

Project Report on
IMAGE CLASSIFICATION Using CNN

Submitted to
The Department of Computer Science and Engineering
In partial fulfillment of the academic requirements of
Jawaharlal Nehru Technological University
For
The award of the degree of

Bachelor of Technology
in
Computer Science and Engineering

By

D.Madhan (18311A05L9)

N.Shashank (18311A05Q0)

T.Karthik (19315A0519)

Under the Guidance of
Ms. Neha
Assistant Professor



Sreenidhi Institute of Science and Technology
Yamnampet, Ghatkesar, R.R. District, Hyderabad - 501301

Affiliated to
Jawaharlal Nehru Technology University
Hyderabad - 500085
Department of Computer Science and Engineering
Sreenidhi Institute of Science and Technology



CERTIFICATE

This is to certify that this Project report on ” **IMAGE CLASSIFICATION Using CNN**”, submitted by D.Madhan(18311A0L9),N.Shashank(18311A0Q0),T.Karthik(19315A0519) in the year 2021 in partial fulfillment of the academic requirements of Jawaharlal Nehru Technological University for the award of the degree of Bachelor of Technology in Computer Science and Engineering, is a bonafide work that has been carried out by them as part of their Academic Minor Project(2021), under our guidance. This report has not been submitted to any other institute or university for the award of any degree.

Ms.Neha
Assistant Professor
Department of CSE
Internal Guide

Dr. Aruna Varanasi
Professor & HOD
Department of CSE
Head of Department

Mr. NV Subba Reddy
Associate Professor
Department of CSE
Project Coordinator

External Examiner

DECLARATION

We, **D.MADHAN(18311A05L9),N.SHASHANK(18311A05Q0),T.KARTHIK(19315A0519)**, students of **SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY, YAMNAMPET, GHATKESAR, of COMPUTER SCIENCE AND ENGINEERING** solemnly declare that the project work, titled “**IMAGE CLASSIFICATION Using CNN**” is submitted to **SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY** for partial fulfillment for the award of degree of Bachelor of technology in **COMPUTER SCIENCE AND ENGINEERING**.

It is declared to the best of our knowledge that the work reported does not form part of any dissertation submitted to any other University or Institute for award of any degree.

ACKNOWLEDGEMENT

I would like to express my gratitude to all the people behind the screen who helped me to transform an idea into a real application.

I would like to express my heart-felt gratitude to my parents without whom I would not have been privileged to achieve and fulfill my dreams. I am grateful to our principal, **Dr. T. Ch. Siva Reddy**, who most ably run the institution and has had the major hand in enabling me to do my project.

I profoundly thank **Dr. Aruna Varanasi**, Head of the Department of Computer Science & Engineering who has been an excellent guide and also a great source of inspiration to my work.

I would like to thank my internal guide **Ms.NEHA** for her technical guidance, constant encouragement and support in carrying out my project at college.

The satisfaction and euphoria that accompany the successful completion of the task would be great but incomplete without the mention of the people who made it possible with their constant guidance and encouragement crowns all the efforts with success. In this context, I would like thank all the other staff members, both teaching and non-teaching, who have extended their timely help and eased my task.

D.Madhan (18311A05L9)

N.Shashank(18311A05Q0)

T.Karthik (19315A0519)

ABSTRACT

Image classification is the most critical use case in digital image analysis. The CNN is a type of Deep Neural Networks (DNN) that consists of many layers such as the Conv layers, Pooling layer, and the fully-connected layer.

Convolutional neural networks (CNNs) are widely used in pattern- and image-recognition problems as they have a number of advantages compared to other techniques.

CIFAR-10 is a very popular computer vision dataset. This dataset is well studied in many types of deep learning research for object recognition.

This dataset consists of 60,000 images divided into 10 target classes, with each category containing 6000 images of shape 32*32. This dataset contains images of low resolution (32*32), which allows researchers to try new algorithms. The 10 different classes of this dataset are:

1. Airplane
2. Car
3. Bird
4. Cat
5. Deer
6. Dog
7. Frog
8. Horse
9. Ship
10. Truck

CIFAR-10 dataset is already available in the datasets module of Keras. We do not need to download it; we can directly import it from `keras.datasets`

INDEX

1. INTRODUCTION	9
1.1 Scope And Proposed Model	9
2. REQUIREMENTS	9
2.1 Software requirements	9
2.2 Hardware Requirements	10
2.3 Packages Requirements	10
3.PYTHON TECHNOLOGY & LIBRARIES	10
3.1 About python:	11
3.2 Understanding an Artificial Neural Network (ANN):	11
3.3 Practical Applications for Artificial Neural Networks (ANNs):	12
3.4 The Convolutional Neural Networks	12
3.5 Numpy:	13
3.6 PANDAS:	14
3.7 Tensor Flow:	16
3.8 KERAS:	17
3.9 UNIFIED MODELING LANGUAGE	18-22
3.10 UML Diagrams	23-26

4.How CNN's Work ?	27-31
5. Walking through the model:	32
5.1 Importing libraries:	32
5.2 Load data and split	35
5.4.One hot encoding	36
5.5. Initializing Sequential Model	37
5.6. Adding Layers	37
5.7. Adding an Optimizer	39
5.8. Compiling and model summary	40
5.9. Train the model	41
5.10.Evaluation metrics	43
5.11.Predictions	43
6. CONCLUSION	44
PO's AND PSO's	45

1. INTRODUCTION

Image classification is the ability to take an image and then successfully classify it into one of a given set of classes. The goal of our work will be to create a model that will be able to identify and classify images accurately.

This project utilizes Convolutional Neural Networks in order to successfully train a model. The dataset used is the Cifar10 dataset.

The Cifar10 dataset imported from keras has the 10 classes and we built a model such that our model predicts the given image with a far decent accuracy without any overfitting or underfitting issues.

1.1 Scope And Proposed Model

- This gives the straight forward approach for identifying different images from the dataset.
- The proposed system has been improved with the both training accuracy and test accuracy without overfitting.
- The proposed system had been optimized with the Adam optimizer for the better results.
- The images can be predicted from the dataset using the Cnn algorithm at a greater ease.
- It improves the quality of image classification.
- The proposed system can be easily adapted to a wide variety of fields, science and technology.

2. REQUIREMENTS

The requirements for the building of the model are

2.1 Software Requirements:

- **Operating System:** Microsoft Windows/Mac-Os/linux
- **Web browser:** Google Chrome (Google Colab)
- **Python Ide:** Jupyter notebook

2.2 Hardware Requirements:

- **Processor** : Intel i5 and more
- **RAM** : Min 4GB

2.3 Packages Requirements :

- Numpy
- Pandas
- Matplotlib
- Keras
- Seaborn
- Tensorflow
- Keras

3.PYTHON TECHNOLOGY & LIBRARIES

3.1 About python:

Python is an interpreted, high-level and general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was created in the late 1980s as a successor to the ABC language. Python 2.0, released in 2000, introduced features like list comprehensions and a garbage collection system with reference counting.

Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3.

The Python 2 language was officially discontinued in 2020 (first planned for 2015), and "Python 2.7.18 is the last Python 2.7 release and therefore the last Python 2 release." No more security patches or other improvements will be released for it. With Python 2's end-of-life, only Python 3.6.x and later are supported.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, a free and open-source reference implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development.

3.2 Understanding an Artificial Neural Network (ANN):

Artificial neural networks are built like the human brain, with neuron nodes interconnected like a web. The human brain has hundreds of billions of cells called neurons. Each neuron is made up of a cell body that is responsible for processing information by carrying information towards (inputs) and away (outputs) from the brain.

An ANN has hundreds or thousands of artificial neurons called processing units, which are interconnected by nodes. These processing units are made up of input and output units. The input units receive various forms and structures of information based on an internal weighting system, and the neural network attempts to learn about the information presented to produce one output report. Just like humans need rules and guidelines to come up with a result or output, ANNs also use a set of learning rules called backpropagation, an abbreviation for backward propagation of error, to perfect their output results.

An ANN initially goes through a training phase where it learns to recognize patterns in data, whether visually, aurally, or textually. During this supervised phase, the network compares its actual output produced with what it was meant to produce—the desired output. The difference between both outcomes is adjusted using backpropagation. This means that the network works backward, going from the output unit to the input units to adjust the weight of its connections between the units until the difference between the actual and desired outcome produces the lowest possible error.

During the training and supervisory stage, the ANN is taught what to look for and what its output should be, using yes/no question types with binary numbers. For example, a bank that wants to detect credit card fraud on time may have four input units fed with these questions: (1) Is the transaction in a different country from the user's resident country? (2) Is the website the card is being used at affiliated with companies or countries on the bank's watch list? (3) Is the transaction amount larger than \$2,000? (4) Is the name on the transaction bill the same as the name of the cardholder?

The bank wants the "fraud detected" responses to be Yes Yes Yes No, which in binary format would be 1 1 1 0. If the network's actual output is 1 0 1 0, it adjusts its results until it delivers an output that coincides with 1 1 1 0. After training, the computer system can alert the bank of pending fraudulent transactions, saving the bank lots of money.

3.3 Practical Applications for Artificial Neural Networks (ANNs):

Artificial neural networks are paving the way for life-changing applications to be developed for use in all sectors of the economy. Artificial intelligence platforms that are built on ANNs are disrupting the traditional ways of doing things. From translating web pages into other languages to having a virtual assistant order groceries online to conversing with chatbots to solve problems, AI platforms are simplifying transactions and making services accessible to all at negligible costs.

