S. B. JAIN INSTITUTE OF TECHNOLOGY, MANAGEMENT & RESEARCH, NAGPUR.

Practical No. 08

Aim: Implement Convolution Neural Network for Deep Learning.

(CNN on CIFAR-10 dataset)

Name of Student : **Shrutika Pradeep Bagdi**

Roll No : **CS22130**

Semester/Year : **VIIth Sem / IVth Year**

Academic Session : **2025-2026 [ODD]**

Date of Performance : _____

Date of Submission : _____

Aim: Implement Convolution Neural Network for Deep Learning. (CNN on CIFAR-10 dataset)

OBJECTIVE/EXPECTED LEARNING OUTCOME:

The objectives and expected learning outcome of this practical are:

- It is computationally ineffective right. So here comes Convolutional Neural Network or CNN. In simple word what CNN does is, it extract the feature of image and convert it into lower dimension without loosing its characteristics.
- To develop learning algorithms for convolution neural networks.

THEORY:

An introductory look at Convolutional Neural Network with theory and code example. I want to write about one of the most important neural networks used in the field of deep learning, especially for image recognition and natural language processing: convolutional neural network, also called "CNN" or "ConvNet".

What is Convolutional neural network?

Convolutional neural networks. Sounds like a weird combination of biology and math with a little CS sprinkled in, but these networks have been some of the most influential innovations in the field of computer vision. 2012 was the first year that neural nets grew to prominence as Alex Krizhevsky used them to win that year's ImageNet competition (basically, the annual Olympics of computer vision), dropping the classification error record from 26% to 15%, an astounding improvement at the time. Ever since then, a host of companies have been using deep learning at the core of their services. Facebook uses neural nets for their automatic tagging algorithms, Google for their photo search, Amazon for their product recommendations, Pinterest for their home feed personalization, and Instagram for their search infrastructure.

Architecture of a Traditional CNN

A convolutional neural network is composed of at least 3 layers:

- **Convolution layer** to perform convolution operations and to generate many feature maps from one image.
- A **pooling layer** to denoise the feature maps by shrinking non-overlapping submatricesinto summary statistics (such as maximums).
- A **dense layer** which is a usual (shallow/deep) neural network that takes flattened inputs.

Working of Convolutional Neural Networks.

Convolutional neural networks are based on neuroscience findings. They are made of layers of artificial neurons called nodes. These nodes are functions that calculate the weighted sum of the inputs and return an activation map. This is the convolution part of the neural network.

Each node in a layer is defined by its weight values. When you give a layer some data, like an image, it takes the pixel values and picks out some of the visual features.

When you're working with data in a CNN, each layer returns activation maps. These maps point out important features in the data set. If you gave the CNN an image, it'll point out features based on pixel values, like colors, and give you an activation function.

