**Experiment 7**

**Aim -** To implement CNN Deep Learning Applications like

i) Image Classification System

ii) Handwritten Digit Recognition System (like MNIST Dataset)

iii) Traffic Signs Recognition

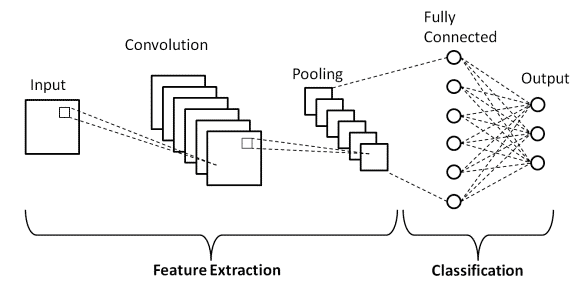
**Theory -**

**Convolutional Neural Networks (CNNs)**

**- Introduction to CNNs:**

Convolutional Neural Networks (CNNs) are a class of deep learning models specifically designed for processing and classifying visual data, such as images and videos. They are inspired by the visual perception process in the human brain and have become the state-of-the-art technique for various computer vision tasks.

**Key Components of a CNN:**



1. Convolutional Layers: These layers apply convolutional operations to the input image. Convolutional filters (also known as kernels) slide over the input image to extract features like edges, textures, and patterns.

2. Pooling Layers: Pooling layers (e.g., Max Pooling) reduce the spatial dimensions of the feature maps, helping to make the network invariant to small changes in the input.

3. Fully Connected Layers: These layers connect every neuron in one layer to every neuron in the next layer, allowing the network to learn complex combinations of features.

**- Working of a CNN:**

1. Input: A CNN starts with an input image, which is passed through a series of convolutional and pooling layers. These layers extract hierarchical features from the input.

2. Flattening: The final convolutional or pooling layer is flattened into a vector.

3. Fully Connected Layers: The flattened vector is passed through one or more fully connected layers to make predictions. These layers learn to classify the input based on the extracted features.

4. Output: The output layer provides the final classification result, often using softmax activation for multi-class classification tasks.

**- Building an Image Classification System for Dogs and Cats**

**- Application Overview:**

In this experiment, we are developing an image classification system to distinguish between images of dogs and cats. This system is designed for binary classification, where the goal is to predict whether a given image contains a dog or a cat.

**Code and Output -**

**- Model training code -**

import tensorflow as tf

from keras.models import Sequential

from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

from keras.preprocessing.image import ImageDataGenerator

import matplotlib.pyplot as plt

import os

# Mount Google Drive if necessary

from google.colab import drive

drive.mount('/content/drive')

# Load and Preprocess the Dataset

train\_datagen = ImageDataGenerator(

rescale=1./255,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True,

validation\_split=0.2

)

train\_generator = train\_datagen.flow\_from\_directory(

'/content/drive/MyDrive/EXP7/train\_set',

target\_size=(128, 128),

batch\_size=32,

class\_mode='binary',

subset='training'

)

# Build the CNN Model

model = Sequential()

model.add(Conv2D(32, (3, 3), activation='relu', input\_shape=(128, 128, 3)))

model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(64, (3, 3), activation='relu'))

model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(128, (3, 3), activation='relu'))

model.add(MaxPooling2D((2, 2)))

model.add(Flatten())

model.add(Dense(128, activation='relu'))

model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

# Train the Model

history = model.fit(

train\_generator,

epochs=20,

)