Artificial neural networks have been applied in all areas of operations. Email service providers use ANNs to detect and delete spam from a user's inbox; asset managers use it to forecast the direction of a company's stock; credit rating firms use it to improve their credit scoring methods; e-commerce platforms use it to personalize recommendations to their audience; chatbots are developed with ANNs for natural language processing; deep learning algorithms use ANN to predict the likelihood of an event; and the list of ANN incorporation goes on across multiple sectors, industries, and countries.

3.4 The Convolutional Neural Networks

In the past few decades, Deep Learning has proved to be a very powerful tool because of its ability to handle large amounts of data. The interest to use hidden layers has surpassed traditional techniques, especially in pattern recognition. One of the most popular deep neural networks is Convolutional Neural Networks.

Since the 1950s, the early days of AI, researchers have struggled to make a system that can understand visual data. In the following years, this field came to be known as Computer Vision. In 2012, computer vision took a quantum leap when a group of researchers from the University of Toronto developed an AI model that surpassed the best image recognition algorithms and that too by a large margin.

The AI system, which became known as AlexNet (named after its main creator, Alex Krizhevsky), won the 2012 ImageNet computer vision contest with an amazing 85 percent accuracy. The runner-up scored a modest 74 percent on the test.

At the heart of AlexNet was Convolutional Neural Networks a special type of neural network that roughly imitates human vision. Over the years CNNs have become a very important part of many Computer Vision applications.

3.5 Numpy:

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays, requiring rewriting some code, mostly inner loops, using NumPy.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars. In comparison, MATLAB boasts a large number of additional toolboxes, notably Simulink, whereas NumPy is intrinsically integrated with Python, a more modern and complete programming language. Moreover, complementary Python packages are available; SciPy is a library that adds more MATLAB-like functionality and

Matplotlib is a plotting package that provides MATLAB-like plotting functionality. Internally, both MATLAB and NumPy rely on BLAS and LAPACK for efficient linear algebra computations.

Python bindings of the widely used computer vision library OpenCV utilize NumPy arrays to store and operate on data. Since images with multiple channels are simply represented as three-dimensional arrays, indexing, slicing or masking with other arrays are very efficient ways to access specific pixels of an image. The NumPy array as universal data structure in OpenCV for images, extracted feature points, filter kernels and many more vastly simplifies the programming workflow and debugging.

3.6 PANDAS:

pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language. It is already well on its way toward this goal.

pandas is well suited for many different kinds of data:

Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet

Ordered and unordered (not necessarily fixed-frequency) time series data.

Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels

Any other form of observational / statistical data sets. The data actually need not be labeled at all to be placed into a pandas data structure

The two primary data structures of pandas, Series (1-dimensional) and DataFrame (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering. For R users, DataFrame provides everything that R's data.frame provides and much more. pandas is built on top of

NumPy and is intended to integrate well within a scientific computing environment with many other 3rd party libraries.

Here are just a few of the things that pandas does well:

Easy handling of missing data (represented as NaN) in floating point as well as non-floating point data

Size mutability: columns can be inserted and deleted from DataFrame and higher dimensional objects

Automatic and explicit data alignment: objects can be explicitly aligned to a set of labels, or the user can simply ignore the labels and let Series, DataFrame, etc. automatically align the data for you in computations

Powerful, flexible group by functionality to perform split-apply-combine operations on data sets, for both aggregating and transforming data

Make it easy to convert ragged, differently-indexed data in other Python and NumPy data structures into DataFrame objects

Intelligent label-based slicing, fancy indexing, and subsetting of large data sets

Intuitive merging and joining data sets

Flexible reshaping and pivoting of data sets

Hierarchical labeling of axes (possible to have multiple labels per tick)

Robust IO tools for loading data from flat files (CSV and delimited), Excel files, databases, and saving / loading data from the ultrafast HDF5 format

Time series-specific functionality: date range generation and frequency conversion, moving window statistics, date shifting and lagging.

Many of these principles are here to address the shortcomings frequently experienced using other languages / scientific research environments. For data scientists, working with data is typically divided into multiple stages: munging and cleaning data, analyzing / modeling it, then organizing the results of the analysis into a form suitable for plotting or tabular display. pandas is the ideal tool for all of these tasks.

3.7 Tensor Flow:

TensorFlow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

Tensorflow is a symbolic math library based on dataflow and differentiable programming. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache License 2.0 in 2015.

TensorFlow is Google Brain's second-generation system. Version 1.0.0 was released on February 11, 2017. While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units). TensorFlow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

TensorFlow computations are expressed as stateful dataflow graphs. The name TensorFlow derives from the operations that such neural networks perform on multidimensional data arrays, which are referred to as tensors. During the Google I/O Conference in June 2016, Jeff Dean stated that 1,500 repositories on GitHub mentioned TensorFlow, of which only 5 were from Google.

In December 2017, developers from Google, Cisco, RedHat, CoreOS, and CaiCloud introduced Kubeflow at a conference. Kubeflow allows operation and deployment of TensorFlow on Kubernetes.

In March 2018, Google announced TensorFlow.js version 1.0 for machine learning in JavaScript.

In Jan 2019, Google announced TensorFlow 2.0. It became officially available in Sep 2019.

In May 2019, Google announced TensorFlow Graphics for deep learning in computer graphics.

3.8 KERAS:

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research.

Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel.

In addition to standard neural networks, Keras has support for convolutional and recurrent neural networks. It supports other common utility layers like dropout, batch normalization, and pooling.

Keras allows users to productize deep models on smartphones (iOS and Android), on the web, or on the Java Virtual Machine. It also allows use of distributed training of deep-learning models on clusters of Graphics processing units (GPU) and tensor processing units (TPU).

3.9 UNIFIED MODELING LANGUAGE

The Unified Modeling Language (UML) is a general-purpose visual modeling language that is used to specify, visualize, construct, and document the artifacts of a software system. It captures decisions and understanding about systems that must be constructed. It is used to understand, design, browse, configure, maintain, and control information about such systems. It is intended for use with all development methods, lifecycle stages, application domains, and media. The modeling language is intended to unify past experience about modeling techniques and to incorporate current software best practices into a standard approach. UML includes semantic concepts, notation, and guidelines. It has static, dynamic, environmental, and organizational parts. It is intended to be supported by interactive visual modeling tools that have code generators and report writers. The UML specification does not define a standard process but is intended to be useful with an iterative development process. It is intended to support most existing object oriented development processes.

The UML captures information about the static structure and dynamic behavior of a system. A system is modeled as a collection of discrete objects that interact to perform work that ultimately benefits an outside user. The static structure defines the kinds of objects important to a system and to its implementation, as well as the relationships among the objects. The dynamic behavior defines the history of objects over time and the communications among objects to accomplish goals.

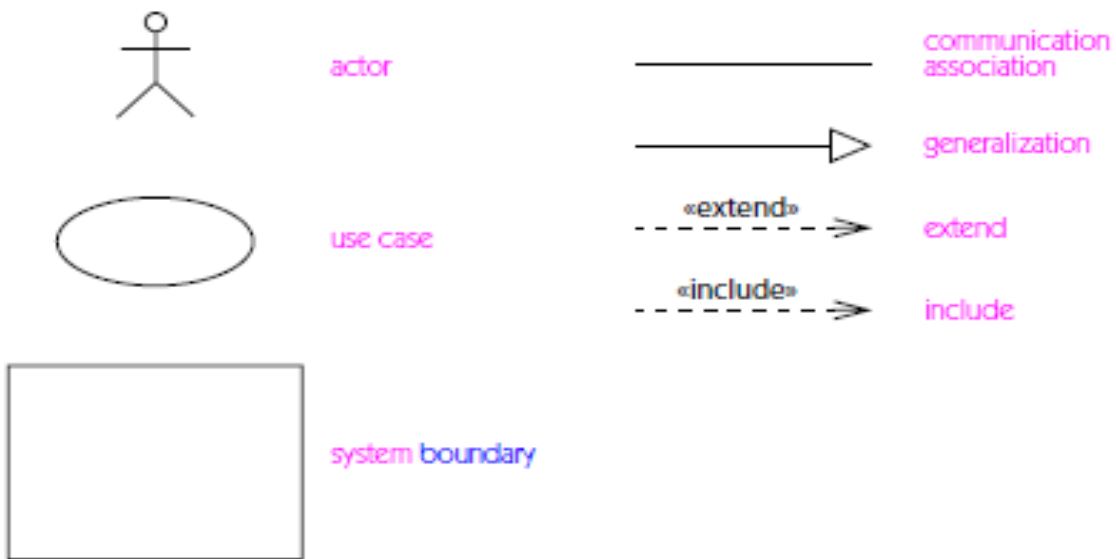
Modeling a system from several separate but related viewpoints permits it to be understood for different purposes.

The UML also contains organizational constructs for arranging models into packages that permit software teams to partition large systems into workable pieces, to understand and control dependencies among the packages, and to manage the versioning of model units in a complex development environment. It contains constructs for representing implementation decisions and for organizing run-time elements into components.