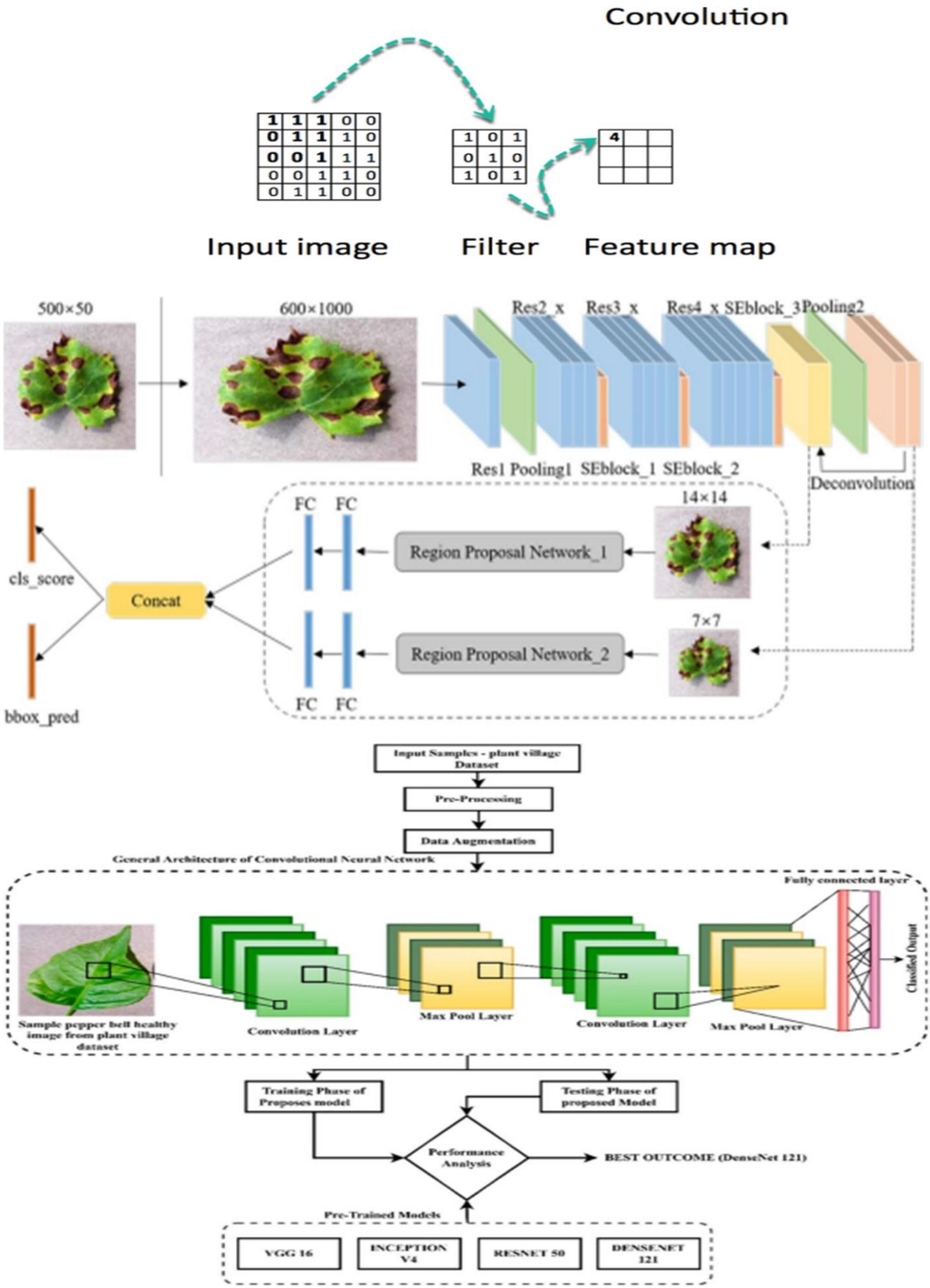
Usually with images, a CNN will initially find the edges of the picture. Then thisslight definition of the image will get passed to the next layer. Then that layer will start detecting things like corners and color groups. Then that image definition will get passed to the next layer and the cycle continues until a prediction is made. As the layers get more defined, this is called max pooling. It only returns the most relevant features from the layer in the activation map. This is what gets passed to each successive layer until you get the final layer.

The last layer of a CNN is the classification layer which determines the predicted value based on the activation map. If you pass a handwriting sample to a CNN, the classification layer will tell you what letter is in the image. This is what autonomous vehicles use to determine whether an object is another car, a person, or some other obstacle.

Training a CNN is similar to training many other machine learning algorithms. You'll start with some training data that is separate from your test data and you'll tune your weights based on the accuracy of the predicted values. Just be careful that you don't overfit your model.

Different types of CNN

- **1D CNN**: With these, the CNN kernel moves in one direction. 1D CNNs are usually used on time-series data.
- **2D CNN**: These kinds of CNN kernels move in two directions. You'll see these used with image labelling and processing.
- **3D CNN**: This kind of CNN has a kernel that moves in three directions. With this type of CNN, researchers use them on 3D images like CT scans and MRIs.


