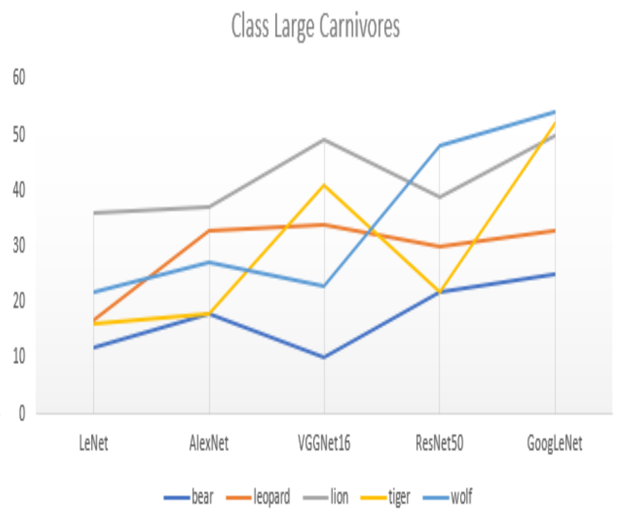

Class Non-Insect Invertebrates
0
10
20
30
40
50
60
LeNet
AlexNet
VGGNet16
ResNet50
GoogLeNet
crab
lobster
snail
spider
worm

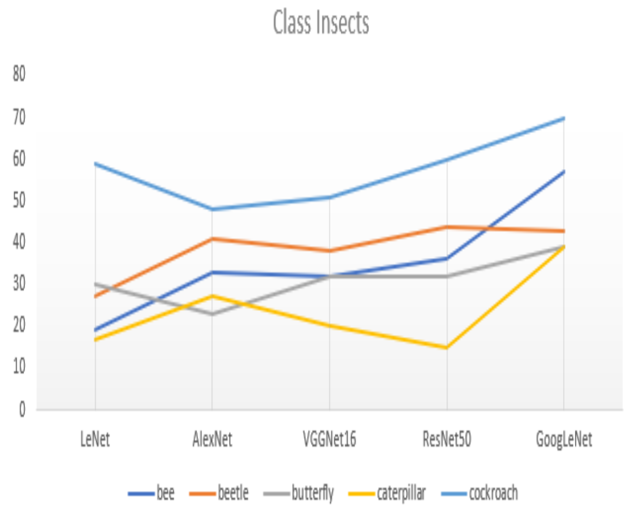

Class Medium-Sized Mammals
0
10
20
30
40
50
60
70
80
LeNet
AlexNet
VGGNet16
ResNet50
GoogLeNet
fox
porcupine
possum
raccoon
skunk

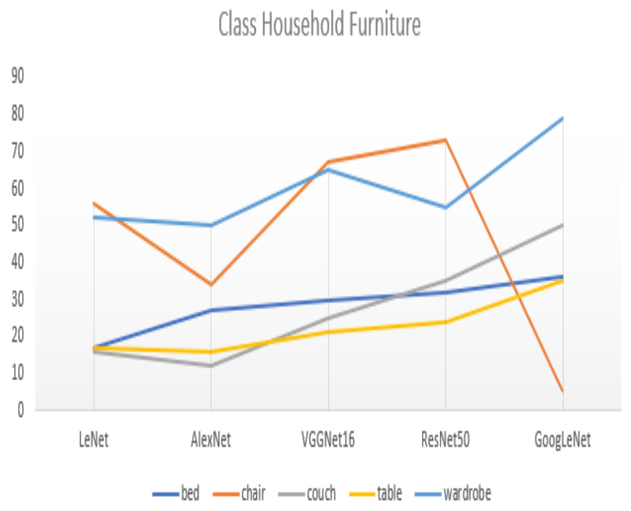

Class Large Omnivores and Herbivores
0
10
20
30
40
50
60
70
80
LeNet
AlexNet
VGGNet16
ResNet50
GoogLeNet
camel
cattle
chimpanzee
elephant
kangaroo