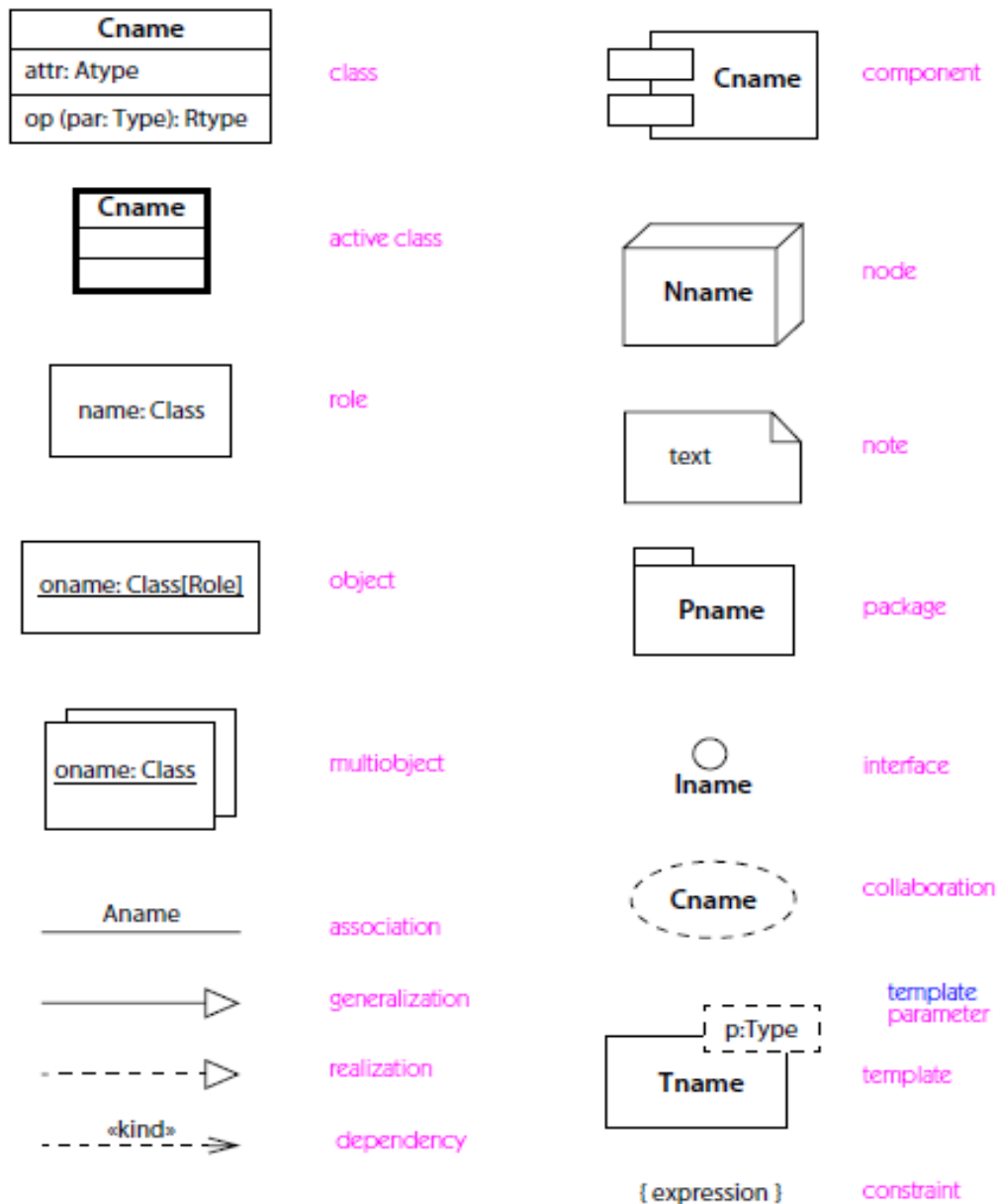
UML is not a programming language. Tools can provide code generators from UML into a variety of programming languages, as well as construct reverseengineered models from existing programs. The UML is not a highly formal language intended for theorem proving. There are a number of such languages, but they are not easy to understand or to use for most purposes. The UML is a general-purpose modeling language. For specialized domains, such as GUI layout, VLSI circuit design, or rule-based artificial intelligence, a more specialized tool with a special language might be appropriate. UML is a discrete modeling language.

It is not intended to model continuous systems such as those found in engineering and physics. UML is intended to be a universal general-purpose modeling language for discrete systems such as those made of software, firmware, or digital logic.

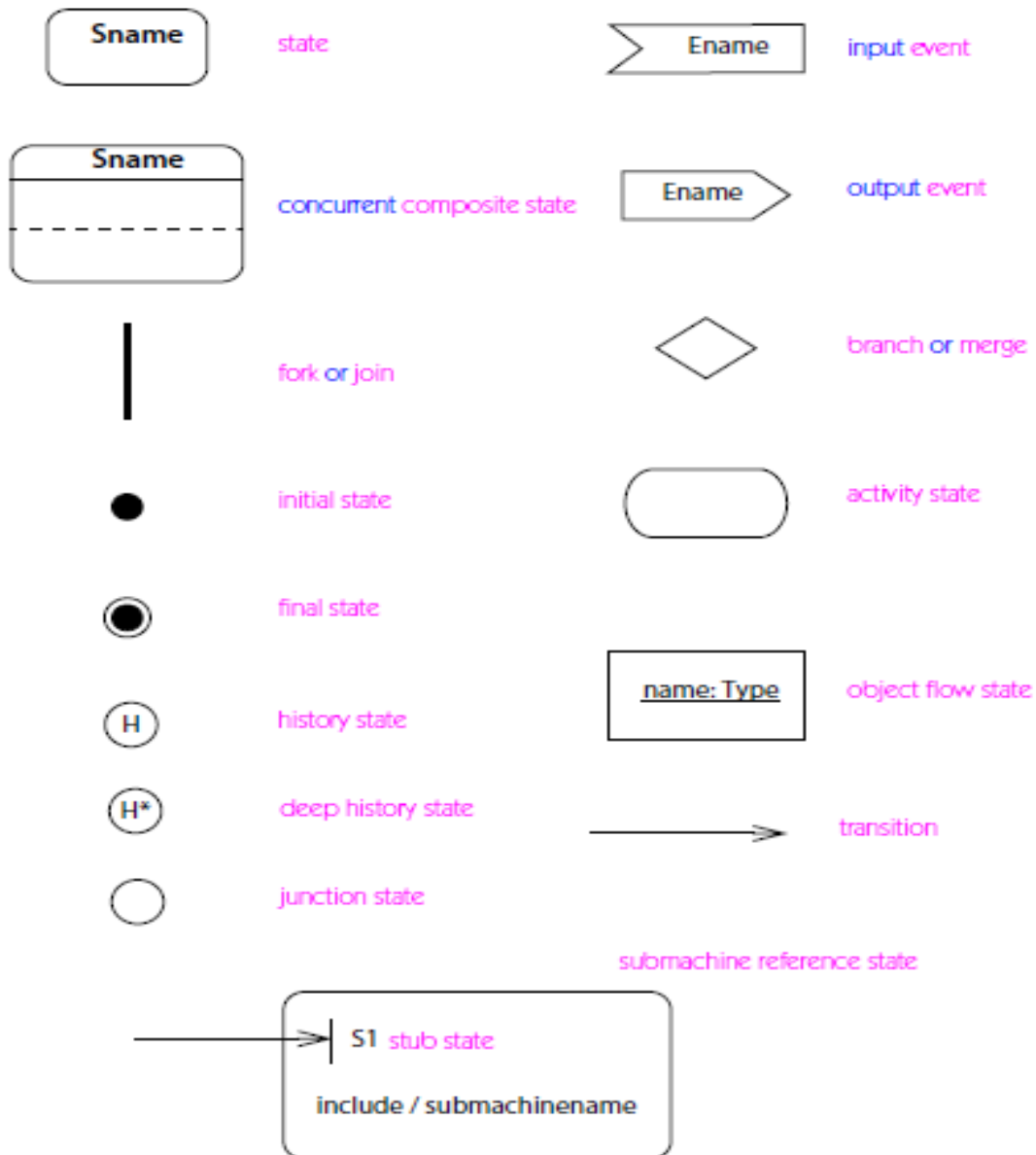
Icons on use case diagrams



Icons on class, component, deployment, and collaboration diagrams



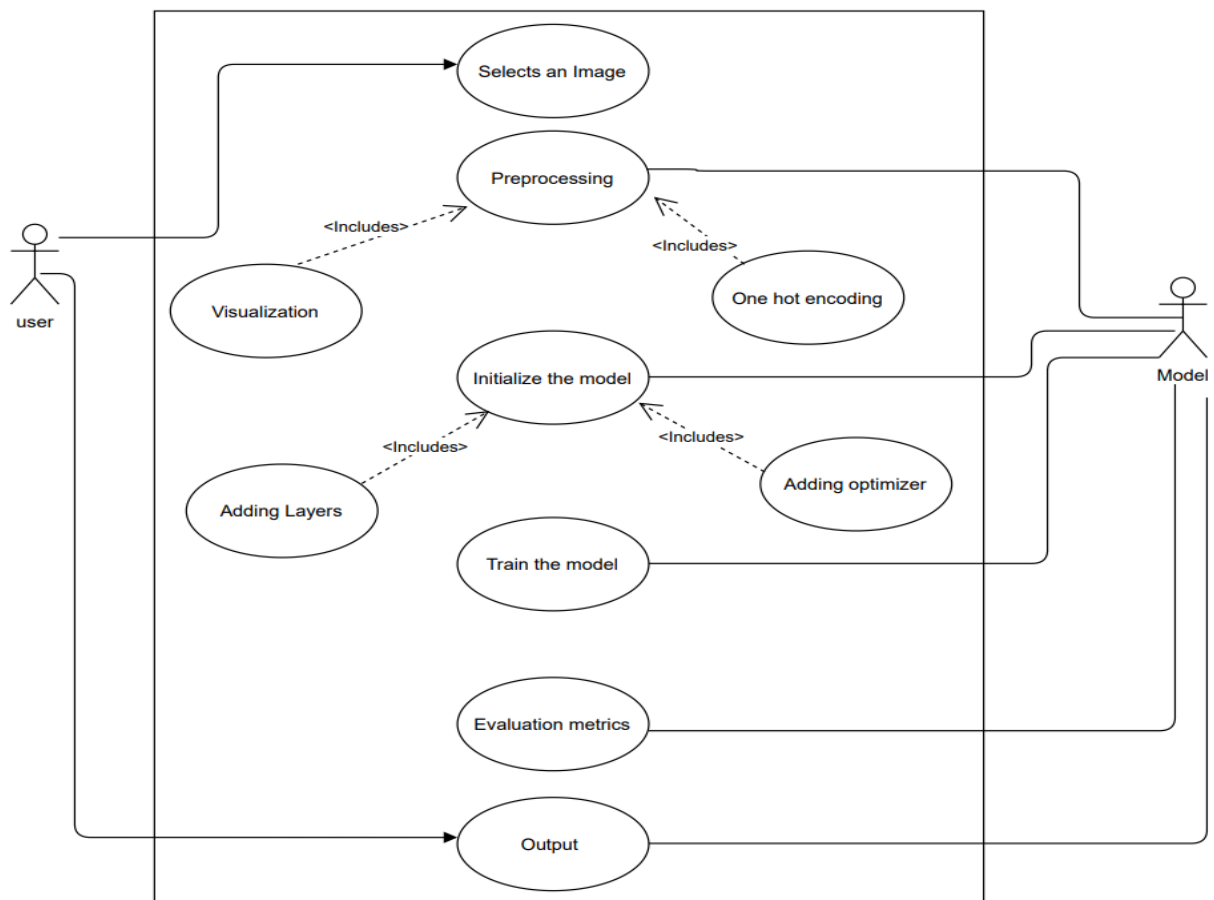
Icons on statechart and activity diagrams