Convolution
Input image
Filter
Feature map
500×50
600×1000
Res2_x
Res3_x
Res4_x SEblock_3Pooling2
Res1 Pooling1 SEblock_1 SEblock_2
Deconvolution
14×14
7×7
FC
FC
Region Proposal Network_1
Region Proposal Network_2
cls_score
bbox_pred
Concat
Input Samples - plant village Dataset
Pre-Processing
Data Augmentation
General Architecture of Convolutional Neural Network
Fully connected layer
Classified Output
Sample pepper bell healthy image from plant village dataset
Convolution Layer
Max Pool Layer
Convolution Layer
Max Pool Layer
Training Phase of Proposes model
Testing Phase of proposed Model
Performance Analysis
BEST OUTCOME (DenseNet 121)
Pre-Trained Models
VGG 16
INCEPTION V4
RESNET 50
DENSENET 121

PROGRAM CODE:

```
from google.colab import drive
drive.mount("/content/drive")
import numpy as np
import pandas as pd
import tensorflow as tp
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPool2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Dense
(X_train,y_test) , (X_test,y_test) = cifar10.load_data()
print(X_train.shape)
print(X_test.shape)
X_train
plt.imshow(X_train[20])
plt.imshow(X_train[5])
for i in range(20):
  #subplot
  plt.subplot(5, 5, i+1)

  #plotting pixel data
  plt.imshow(X_train[i], cmap=plt.get_cmap('gray'))

  #show the figure

#Shrutika Bagdi (CS22130)
plt.show()
print(X_train.shape)
print(X_test.shape)
X_train[0]
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], X_train.shape[2], 3))
X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], X_test.shape[2], 3))
print(X_train.shape)
print(X_test.shape)
X_train[0]
X_train = X_train.astype('float32') / 255.0
X_test = X_test.astype('float32') / 255.0
X_train[0]
model = Sequential()
model.add(Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1)))
model.add(MaxPool2D(2,2))
model.add(Flatten())
```

```
model.add(Dense(100,activation='relu'))
model.add(Dense(10,activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
model.fit(X_train,y_train,epochs=5)
model.fit(X_test,y_test,epochs=5)
```

OUTPUT (SCREENSHOT):

```
[ ]
#Name:Shrutika Pradeep Bagdi_CS22130
from google.colab import drive
```

```
[ ]
#Name:Shrutika Pradeep Bagdi_CS22130
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

```
[ ]
#Shrutika Bagdi (CS22130)
import numpy as np
import pandas as pd
import tensorflow as tp
import matplotlib.pyplot as plt
```

```
[ ]
#Shrutika Bagdi (CS22130)
# importing the required libraries
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPool2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Dense
```

```
#loading data
(X_train,y_test) , (X_test,y_test) = cifar10.load_data()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 ━━━━━━━━━━━━━━━━━━━━ 3s 0us/step
```

```
#Shrutika Bagdi (CS22130)
print(X_train.shape)
print(X_test.shape)
```

```
(50000, 32, 32, 3)
(10000, 32, 32, 3)
```

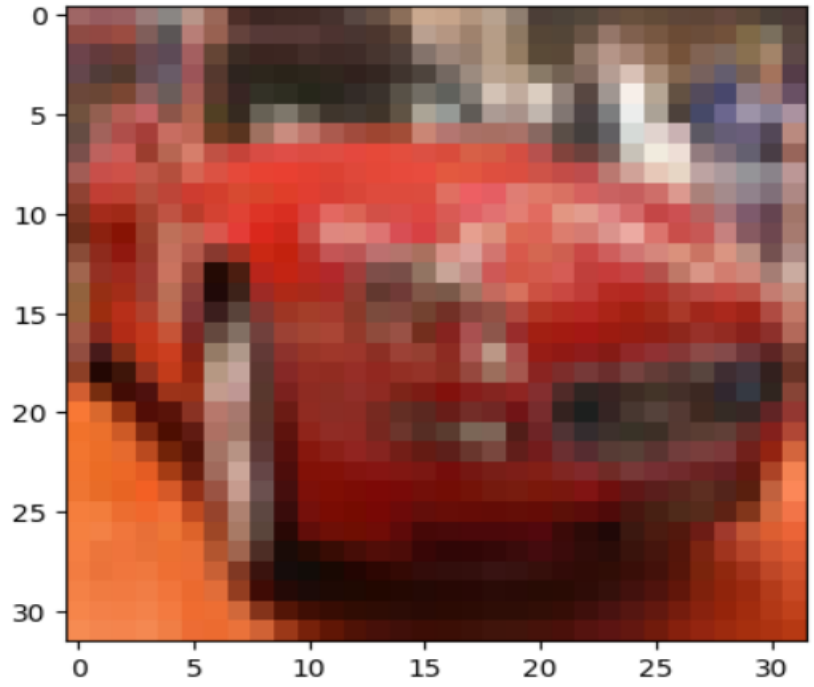
```
#Shrutika Bagdi (CS22130)
X_train
```