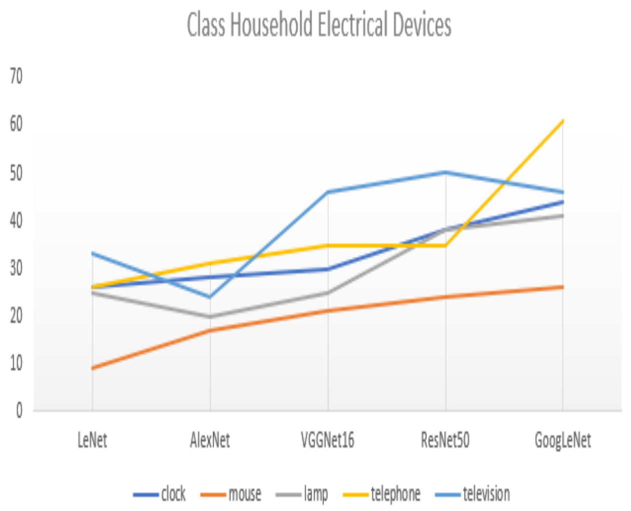

Class Large Natural Outdoor Scenes
0
10
20
30
40
50
60
70
80
LeNet
AlexNet
VGGNet16
ResNet50
GoogLeNet
cloud
forest
mountain
plain
sea

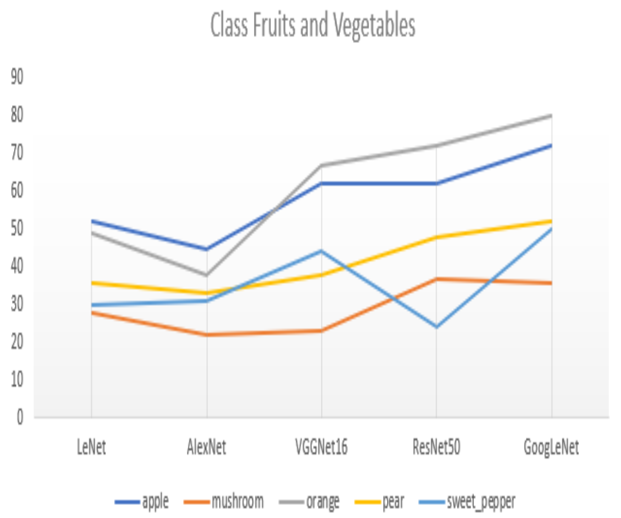

Class Large Man-Made Outdoor Things
0
10
20
30
40
50
60
70
80
90
LeNet
AlexNet
VGGNet16
ResNet50
GoogLeNet
bridge
castle
house
road
skyscraper

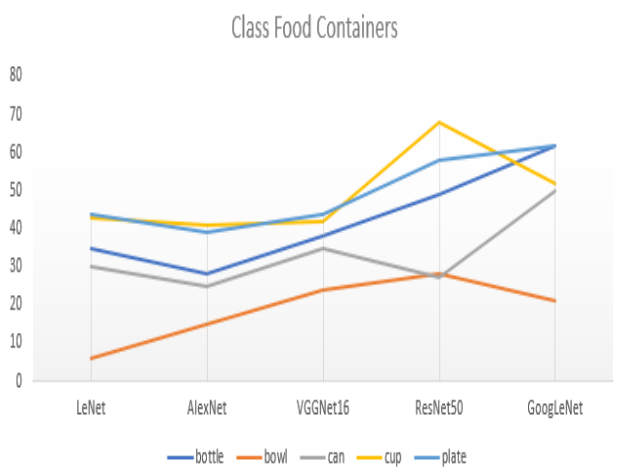

Class Large Carnivores
0
10
20
30
40
50
60
LeNet
AlexNet
VGGNet16
ResNet50
GoogLeNet
bear
leopard
lion
tiger
wolf

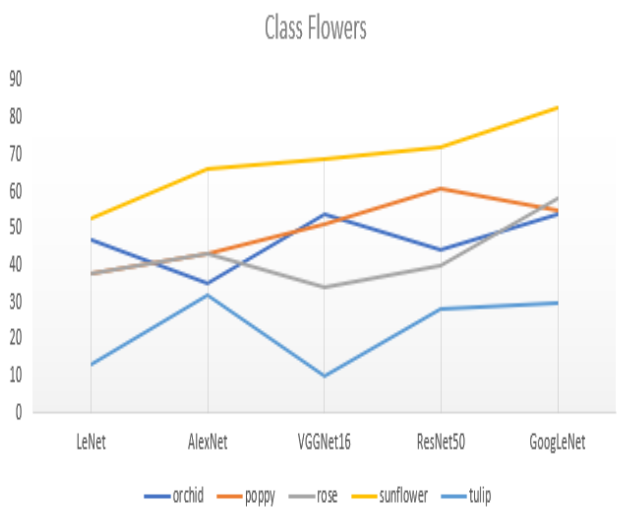

Class Insects
80
70
60
50
40
30
20
10
0
LeNet
AlexNet
VGGNet16
ResNet50
GoogLeNet
bee
beetle
butterfly
caterpillar
cockroach