3.10 UML Diagrams:

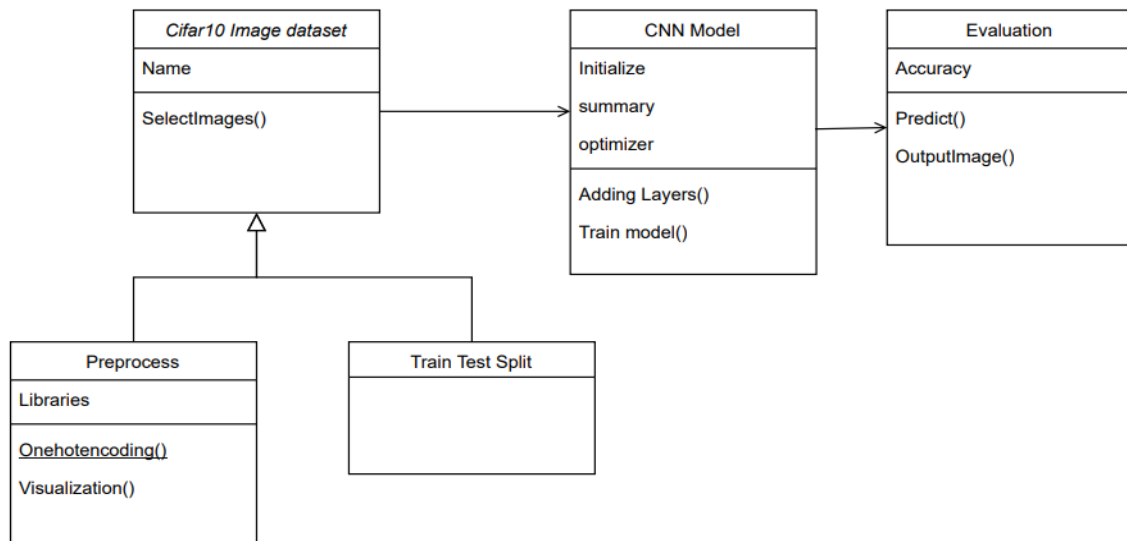
Uml diagrams will help the software and analysis easy to understand . uml diagrams capture the entire software into 9 diagrams or views . these 9 diagrams will helps to understand the analysis and better to implement them

3.10.1 Use case diagram:



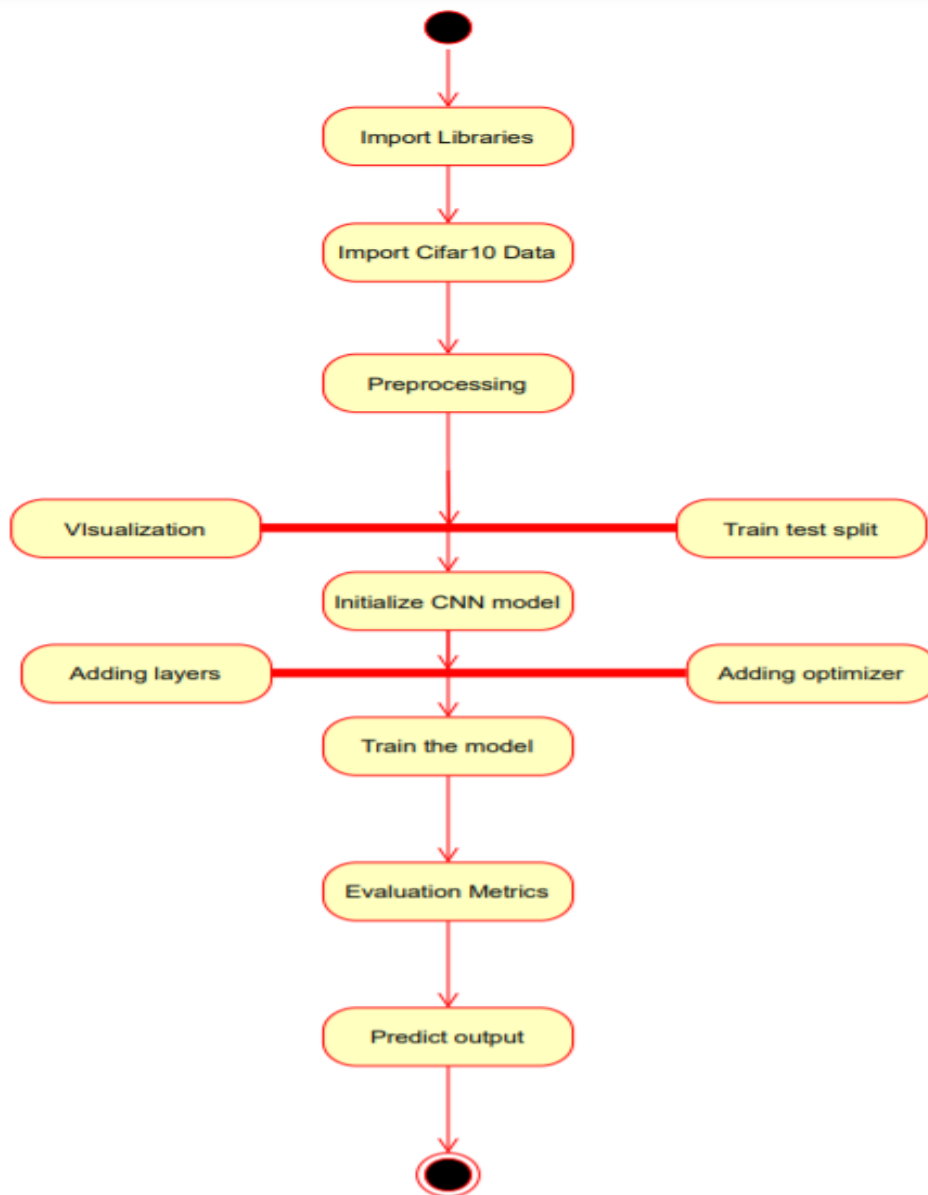
Use case diagram is basically capture the behaviour of analysis and helps to understand which person do which tasks and how tasks are related to each other

3.10.2 Class Diagram:



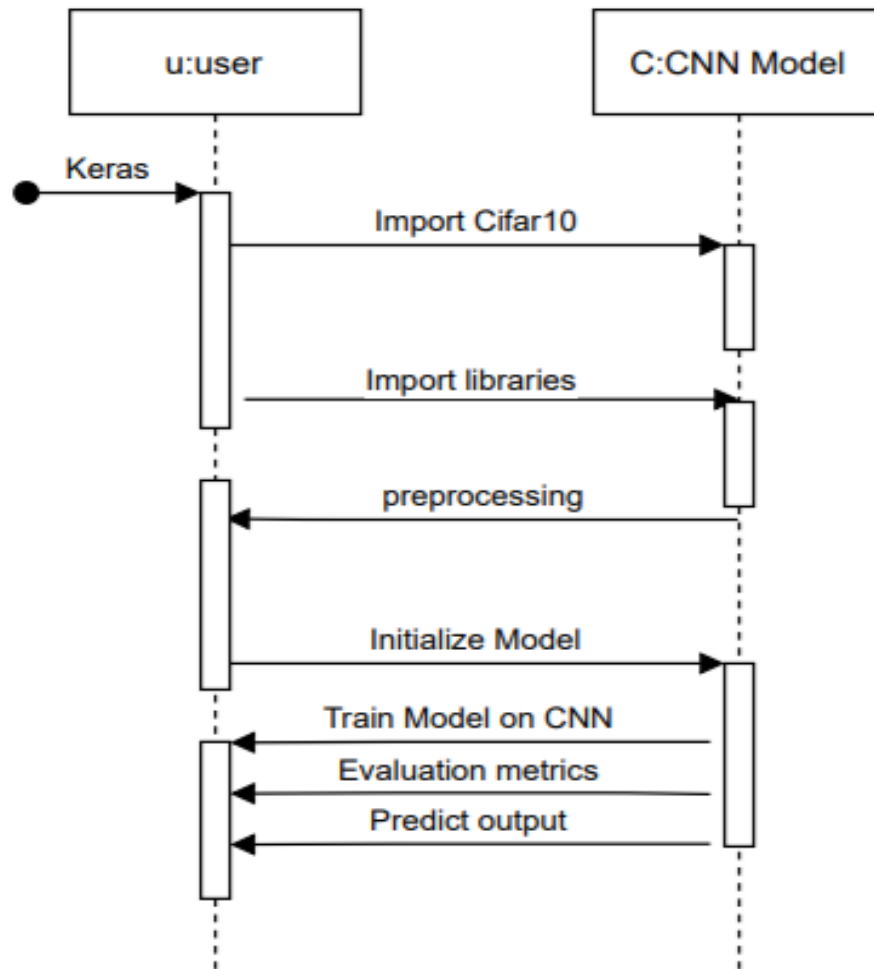
Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