```
array([[[[ 59,  62,  63],
         [ 43,  46,  45],
         [ 50,  48,  43],
         ...,
         [158, 132, 108],
         [152, 125, 102],
         [148, 124, 103]],

        [[ 16,  20,  20],
         [  0,   0,   0],
         [ 18,   8,   0],
```

```
#Shrutika Bagdi (CS22130)
plt.imshow(X_train[5])
```

```
<matplotlib.image.AxesImage at 0x7f828e1d0350>
```

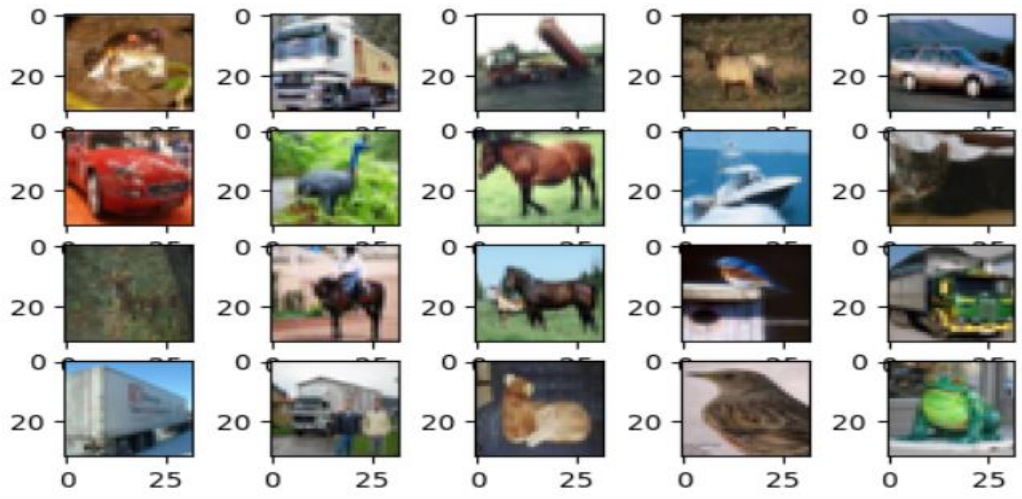



```
[ ]
#Shrutika Bagdi (CS22130)
for i in range(20):
  #subplot
  plt.subplot(5, 5, i+1)

  #plotting pixel data
  plt.imshow(X_train[i], cmap=plt.get_cmap('gray'))

  #show the figure

#Shrutika Bagdi (CS22130)
plt.show()
```



```
#Shrutika Bagdi (CS22130)
print(X_train.shape)
print(X_test.shape)
```

```
(50000, 32, 32, 3)
(10000, 32, 32, 3)
```

```
#Shrutika Bagdi (CS22130)
X_train[0]
```

ndarray (32, 32, 3) show data

```
#Shrutika Bagdi (CS22130)
#reshaping data
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], X_train.shape[2], 3))
X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], X_test.shape[2], 3))
```

```
#Shrutika Bagdi (CS22130)
#checking the shape after reshaping
print(X_train.shape)
print(X_test.shape)
```

```
(50000, 32, 32, 3)
(10000, 32, 32, 3)
```

```
#Shrutika Bagdi (CS22130)
#checking the represrntation of image after flattening
X_train[0]
```

ndarray (32, 32, 3) show data