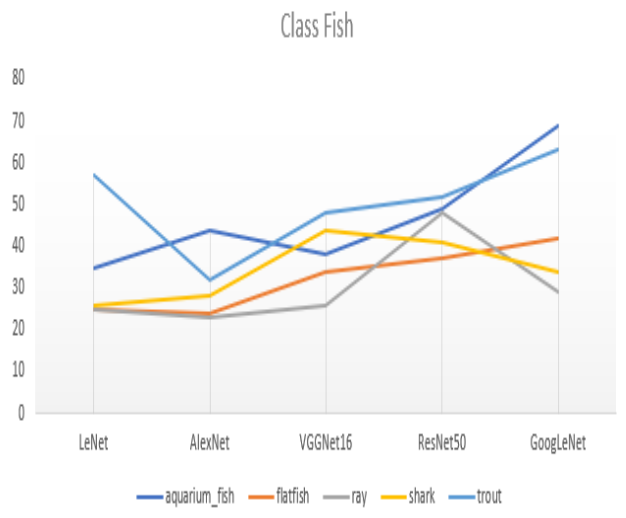

Class Household Furniture
90
80
70
60
50
40
30
20
10
0
LeNet
AlexNet
VGGNet16
ResNet50
GoogLeNet
bed
chair
couch
table
wardrobe

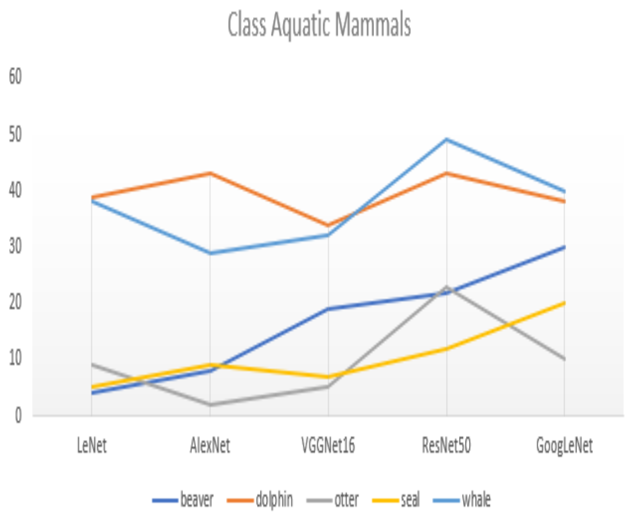

Class Household Electrical Devices
70
60
50
40
30
20
10
0
LeNet
AlexNet
VGGNet16
ResNet50
GoogLeNet
clock
mouse
lamp
telephone
television


Class Fruits and Vegetables
90
80
70
60
50
40
30
20
10
0
LeNet
AlexNet
VGGNet16
ResNet50
GoogLeNet
apple
mushroom
orange
pear
sweet_pepper