# Print the Training Accuracy

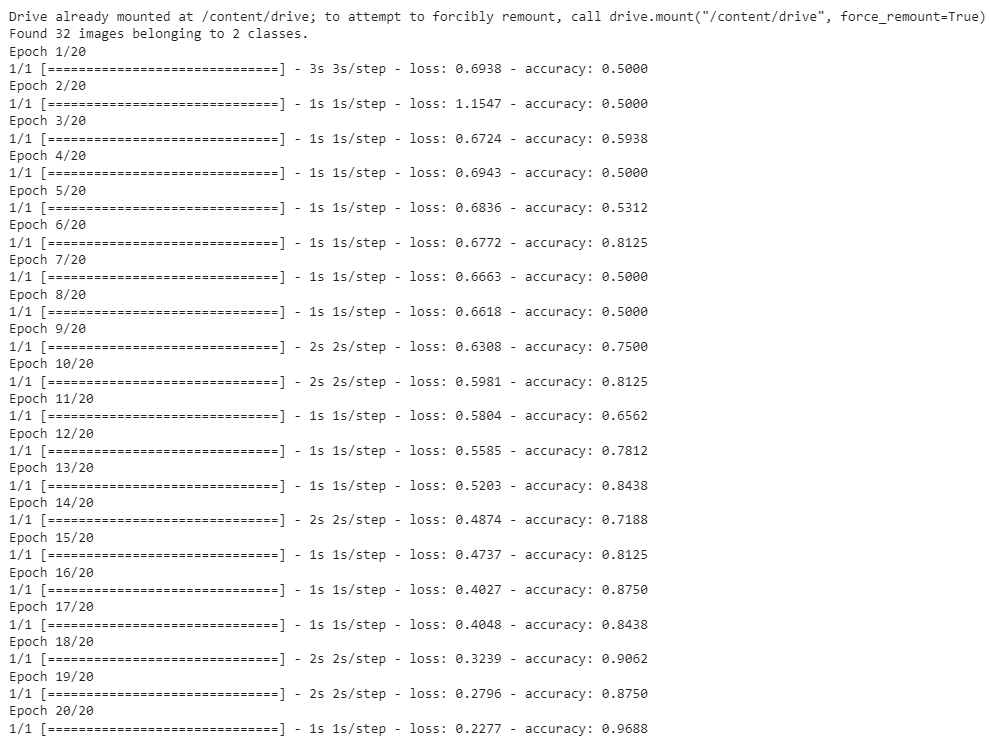
train\_accuracy = history.history['accuracy'][-1]

print(f'Training Accuracy: {train\_accuracy}')

# Save the trained model for future use

model.save('/content/drive/MyDrive/EXP7/trained\_model.h5')

**- Output of training -**



**Accuracy -**

****

**- Trying out on a user given image code -**

##Classiifying the image given by the user

import tensorflow as tf

from keras.preprocessing.image import load\_img, img\_to\_array

import numpy as np

# Function to classify the user-uploaded image

def classify\_image(image\_path, model):

img = load\_img(image\_path, target\_size=(128, 128))

img = img\_to\_array(img)

img = np.expand\_dims(img, axis=0)

img = img / 255.0 # Normalize the image data

prediction = model.predict(img)

if prediction[0] > 0.5:

return "It's a cat!"

else:

return "It's a dog!"

model = tf.keras.models.load\_model('/content/drive/MyDrive/EXP7/trained\_model.h5')

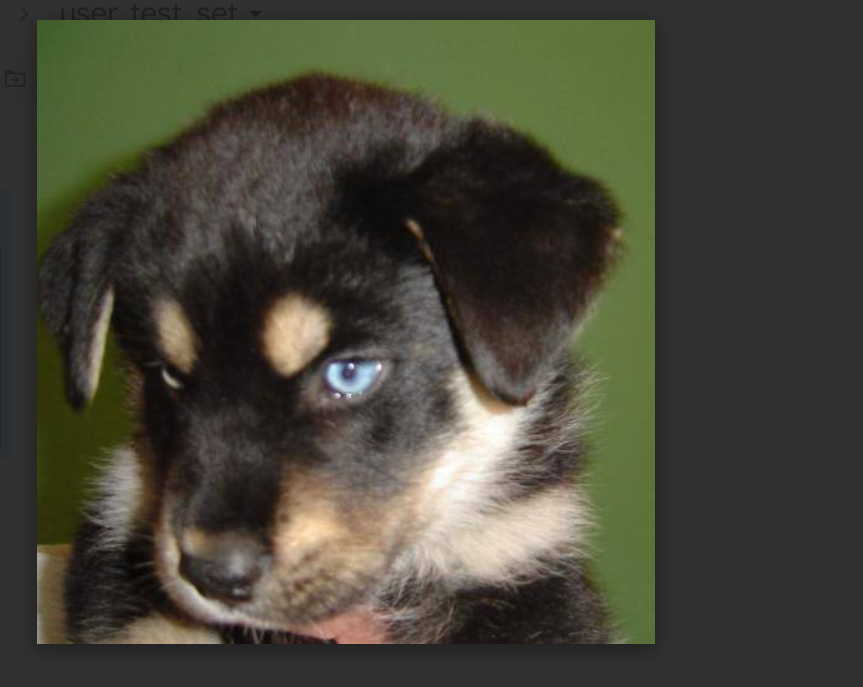
# Example usage:

image\_path = '/content/drive/MyDrive/EXP7/user\_test\_set/sample\_image1.jpg'

result = classify\_image(image\_path, model)

print(result)

**- User image -**

****

**- Output -**

****

**Conclusion -**

Thus we have successfully implemented CNN model by building an image classification system to differentiate dogs and cats, with an accuracy of **96%.**