```
#Shrutika Bagdi (CS22130)
#normalizing the pixel values
X_train = X_train.astype('float32') / 255.0
X_test = X_test.astype('float32') / 255.0
```

```
#Shrutika Bagdi (CS22130)
#defining model
model = Sequential()
```

```
#Shrutika Bagdi (CS22130)
#adding convolution layer
model.add(Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1)))
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:113: UserWarnir
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
#Shrutika Bagdi (CS22130)
#adding pooling layer
model.add(MaxPool2D(2,2))
```

```
#Shrutika Bagdi (CS22130)
#adding fully connected layer
#Dense=Nodes are connected with each other
model.add(Flatten())
model.add(Dense(100,activation='relu'))
```

```
#Shrutika Bagdi (CS22130)
#adding output layer
#softmax = give Probability
model.add(Dense(10,activation='softmax'))
```

```
[ ]  #Shrutika Bagdi (CS22130)
     #Spliting the model
     model.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=['accuracy'])

[ ]  (X_train, y_train), (X_test, y_test) = cifar10.load_data()

[ ]  #Shrutika Bagdi (CS22130)
     #fitting the model
     model.fit(X_train,y_train,epochs=5)

     Epoch 1/5
     1563/1563 ━━━━━━━━━━━━━━━━━━━━ 49s 31ms/step - accuracy: 0.3884 - loss: 1.6901
     Epoch 2/5
     1563/1563 ━━━━━━━━━━━━━━━━━━━━ 79s 29ms/step - accuracy: 0.5661 - loss: 1.2267
     Epoch 3/5
     1563/1563 ━━━━━━━━━━━━━━━━━━━━ 46s 29ms/step - accuracy: 0.6107 - loss: 1.1088
     Epoch 4/5
     1563/1563 ━━━━━━━━━━━━━━━━━━━━ 82s 29ms/step - accuracy: 0.6409 - loss: 1.0299
     Epoch 5/5
     1563/1563 ━━━━━━━━━━━━━━━━━━━━ 82s 29ms/step - accuracy: 0.6649 - loss: 0.9625
     <keras.src.callbacks.history.History at 0x7f823d28be10>

[ ]  #Shrutika Bagdi (CS22130)
     #fitting the model
     model.fit(X_test,y_test,epochs=5)

     Epoch 1/5
     313/313 ━━━━━━━━━━━━━━━━━━━━ 10s 31ms/step - accuracy: 0.6157 - loss: 1.1126
     Epoch 2/5
     313/313 ━━━━━━━━━━━━━━━━━━━━ 8s 25ms/step - accuracy: 0.6522 - loss: 0.9866
     Epoch 3/5
     313/313 ━━━━━━━━━━━━━━━━━━━━ 11s 27ms/step - accuracy: 0.7033 - loss: 0.8682
     Epoch 4/5
     313/313 ━━━━━━━━━━━━━━━━━━━━ 10s 31ms/step - accuracy: 0.7189 - loss: 0.8286
     Epoch 5/5
     313/313 ━━━━━━━━━━━━━━━━━━━━ 8s 25ms/step - accuracy: 0.7378 - loss: 0.7656
     <keras.src.callbacks.history.History at 0x7f823c91f290>
```

CONCLUSION:

_____

_____

_____

_____

_____

DISCUSSION AND VIVA VOCE:

1. Why do we prefer CNNs over fully connected networks for image tasks?
2. What is the purpose of the convolution operation in CNN?
3. Define kernel (or filter) in CNN.

4. What is the difference between stride and padding?
5. What is the difference between Max Pooling and Average Pooling?
6. What is the role of the ReLU activation function in CNNs?
7. What is the difference between feature extraction and classification in CNNs?
8. What are some common applications of CNNs?
9. What is the vanishing gradient problem, and how is it handled in CNNs?
10. What is transfer learning in CNNs?

REFERENCE:

- https://towardsdatascience.com/convolutional-neural-networks-cnns-a-practical-perspective c7b3b2091aa8
- https://en.wikipedia.org/wiki/Convolutional_neural_network#References