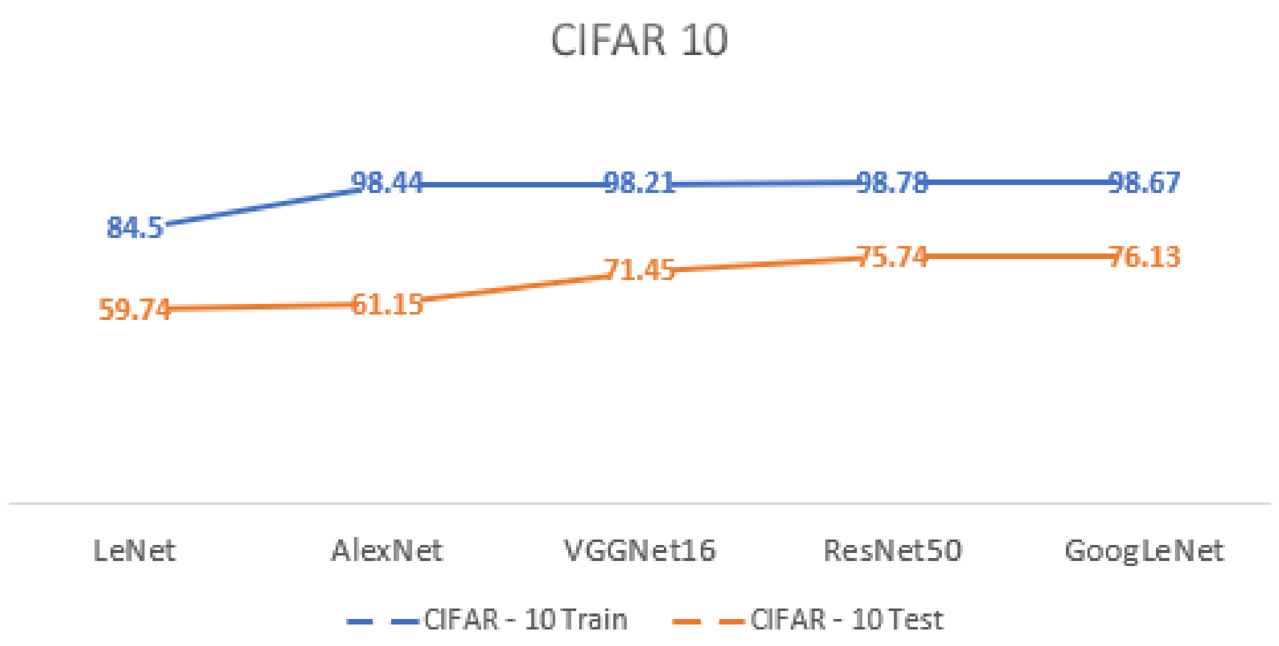

Class Food Containers
80
70
60
50
40
30
20
10
0
LeNet
AlexNet
VGGNet16
ResNet50
GoogLeNet
bottle
bowl
can
cup
plate


Class Flowers
90
80
70
60
50
40
30
20
10
0
LeNet
AlexNet
VGGNet16
ResNet50
GoogLeNet
orchid
poppy
rose
sunflower
tulip