3.10.4 Activity Diagram



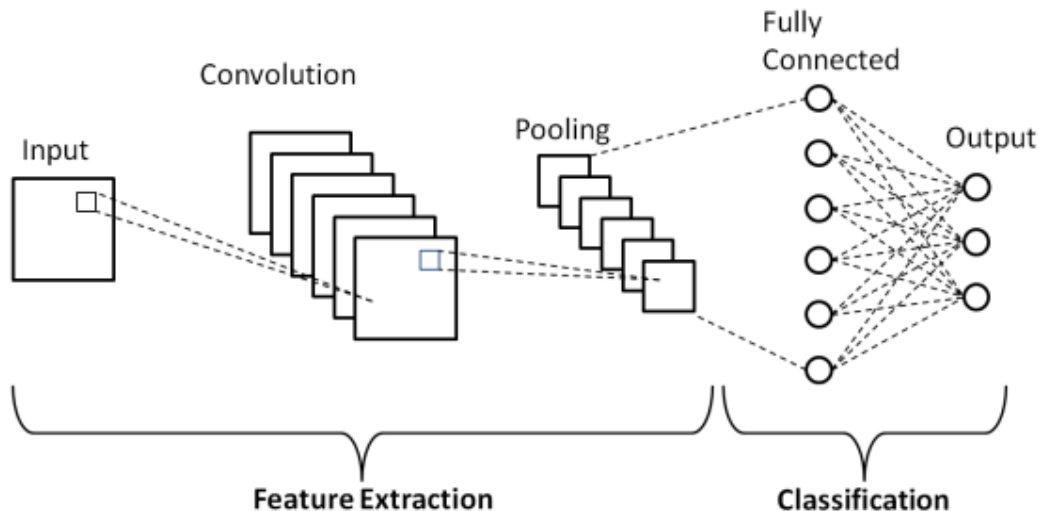
Activity diagram is basically a flowchart to represent the flow from one activity to another activity

3.10.5 Sequence Diagram:



A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together.

How CNN's Work ?



The Convolutional Neural Networks contains various layers included in it and the input image has to pass through these layers to have a desired output.

The layers included are:

1. Convolutional layer
2. Pooling layer
3. Fully connected layer

4.1. The Convolutional Layer:

It is the place where the convolution operation takes place.

Filter / kernel : A matrix which contains weights , which are learnt through back propagation , used to detect the edges and features from a given input image.

Feature Map: output generated when a filter is convolved over input image.

Convolution operation: This technique involves the process of convolving a given filter over an input image to attain feature map. This is what happens in the convolutional layer.

0	0	0	0	0	0	0
0	2	4	9	1	4	0
0	2	1	4	4	6	0
0	1	1	2	9	2	0
0	7	3	5	1	3	0
0	2	3	4	8	5	0
0	0	0	0	0	0	0

Image

×

1	2	3
-4	7	4
2	-5	1

Filter /
Kernel

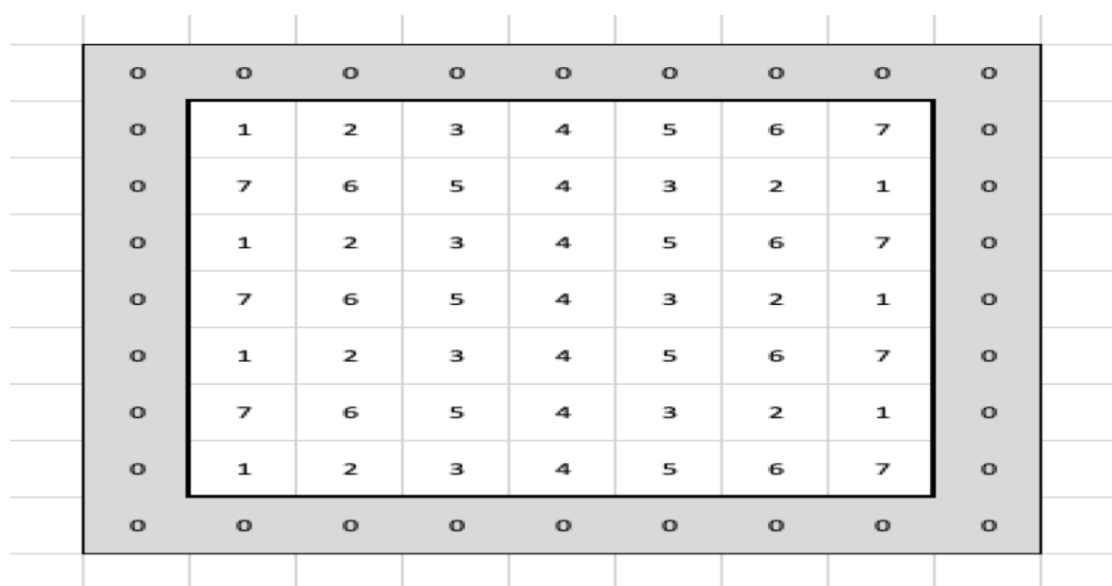
=

21	59	37	-19	2
30	51	66	20	43
-14	31	49	101	-19
59	15	53	-2	21
49	57	64	76	10

Feature

Padding: Adding an extra layer throughout the circumference of our original image to retain it's features without being shrinked, by convention we pad with zeros.

It is a hyperparameter which we need to tune.



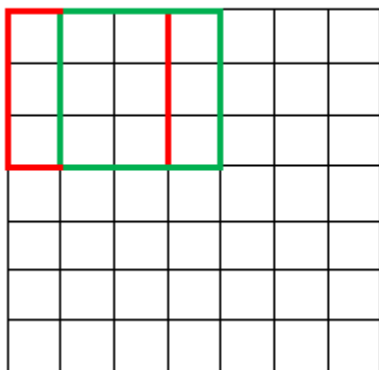
Reasons for why we go for padding :

1. As the convolution operation happens repeatedly there is a problem of our input image being shrunked , so it's original from will be lost till it reaches the last step.
2. When filter is convolved over an input image , then only once the top left corner is occurred and pixel in the middle would come several times, this leads to throwaway of information from corners.

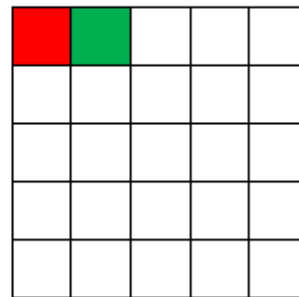
STRIDE: Stride is the amount by which the kernel is moved by as the kernel is passed over the image.

Stride in this context means the step of the convolution operation, it is also a hyperparameter.

7 x 7 Input Volume



5 x 5 Output Volume



4.2.The Pooling Layer

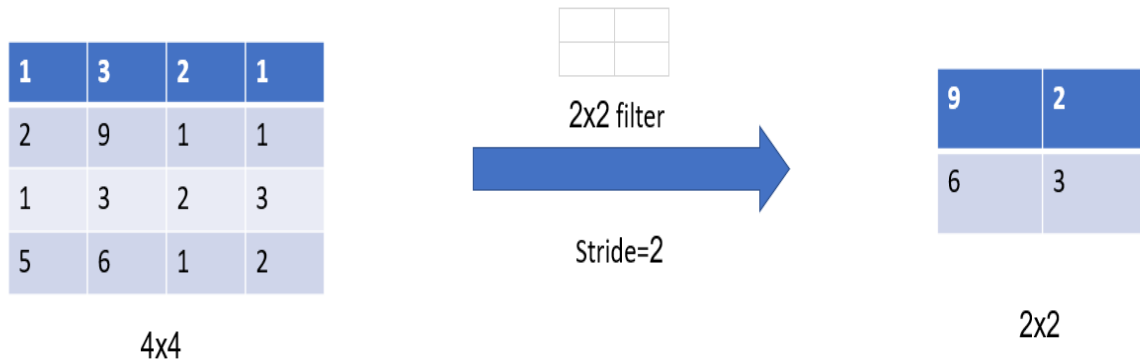
Other than convolutional layer , ConvNets often also use pooling layers

- To reduce the size of the representation.
- To speed the computation.
- Make some of the features that detects a bit more robust.

