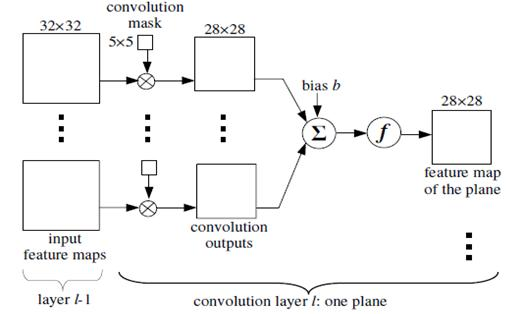
**4.1 SYSTEM ARCHITECTURE:**



**Figure 2.** CNN system architecture.

**4.2 DATA FLOW DIAGRAM:**

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

DFD DIAGRAM

**START**

Genarete CNN Train & Test Model

Upload Test image

Algorithms algorithu

Classify picture in image

**End process**

**4.3 UML DIAGRAMS:**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**GOALS:**

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Integrate best practices.

**USE CASE DIAGRAM:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can bedepicted.



**CLASS DIAGRAM:**

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



**SEQUENCE DIAGRAM:**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



Coding

#{'Forgery': 0, 'Real': 1}

from tkinter import \*

import tkinter

from tkinter import filedialog

import numpy as np

from tkinter.filedialog import askopenfilename

import pandas as pd

from keras.optimizers import Adam

from keras.models import model\_from\_json

from tkinter import simpledialog

from keras.models import Sequential

from keras.layers import Convolution2D

from keras.layers import MaxPooling2D

from keras.layers import Flatten

from keras.layers import Dense,Activation,BatchNormalization

import os

from keras.preprocessing import image

from keras.preprocessing.image import ImageDataGenerator

from tkinter import messagebox

import cv2

from imutils import paths

import imutils

import cv2

import numpy as np

main = tkinter.Tk()

main.title("Detecting Artificial Images through Local Binary Patterns") #designing main screen

main.geometry("700x500")

global filename

global loaded\_model

def get\_pixel(img, center, x, y):

new\_value = 0

try:

if img[x][y] >= center:

new\_value = 1

except:

pass

return new\_value

def lbp\_calculated\_pixel(img, x, y):

center = img[x][y]

val\_ar = []

val\_ar.append(get\_pixel(img, center, x-1, y+1)) # top\_right

val\_ar.append(get\_pixel(img, center, x, y+1)) # right

val\_ar.append(get\_pixel(img, center, x+1, y+1)) # bottom\_right

val\_ar.append(get\_pixel(img, center, x+1, y)) # bottom

val\_ar.append(get\_pixel(img, center, x+1, y-1)) # bottom\_left

val\_ar.append(get\_pixel(img, center, x, y-1)) # left

val\_ar.append(get\_pixel(img, center, x-1, y-1)) # top\_left

val\_ar.append(get\_pixel(img, center, x-1, y)) # top

power\_val = [1, 2, 4, 8, 16, 32, 64, 128]

val = 0

for i in range(len(val\_ar)):

val += val\_ar[i] \* power\_val[i]

return val

def upload(): #function to upload tweeter profile

global filename

filename = filedialog.askopenfilename(initialdir="testimages")

messagebox.showinfo("File Information", "image file loaded")