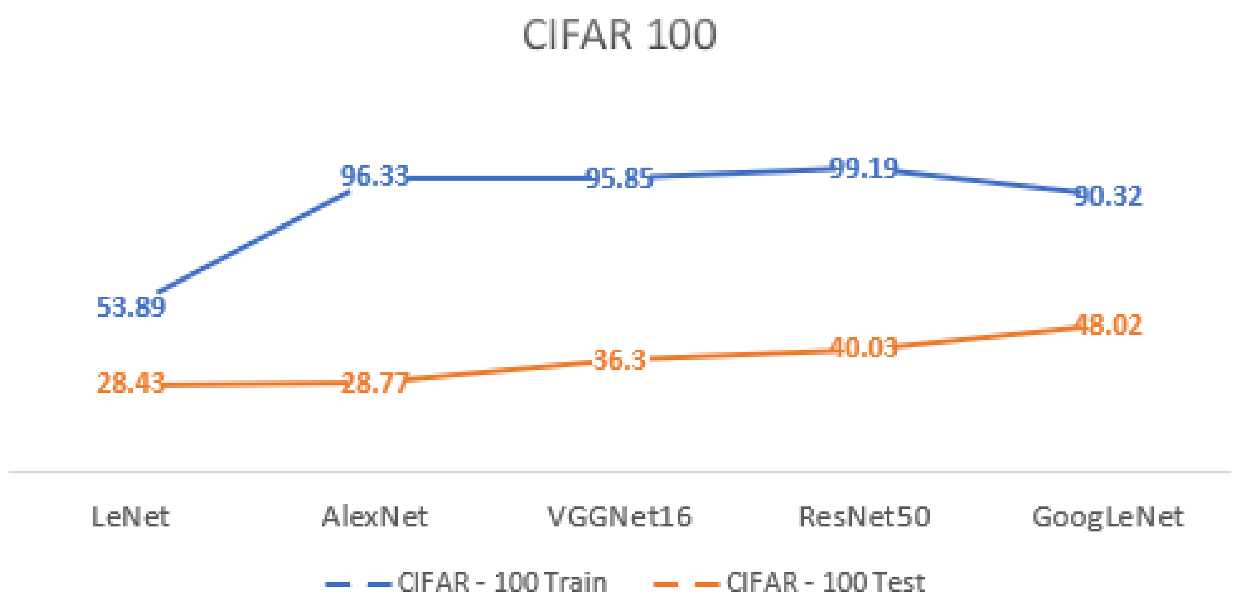

Class Fish
80
70
60
50
40
30
20
10
0
LeNet
AlexNet
VGGNet16
ResNet50
GoogLeNet
aquarium_fish
flatfish
ray
shark
trout


Class Aquatic Mammals
60
50
40
30
20
10
0
LeNet
AlexNet
VGGNet16
ResNet50
GoogLeNet
beaver
dolphin
otter
seal
whale

Overall Accuracy on Ciphar10:



Overall Accuracy on Ciphar100:



Learning Outcome :

The work analyzed the prediction accuracy of Five different convolutional neural networks (CNN) on most popular training and test datasets namely CIFAR10 and CIFAR100. Our main purpose was to find out the accuracy of the different networks on same datasets and evaluating the consistency of prediction by each of this CNN. We have presented a thorough prediction analysis for comparing the networks' performance for different classes of objects. It is important to note that complex frames often create confusion for the network to detect and recognize the scene. Hence, more the number of layers, more will be the training and therefore, higher the rate of accuracy in prediction will be achieved.

GoogLeNET accuracy is the highest. Although the network is of 22 layers. But as different convolutions are used and auxiliary classifiers are implemented the problem for gradient vanishing is minimized hence the increase in accuracy.

ResNET's accuracy is tantamount to GoogLeNET , so it concludes that the more the number of layer the more is accuracy.

VGGNET accuracy is moderate in comparison to others. It took lot of time in training because of lots of parameters making it computationally expensive and time consuming.

AlexNET and LENET are rudimentary with few layers and less parameters so it causes underfitting. Therefore, the accuracy is on the lower side.

Best Performance was of GoogLeNET and ResNet50 across different classes

Following are the classes with maximum accuracies

Classes	GoogLeNT	ResNET50
Sunflower	83	72
Road	81	81
Orange	80	72
Plain	76	76

It can further be summed up that neural networks are new and best emerging techniques for making a machine intelligent for solving many real-life object categorization problems. Many types of research and works are being done on it. It has wide applications and it is easy and flexible to integrate into various platforms. The hardware requirements may not allow the network to be trained on normal desktop work but just with nominal requirements one can train the network and generate the desired model.

Bibliography:

https://en.wikipedia.org/wiki/Convolutional_neural_network

https://cs231n.github.io/convolutional-networks/

https://arxiv.org/abs/1409.4842

http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf

https://arxiv.org/abs/1602.07261

http://www.robots.ox.ac.uk/~vgg/research/very_deep/

https://arxiv.org/abs/1512.03385

https://medium.com/analytics-vidhya/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

https://www.learnopencv.com/understanding-alexnet/

https://www.pyimagesearch.com/2016/08/01/lenet-convolutional-neural-network-in-python/

https://engmrk.com/lenet-5-a-classic-cnn-architecture/

https://neurohive.io/en/popular-networks/vgg16/

https://www.kaggle.com/keras/resnet50

https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43022.pdf

https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/googlenet.html

Internship Documents

STUDENT'S WEEKLY DIARY/ LOG

WEEK-I	DATE from _____01-01-20___ to ____07-01-20__________		
Department/Division	CS-IT	Name of finished Product	
Name of HOD/ Supervisor With e-mail id	Dr. Ashish Bansal		
	ashish.bansal@suas.ac.in		
Main points of the week	Problem statement finalization and research		

WEEK-2	DATE from _____08-01-20__ to ____14-01-20_____		
Department/Division	CS-IT	Name of finished Product	
Name of HOD/ Supervisor With e-mail id	Dr. Ashish Bansal		
	ashish.bansal@suas.ac.in		
Main points of the week	Writing Abstract and gathering resources		