There can be various types of pooling but two of them are most commonly known by all:

- Max pooling
- Average pooling

Example for max pooling:

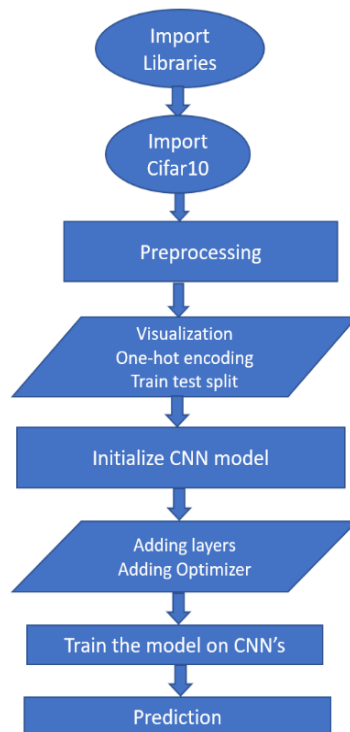


4.3.Fully Connected Layer

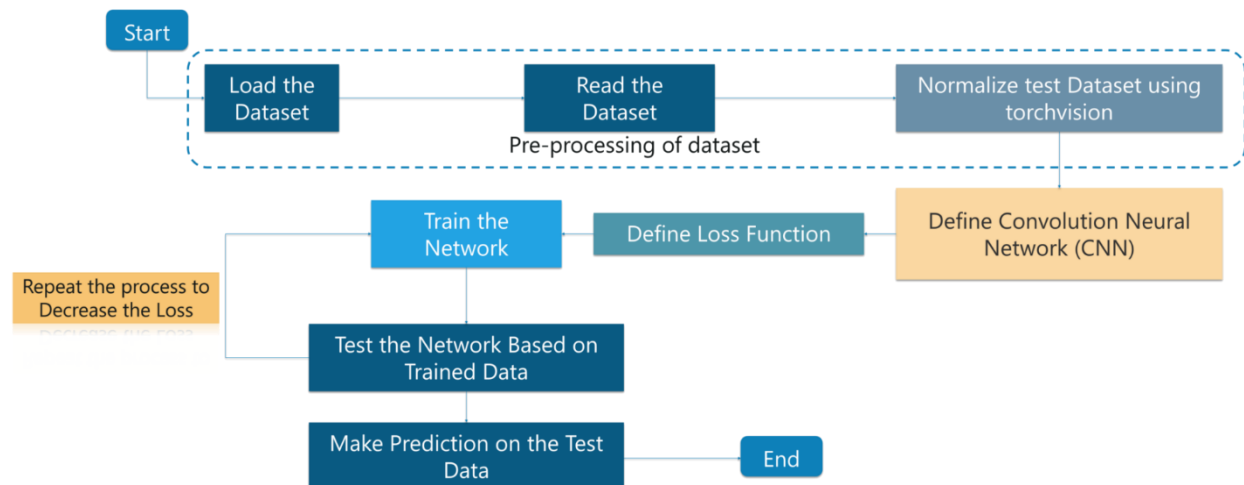
- ✓ A fully connected layer also known as the dense layer, in which the results of the convolutional layers are fed through one or more neural layers to generate a prediction.
- ✓ In case of a fully connected layer, all the elements of all the features of the previous layer get used in the calculation of each element of each output feature, this also includes an activation function based on our business requirement.
- ✓ In between the convolutional layer and the fully connected layer, there is a 'Flatten' layer. Flattening transforms a two-dimensional matrix of features into a vector that can be fed into a fully connected neural network classifier.

System Architecture:

Architecture



Data Flow Diagram:



5. Walking through the model:

5.1 Importing libraries:

▼ Importing Libraries

```
[ ] import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import tensorflow as tf
```

```
▶ from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Dropout,Activation,Flatten
from tensorflow.keras.layers import Conv2D,MaxPooling2D
```

▼ Assigning Values

```
[ ] num_classes=10
```

from tensorflow.keras.datasets import cifar10

The cifar10 dataset is contained in the keras library itself. By using the above statement we will import the dataset for building the model.

from tensorflow.keras.models import sequential

Model groups layers into an object with training and inference features. Sequential groups a linear stack of layers into a tf.keras.model. Sequential provides training and inference features on this model

from tensorflow.keras import layers

A Sequential model is appropriate for a **plain stack of layers** where each layer has exactly one input tensor and one output tensor. So from the above library we import layers for building the sequential model.

from tensorflow.keras.layers import Dense

Dense implements the operation: $\text{output} = \text{activation}(\text{dot}(\text{input}, \text{kernel}) + \text{bias})$ where activation is the element-wise activation function passed as the activation argument, kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer (only applicable if use_bias is True). These are all attributes of Dense.

from tensorflow.keras.layers import Dropout

The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. Inputs not set to 0 are scaled up by $1/(1 - \text{rate})$ such that the sum over all inputs is unchanged.

Note that the Dropout layer only applies when training is set to True such that no values are dropped during inference. When using model.fit, training will be appropriately set to True automatically, and in other contexts, you can set the kwarg explicitly to True when calling the layer.

(This is in contrast to setting trainable=False for a Dropout layer. trainable does not affect the layer's behavior, as Dropout does not have any variables/weights that can be frozen during training.)

from tensorflow.keras.layers import Activation

Applies an activation function to an output. Few of the activation functions available are relu, sigmoid, softmax, tanh.

from tensorflow.keras.layers import Flatten

Flattens the input. Does not affect the batch size.

from tensorflow.keras.layers import Conv2D

2D convolution layer (e.g. spatial convolution over images). This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. If use bias is True, a bias vector is created and added to the outputs. Finally, if activation is not None, it is applied to the outputs as well.

from tensorflow.keras.layers import MaxPooling2D

Max pooling operation for 2D spatial data. Downsamples the input along its spatial dimensions (height and width) by taking the maximum value over an input window (of size defined by pool_size) for each channel of the input. The window is shifted by strides along each dimension.

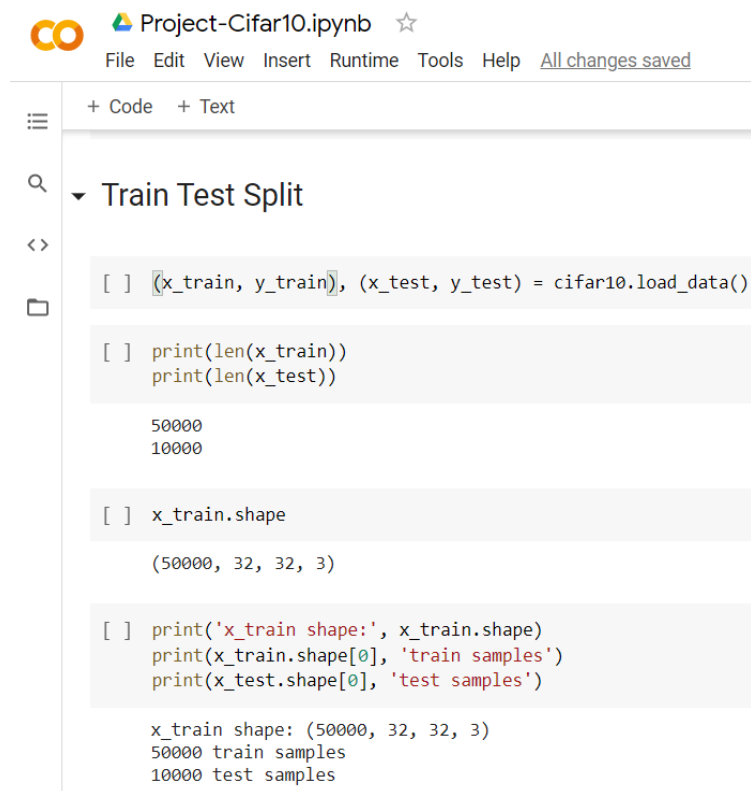
from tensorflow.keras.callbacks import ModelCheckpoint

ModelCheckpoint callback is used in conjunction with training using model.fit() to save a model or weights (in a checkpoint file) at some interval, so the model or weights can be loaded later to continue the training from the state saved.

A few options this callback provides include:

- Whether to only keep the model that has achieved the "best performance" so far, or whether to save the model at the end of every epoch regardless of performance.
- Definition of 'best'; which quantity to monitor and whether it should be maximized or minimized.
- The frequency it should save at. Currently, the callback supports saving at the end of every epoch, or after a fixed number of training batches.
- Whether only weights are saved, or the whole model is saved.

5.2 Load data and split



The screenshot shows a Jupyter Notebook titled "Project-Cifar10.ipynb". The interface includes a menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", "Help", and "All changes saved". Below the menu bar, there are tabs for "+ Code" and "+ Text". The notebook content is organized into a sidebar with a search icon, a "Train Test Split" section, and a code editor. The code editor contains the following Python code:

```
[ ] [[x_train, y_train], (x_test, y_test) = cifar10.load_data()

[ ] print(len(x_train))
    print(len(x_test))

    50000
    10000

[ ] x_train.shape

    (50000, 32, 32, 3)

[ ] print('x_train shape:', x_train.shape)
    print(x_train.shape[0], 'train samples')
    print(x_test.shape[0], 'test samples')

    x_train shape: (50000, 32, 32, 3)
    50000 train samples
    10000 test samples
```

In this step we will load the cifar10 data to our model and we will split the data into train and test sets.

As we have in total of 60000 images, the train set now contains the 50000 images and test set contains 10000 images respectively.

The dimensions of the images in the dataset is (32,32,3), as they are colored.

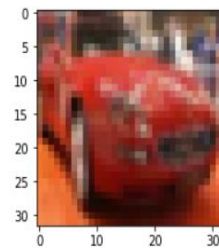
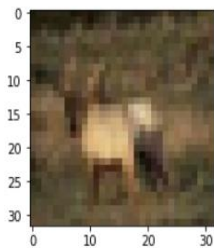
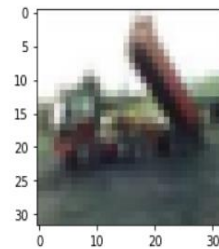
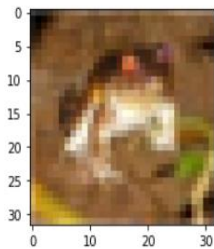
5.3. Visualizing the data

Here we look at the images present in our data in a random manner to get the idea of our dataset.

We can take any number of images that we intend to look at.