WEEK-3	DATE from ______15-01-20___ to _______20-01-20_____		
Department/Division	CS-IT	Name of finished Product	
Name of HOD/ Supervisor With e-mail id	Dr. Ashish Bansal		
	ashish.bansal@suas.ac.in		
Main points of the week	Theory analysis of artificial intelligence		

WEEK-4	DATE from ____21-01-20____ to _____28-01-20_______		
Department/Division	CS-IT	Name of finished Product	
Name of HOD/ Supervisor With e-mail id	Dr. Ashish Bansal		
	ashish.bansal@suas.ac.in		
Main points of the week	Gathering various research paper and compiling resources		

WEEK-5	DATE from ____29-01-20___ to ______04-02-20_____		
Department/Division	CS-IT	Name of finished Product	
Name of HOD/ Supervisor With e-mail id	Dr. Ashish Bansal		
	ashish.bansal@suas.ac.in		
Main points of the week	Analyzing Machine learning algorithms and getting cetrifications		

WEEK-6	DATE from ___05-02-20_____ to ______11-02-20____		
Department/Division	CS-IT	Name of finished Product	
Name of HOD/ Supervisor With e-mail id	Dr. Ashish Bansal		
	ashish.bansal@suas.ac.in		
Main points of the week	Theory analysis of deep learning algorithms		

WEEK-7	DATE from _____12-02-20___ to ____19-02-20__________		
Department/Division	CS-IT	Name of finished Product	
Name of HOD/ Supervisor With e-mail id	Dr. Ashish Bansal		
	ashish.bansal@suas.ac.in		
Main points of the week	Researching about Neural Networks and their different types		

WEEK-8	DATE from _____21-02-20__ to ____28-02-20_____		
Department/Division	CS-IT	Name of finished Product	
Name of HOD/ Supervisor With e-mail id	Dr. Ashish Bansal		
	ashish.bansal@suas.ac.in		
Main points of the week	Creating first Neural Network and understanding its working		

WEEK-9	DATE from ______01-03-20___ to _______8-03-20_____		
Department/Division	CS-IT	Name of finished Product	
Name of HOD/ Supervisor With e-mail id	Dr. Ashish Bansal		
	ashish.bansal@suas.ac.in		
Main points of the week	AlexNet implementation on Ciphar10		

WEEK-10	DATE from ____09-03-20____ to _____15-03-20_______		
Department/Division	CS-IT	Name of finished Product	
Name of HOD/ Supervisor With e-mail id	Dr. Ashish Bansal		
	ashish.bansal@suas.ac.in		
Main points of the week	LENET implementation on Ciphar100 and ciphar10		

WEEK-11	DATE from ____16-03-20___ to ______23-03-20_____		
Department/Division	CS-IT	Name of finished Product	
Name of HOD/ Supervisor With e-mail id	Dr. Ashish Bansal		
	ashish.bansal@suas.ac.in		
Main points of the week	Alexnet Implementation on Ciphar100		

WEEK-12	DATE from ___24-03-20_____ to ______1-04-20____		
Department/Division	CS-IT	Name of finished Product	
Name of HOD/ Supervisor With e-mail id	Dr. Ashish Bansal		
	ashish.bansal@suas.ac.in		
Main points of the week	Implementing VGGNET on ciphar10		

WEEK-13	DATE from ___2-04-20_____ to ______9-04-20____		
Department/Division	CS-IT	Name of finished Product	
Name of HOD/ Supervisor With e-mail id	Dr. Ashish Bansal		
	ashish.bansal@suas.ac.in		
Main points of the week	Implementing VGGNET on ciphar100 and compiling both		

WEEK-14	DATE from ___10-04-20_____ to ______18-04-20____		
Department/Division	CS-IT	Name of finished Product	
Name of HOD/ Supervisor With e-mail id	Dr. Ashish Bansal		
	ashish.bansal@suas.ac.in		
Main points of the week	Creating and implementing ResNET on ciphar 10 and ciphar 100		

WEEK-15	DATE from ___19-04-20_____ to ______25-04-20____		
Department/Division	CS-IT	Name of finished Product	
Name of HOD/ Supervisor With e-mail id	Dr. Ashish Bansal		
	ashish.bansal@suas.ac.in		
Main points of the week	Training of GoogLENET inception model on ciphar 10 and 100		

WEEK-16	DATE from ___26-04-20_____ to ______2-05-20____		
Department/Division	CS-IT	Name of finished Product	
Name of HOD/ Supervisor With e-mail id	Dr. Ashish Bansal		
	ashish.bansal@suas.ac.in		
Main points of the week	Comparing and contrasting various networks		

WEEK-17	DATE from ___03-05-20_____ to ______09-05-20____		
Department/Division	CS-IT	Name of finished Product	
Name of HOD/ Supervisor With e-mail id	Dr. Ashish Bansal		
	ashish.bansal@suas.ac.in		
Main points of the week	Compiling output and ranking the networks according to result		

WEEK-18	DATE from ___10-05-20_____ to ______15-05-20____		
Department/Division	CS-IT	Name of finished Product	
Name of HOD/ Supervisor With e-mail id	Dr. Ashish Bansal		
	ashish.bansal@suas.ac.in		
Main points of the week	Report formation and final result calculations		

Signature of Industry Supervisor

STUDENT FEEDBACK OF INTERNSHIP (TO BE FILLED BY STUDENTS AFTER INTERNSHIP COMPLETION)

Student Name: ________HARSHIT AGRAWAL______Date: ________15-May-2020___________

Industrial Supervisor: _____Dr. Ashish Bansal_________ Title: ___Research Project_____________

Supervisor Email: __ashish.bansal@suas.ac.in______________Internship is: ________ Unpaid_______

Company/Organization: ___________Symbiosis University of Applied Sciences___________

Internship Address: __Bada Bangadda Super Corridor, Near Airport, Indore, Madhya Pradesh 453112______

Faculty Coordinator: __Dr. Ashish Bansal_________Department: Computer Science and Information technology

Dates of Internship: From______01-Jan-2020______To__________15-May-2020______________

Please fill out the above in full detail

Give a brief description of your internship work (title and tasks for which you were responsible): Was your internship experience related to your major area of study?

________✓______Yes, to a large degree_______________________Yes, to Slight degree _____________ No, not related at all Indicate the degree to which you agree or disagree with the following statements.

This experience has:	Strongly Agree	Agree	No Opinion	Disagree	Strongly Agree
Given me the opportunity to explore a career field	✓				
Allowed me to apply classroom theory to practice	✓				
Helped me develop my decision-making and problem-solving Skills	✓				
Expanded my knowledge about the work world prior to permanent employment	✓				
Helped me develop my written and oral communication skills	✓				
Provided a chance to use leadership skills (influence others, develop ideas with others. stimulate decision-making and action)	✓				

This experience has:	Strongly Agree	Agree	No Opinion	Disagree	Strongly Agree
Expanded my sensitivity to the ethical implications of the work involved	✓				
Made it possible for me to be more confident in new situations	✓				
Give me a chance to improve my interpersonal skills	✓				
Helped me learn to handle responsibility and use my time wisely	✓				
Helped me discover new aspects of myself that I didn't know existed before	✓				
Helped me develop new interests and abilities	✓				
Helped me clarify my career goals	✓				
Provided me with contacts which may lead to future employment	✓				
Allowed me to acquire information and/ or use equipment not available at my Institute	✓				

In the Institute internship program, faculty members are expected to be mentors for students. Do you feel that your faculty coordinator served such a function? Why or why not?

A:- Yes, throughout the internship he helped me a lot and provided guidance whenever required.

How well were you able to accomplish the initial goals, tasks and new skills that were set down in your learning contract? In what ways were you able to take a new direction or expand beyond your contract? Why were some goals not accomplished adequately?

A: I achieved all the goals which were set initially. During this research work I've developed many different skills which were not known by me previously. Due to lack of resources due to the pandemic some of my research work is not completed in the way I wanted

In what areas did you most develop and improve?

A: In the programming and the research analysis area of the internship

What has been the most significant accomplishment or satisfying moment of your internship? What did you dislike about the internship?

A: Reaching the final result was the most significant accomplishment. No particular dislikes.

Considering your overall experience, how would you rate this internship? (circle one).
(Satisfactory/Good/[✓ Excellent])

Give suggestions as to how your internship experience could have been improved. (Could you have handled added responsibility? Would you have liked more discussions with your professor concerning your internship? Was closer supervision needed? Was more of an orientation required?)