▼ Visualizing the dataset

```
[ ] n=6
plt.figure(figsize=(20,10))
for i in range(n):
    plt.subplot(330+1+i)
    plt.imshow(x_train[i])
plt.show()
```



5.4. One hot encoding

One hot encoding is **one method of converting data to prepare it for an algorithm and get a better prediction**. With one-hot, we convert each categorical value into a new categorical column and assign a binary value of 1 or 0 to those columns. Each integer value is represented as a binary vector.

▼ One hot encoding

```
▶ # Convert class vectors to binary class matrices.
y_train = tf.keras.utils.to_categorical(y_train, num_classes)
y_test = tf.keras.utils.to_categorical(y_test, num_classes)
```

```
[ ] x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
```

5.5. Initializing Sequential Model

▼ Initialize the model

```
[ ] # Initialize the model  
    model = Sequential()
```

5.6. Adding Layers

At this step we will add the convolutional layers, pooling layers, fully connected layers with all the hyperparameter being tuned.

The tuning of the hyperparameters plays a vital role in improving the results.



▼ Adding layers



```
[ ] # Create the model with two 32 convolution filters -> pooling layer -> two 64 conv filters -> pooling layer -> flattening -> fully conncted layer
    model.add(Conv2D(32, (3, 3), padding='same',
                     input_shape=x_train.shape[1:]))
    model.add(Activation('relu'))
    model.add(Conv2D(32, (3, 3)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.3))
```

```
[ ] model.add(Conv2D(64, (3, 3), padding='same'))
    model.add(Activation('relu'))
    model.add(Conv2D(64, (3, 3)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.3))
```

```
[ ] model.add(Flatten())
    model.add(Dense(512))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))
    model.add(Dense(num_classes))
    model.add(Activation('softmax'))
```



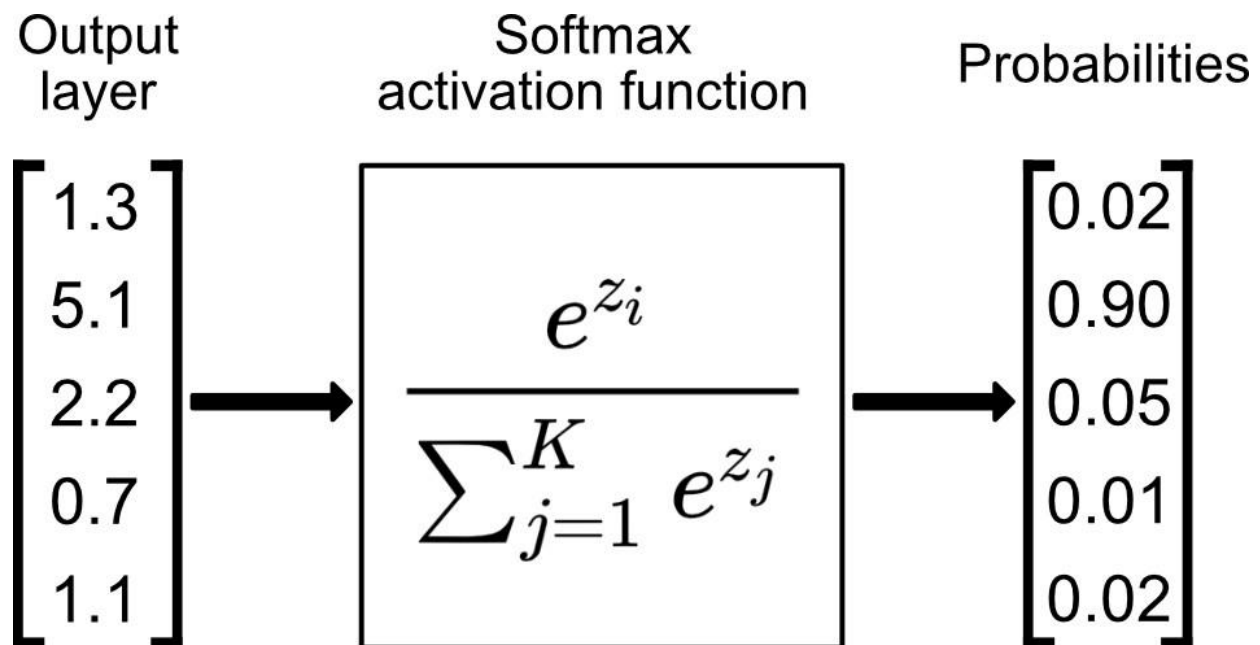
Softmax Activation Function:

Softmax is a mathematical function that converts a vector of numbers into a vector of probabilities, where the probabilities of each value are proportional to the relative scale of each value in the vector.

The most common use of the softmax function in applied machine learning is in its use as an activation function in a neural network model. Specifically, the network is configured to output N values, one for each class in the classification task, and the softmax function is used to normalize the outputs, converting them from weighted sum values into probabilities that sum to one. Each value in the output of the softmax function is interpreted as the probability of membership for each class.

The softmax function is used as the activation function in the output layer of neural network models that predict a multinomial probability distribution. That is,

softmax is used as the activation function for multi-class classification problems where class membership is required on more than two class labels.



5.7. Adding an Optimizer

Optimizers are algorithms or methods used to change the attributes of your neural network such as weights and learning rate in order to reduce the **losses**.

Optimization algorithms or strategies are responsible for reducing the losses and to provide the most accurate results possible.

▼ Adding an Optimizer

```
[ ] # initiate RMSprop optimizer
    opt = keras.optimizers.Adam(learning_rate=1.0e-4)
```

Adam Optimizer

The **Adam optimization algorithm** is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications in computer vision and natural language processing.

Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data.

Instead of adapting the parameter learning rates based on the average first moment (the mean) as in RMSProp, Adam also makes use of the average of the second moments of the gradients (the uncentered variance).

5.8. Compiling and model summary

The `compile()` method: **specifying a loss, metrics, and an optimizer**. To train a model with `fit()`, you need to specify a loss function, an optimizer, and optionally, some metrics to monitor.

```
[ ] # Let's train the model using Adam
    model.compile(loss='categorical_crossentropy',
                  optimizer=opt,
                  metrics=['accuracy'])
```

▼ Model summary

```
[ ] model.summary()
```

Model: "sequential_3"

Layer (type)	Output Shape	Param #
=====		
conv2d_12 (Conv2D)	(None, 32, 32, 32)	896
activation_18 (Activation)	(None, 32, 32, 32)	0
conv2d_13 (Conv2D)	(None, 30, 30, 32)	9248
activation_19 (Activation)	(None, 30, 30, 32)	0
max_pooling2d_6 (MaxPooling2D)	(None, 15, 15, 32)	0
dropout_9 (Dropout)	(None, 15, 15, 32)	0

5.9. Train the model

Model fitting is **a measure of how well a machine learning model generalizes to similar data to that on which it was trained**. A model that is well-fitted produces more accurate outcomes. ... Each machine learning algorithm has a basic set of parameters that can be changed to improve its accuracy.

+ Code + Text

Train the model

```
[ ] checkpoint = ModelCheckpoint('best_model_simple.h5', # model filename
                                monitor='val_loss', # quantity to monitor
                                verbose=0, # verbosity - 0 or 1
                                save_best_only= True, # The latest best model will not be overwritten
                                mode='auto') # The decision to overwrite model is made
                                                # automatically depending on the quantity to monitor
```

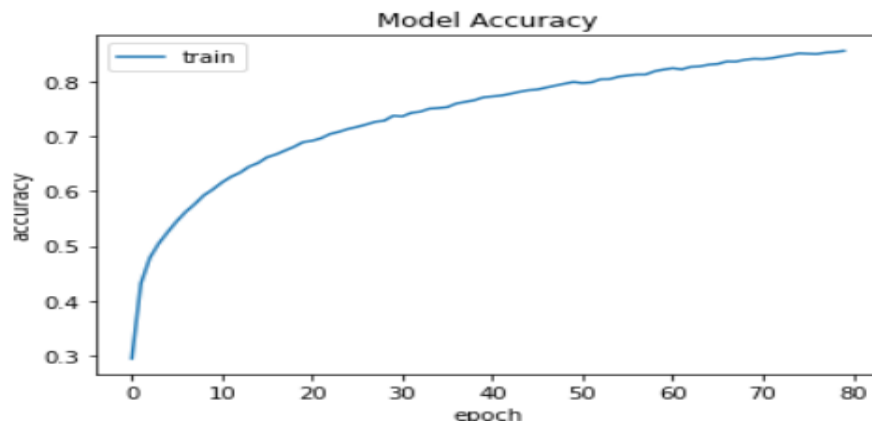
```
[ ] history=model.fit(x_train, y_train,
                      batch_size=128,
                      epochs=80,
                      validation_data=(x_test, y_test),
                      shuffle=True, verbose=1, callbacks=[checkpoint])
```

```
Epoch 51/80
391/391 [=====] - 9s 22ms/step - loss: 0.5737 - accuracy: 0.7970 - val_loss: 0.6474 - val_accuracy: 0.7732
Epoch 52/80
391/391 [=====] - 9s 22ms/step - loss: 0.5658 - accuracy: 0.7984 - val_loss: 0.6380 - val_accuracy: 0.7807
Epoch 53/80
391/391 [=====] - 9s 22ms/step - loss: 0.5555 - accuracy: 0.8040 - val_loss: 0.6375 - val_accuracy: 0.7775
Epoch 54/80
391/391 [=====] - 9s 22ms/step - loss: 0.5502 - accuracy: 0.8042 - val_loss: 0.6509 - val_accuracy: 0.7777
Epoch 55/80
391/391 [=====] - 9s 22ms/step - loss: 0.5446 - accuracy: 0.8087 - val_loss: 0.6414 - val_accuracy: 0.7811
Epoch 56/80
391/391 [=====] - 9s 22ms/step - loss: 0.5327 - accuracy: 0.8108 - val_loss: 0.6308 - val_accuracy: 0.7842
Epoch 57/80
391/391 [=====] - 9s 22ms/step - loss: 0.5294 - accuracy: 0.8126 - val_loss: 0.6326 - val_accuracy: 0.7843
Epoch 58/80
391/391 [=====] - 9s 22ms/step - loss: 0.5242 - accuracy: 0.8127 - val_loss: 0.6306 - val_accuracy: 0.7813
Epoch 59/80
391/391 [=====] - 9s 22ms/step - loss: 0.5188 - accuracy: 0.8187 - val_loss: 0.6249 - val_accuracy: 0.7847
```