A: I am fully satisfied with the internship experience.

ATTENDANCE SHEET

Name and Address of Organization

___Symbiosis University of Applied Sciences _

________near Bada Bangadda_____________

______ Indore, Madhya Pradesh_453112_____

Name of Student	Harshit Agrawal
Enrollment No.	2016AB001031
Name of Course	Btech in CS-IT
Date of Commencement of Training	01-Jan-2020
Date of Completion of Training	15-May-2020

Signature of the student: Harshit

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Jan 2020	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Note:

1. Attendance Sheet should remain affixed in Daily Training Diary. **Do not remove or tear it off.**
2. Student should sign/initial in the attendance column. Do not mark 'P'.
3. Holidays should be marked in **Red Ink** in attendance column. Absent should be marked as **'A'in Red Ink**.

Signature of Company Internship supervisor with (company stamp/ seal)

(Name____) Contact No.

ATTENDANCE SHEET

Name and Address of Organization

___Symbiosis University of Applied Sciences _

_______ near Bada Bangadda______________

______ Indore, Madhya Pradesh 453112_____

Name of Student	Harshit Agrawal
Enrollment No.	2016AB001031
Name of Course	Btech in CS-IT
Date of Commencement of Training	01-Jan-2020
Date of Completion of Training	15-May-2020

Signature of the student: Harshit

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Feb 2020	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]		

Note:

1. Attendance Sheet should remain affixed in Daily Training Diary. **Do not remove or tear it off.**
2. Student should sign/initial in the attendance column. Do not mark 'P'.
3. Holidays should be marked in **Red Ink** in attendance column. Absent should be marked as **'A'in Red Ink**.

Signature of Company Internship supervisor with (company stamp/ seal)

(Name____) Contact No.

ATTENDANCE SHEET

Name and Address of Organization

___Symbiosis University of Applied Sciences _

_______ near Bada Bangadda_____________

_____ Indore, Madhya Pradesh 453112____

Name of Student	Harshit Agrawal
Enrollment No.	2016AB001031
Name of Course	Btech in CS-IT
Date of Commencement of Training	01-Jan-2020
Date of Completion of Training	15-May-2020

Signature of the student: Harshit

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Mar 2020	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Note:

1. Attendance Sheet should remain affixed in Daily Training Diary. **Do not remove or tear it off.**
2. Student should sign/initial in the attendance column. Do not mark 'P'.
3. Holidays should be marked in **Red Ink** in attendance column. Absent should be marked as **'A'in Red Ink**.

Signature of Company Internship supervisor with (company stamp/ seal)

(Name____) Contact No.

ATTENDANCE SHEET

Name and Address of Organization

___Symbiosis University of Applied Sciences _

________near Bada Bangadda_____________

______ Indore, Madhya Pradesh_453112_____

Name of Student	Harshit Agrawal
Enrollment No.	2016AB001031
Name of Course	Btech in CS-IT
Date of Commencement of Training	01-Jan-2020
Date of Completion of Training	15-May-2020

Signature of the student: Harshit

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Apr 2020	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	

Note:

1. Attendance Sheet should remain affixed in Daily Training Diary. **Do not remove or tear it off.**
2. Student should sign/initial in the attendance column. Do not mark 'P'.
3. Holidays should be marked in **Red Ink** in attendance column. Absent should be marked as **'A'in Red Ink**.

Signature of Company Internship supervisor with (company stamp/ seal)

(Name____) Contact No.

ATTENDANCE SHEET

Name and Address of Organization

___Symbiosis University of Applied Sciences _

________near Bada Bangadda______________

______ Indore, Madhya Pradesh 453112_____

Name of Student	Harshit Agrawal
Enrollment No.	2016AB001031
Name of Course	Btech in CS-IT
Date of Commencement of Training	01-Jan-2020
Date of Completion of Training	15-May-2020

Signature of the student: Harshit

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
May 2020	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]																

Note:

1. Attendance Sheet should remain affixed in Daily Training Diary. **Do not remove or tear it off.**
2. Student should sign/initial in the attendance column. Do not mark 'P'.
3. Holidays should be marked in **Red Ink** in attendance column. Absent should be marked as **'A'in Red Ink**.

Signature of Company Internship supervisor with (company stamp/ seal)

(Name_______________________________________) Contact No.