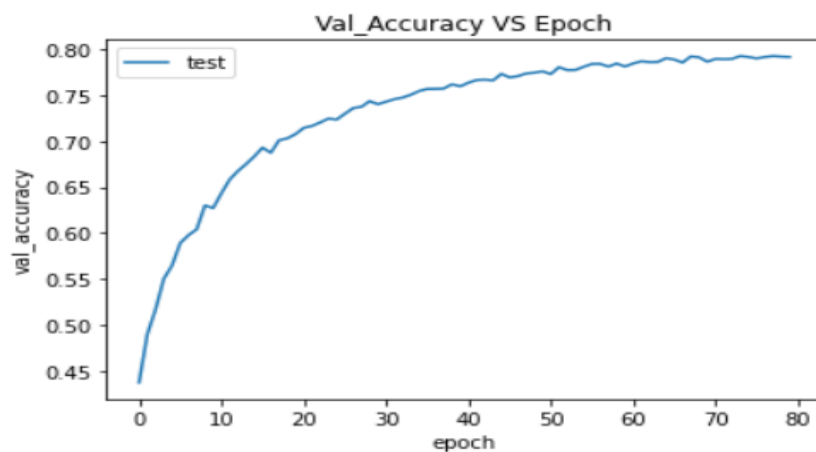
+ Code + Text

```
Epoch 66/80
391/391 [=====] - 9s 22ms/step - loss: 0.4760 - accuracy: 0.8319 - val_loss: 0.6148 - val_accuracy: 0.7890
Epoch 67/80
391/391 [=====] - 9s 22ms/step - loss: 0.4639 - accuracy: 0.8365 - val_loss: 0.6262 - val_accuracy: 0.7857
Epoch 68/80
391/391 [=====] - 9s 22ms/step - loss: 0.4621 - accuracy: 0.8362 - val_loss: 0.6168 - val_accuracy: 0.7924
Epoch 69/80
391/391 [=====] - 9s 22ms/step - loss: 0.4544 - accuracy: 0.8395 - val_loss: 0.6152 - val_accuracy: 0.7915
Epoch 70/80
391/391 [=====] - 9s 22ms/step - loss: 0.4488 - accuracy: 0.8413 - val_loss: 0.6193 - val_accuracy: 0.7866
Epoch 71/80
391/391 [=====] - 9s 22ms/step - loss: 0.4472 - accuracy: 0.8408 - val_loss: 0.6199 - val_accuracy: 0.7897
Epoch 72/80
391/391 [=====] - 9s 22ms/step - loss: 0.4434 - accuracy: 0.8424 - val_loss: 0.6176 - val_accuracy: 0.7894
Epoch 73/80
391/391 [=====] - 9s 22ms/step - loss: 0.4332 - accuracy: 0.8457 - val_loss: 0.6206 - val_accuracy: 0.7897
Epoch 74/80
391/391 [=====] - 9s 22ms/step - loss: 0.4299 - accuracy: 0.8479 - val_loss: 0.6102 - val_accuracy: 0.7928
Epoch 75/80
391/391 [=====] - 9s 22ms/step - loss: 0.4247 - accuracy: 0.8513 - val_loss: 0.6149 - val_accuracy: 0.7918
Epoch 76/80
391/391 [=====] - 9s 22ms/step - loss: 0.4197 - accuracy: 0.8505 - val_loss: 0.6126 - val_accuracy: 0.7902
Epoch 77/80
391/391 [=====] - 9s 22ms/step - loss: 0.4196 - accuracy: 0.8499 - val_loss: 0.6204 - val_accuracy: 0.7918
Epoch 78/80
391/391 [=====] - 9s 22ms/step - loss: 0.4120 - accuracy: 0.8527 - val_loss: 0.6152 - val_accuracy: 0.7928
Epoch 79/80
391/391 [=====] - 9s 22ms/step - loss: 0.4088 - accuracy: 0.8537 - val_loss: 0.6100 - val_accuracy: 0.7922
Epoch 80/80
391/391 [=====] - 9s 22ms/step - loss: 0.4055 - accuracy: 0.8558 - val_loss: 0.6139 - val_accuracy: 0.7918
```

```
[ ] import matplotlib.pyplot as plt
plt.plot(history.history['accuracy'])
plt.title("Model Accuracy")
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train'],loc='upper left')
plt.show()
```



```
[ ] plt.plot(history.history['val_accuracy'])
plt.title("Val_Accuracy VS Epoch")
plt.ylabel('val_accuracy')
plt.xlabel('epoch')
plt.legend(['test'],loc='upper left')
plt.show()
```



5.10.Evaluation metrics

▼ Evaluation metrics

```
[ ] _,acc=model.evaluate(x_test,y_test)
    print("Accuracy is:%.2f%%"%(acc*100))

313/313 [=====] - 2s 6ms/step - loss: 0.6139 - accuracy: 0.7918
Accuracy is:79.18%
```

5.11.Predictions

```
[ ] classes=['airplane','automobile','bird','cat','deer','dog','frog','horse','ship','truck']
    print(classes)
```

```
['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
```

```
▶ image=x_test[0]
  pred = model.predict(np.array([image]))
  pred1=pred.flatten()
  k=dict(zip(classes,pred1))
  print(k)
```

```
➞ {'airplane': 1.5422294e-05, 'automobile': 0.0014015617, 'bird': 0.0001542751, 'cat': 0.68388957, 'deer': 5.639033e-06,
```



```
[ ] all_values=k.values()
    max_value=max(all_values)
    d={v:k for (v,k) in k.items() if k==max_value}
    print(d)
```

```
{'cat': 0.68388957}
```

6. CONCLUSION

- The CNN is a type of Deep Neural Networks (DNN) that consists of many layers such as the Conv layers, Pooling layer, and the fully-connected layer.
- Convolutional neural networks (CNNs) are widely used in pattern- and image-recognition problems as they have a number of advantages compared to other techniques.
- CNN's has its own importance in the field of computer vision with wide many applications.
- The Cifar10 dataset imported from keras has the 10 classes and we build a model such that our model predicts the given image with a far decent accuracy.
- The image classification is just on of the vital applications of computer vision and it has helped in understanding the convolution operation and various layers involved in Convolutional neural networks.
- As the future is going to be more towards automation and digitalization these technologies would definitely help us in understanding what happens behind the scenes.

		Title	Guide Name
Roll No	Name		
18311A05L9 18311A05Q0 19315A0519	D.MADHAN N.SHASHANK T.KARTHIK	IMAGE CLASSIFICATION Using CNN	Ms.NEHA

ABSTRACT

The image classification is just one of the vital applications of computer vision and it has helped in understanding the convolution operation and various layers involved in Convolutional neural networks.

CIFAR-10 is a very popular computer vision dataset. This dataset is well studied in many types of deep learning research for object recognition. This dataset consists of 60,000 images divided into 10 target classes, with each category containing 6000 images of shape 32*32. This dataset contains images of low resolution (32*32), which allows researchers to try new algorithms. The 10 different classes of this dataset are:

1. Airplane
2. Car
3. Bird
4. Cat
5. Deer
6. Dog
7. Frog
8. Horse
9. Ship
10. Truck

CIFAR-10 dataset is already available in the datasets module of Keras. We do not need to download it; we can directly import it from keras.datasets

Internal Guide

Ms.Neha
Assistant Professor
Department of CSE

Project Coordinator


Mr. NV Subba Reddy
Associate Professor
Department of CSE

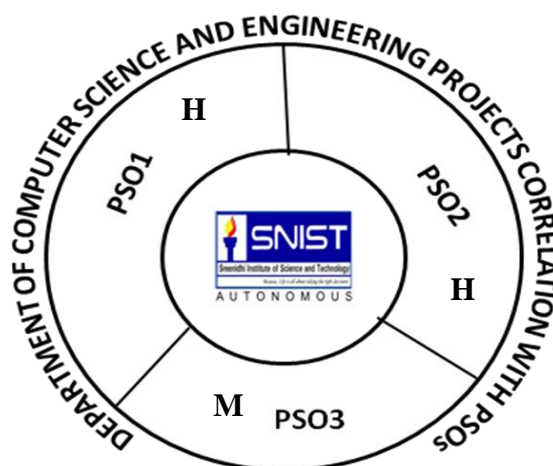
Head of Department

Dr.Aruna Varanasi
Professor & HOD
Department of CSE

Roll No	Name	Title	Guide Name
18311A05L9 18311A05Q0 19315A0519	D.MADHAN N.SHASHANK T.KARTHIK	IMAGE CLASSIFICATION Using CNN	Ms.Neha

BATCH NO:	ROLL NO:	PRODUCT/APP	ETHICS	RESEARCH	SOCIALSCIENCE	SAFETY
18	18311A05L9					
18	18311A05Q0					
18	19315A0519					

 SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING Projects Correlation with POs											
PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
H	H	M	M	H	L	M	H	H	M	M	M



H High
M Moderate
L Low

Guide

HOD