|  |  |
| --- | --- |
| **Ex no : 8** | **Convolutional neural networks for image classification** |
| **Date :** |

**Aim**

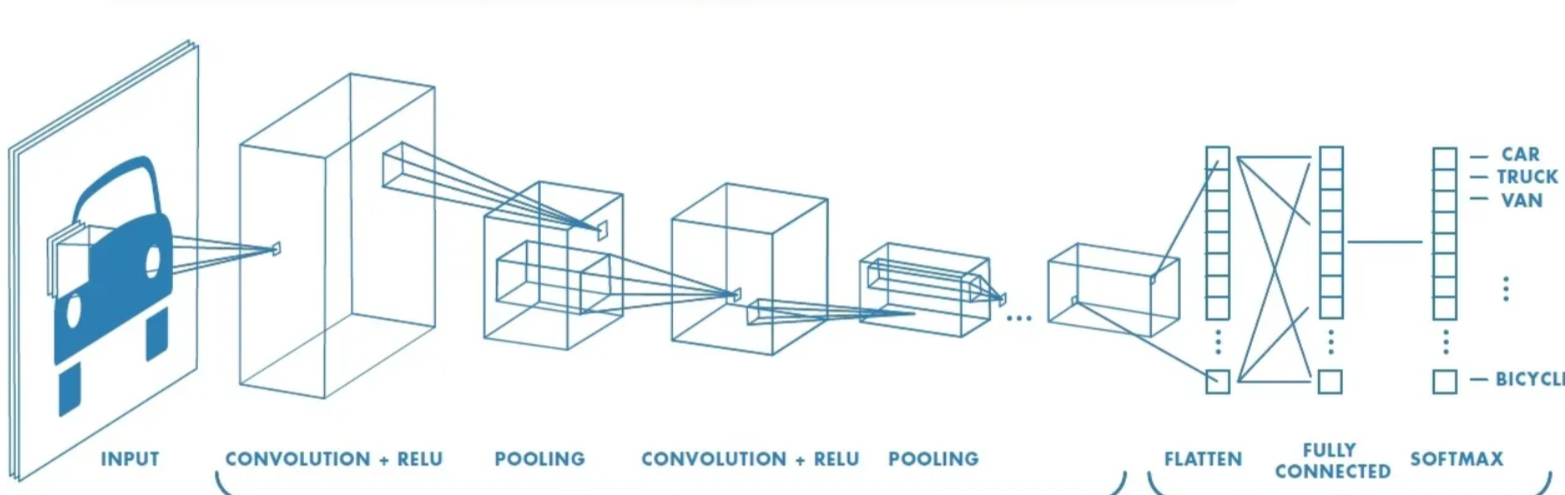
To implement convolutional neural networks for human emotion image classification.

**Basic Theory of CNN**

In neural networks, Convolutional neural network (CNNs) is one of the main categories to do images recognition, images classifications. Objects detections, recognition faces etc., are some of the areas where CNNs are widely used.

CNN image classifications takes an input image, process it and classify it under certain categories (Eg., Dog, Cat, Tiger, Lion). Computers sees an input image as array of pixels and it depends on the image resolution. Based on the image resolution, it will see h x w x d( h = Height, w = Width, d = Dimension ). Eg., An image of 6 x 6 x 3 array of matrix of RGB (3 refers to RGB values) and an image of 4 x 4 x 1 array of matrix of grayscale image.

Technically, deep learning CNN models to train and test, each input image will pass it through a series of convolution layers with filters (Kernals), Pooling, fully connected layers (FC) and apply Softmax function to classify an object with probabilistic values between 0 and 1. The below figure is a complete flow of CNN to process an input image and classifies the objects based on values.



**Code**

**Download the dataset from** [**https://github.com/nicknochnack/ImageClassification/tree/main/data**](https://github.com/nicknochnack/ImageClassification/tree/main/data)

import tensorflow as tf

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

#Load Data

data **=** tf**.**keras**.**utils**.**image\_dataset\_from\_directory('data')

data\_iterator **=** data**.**as\_numpy\_iterator()

batch **=** data\_iterator**.**next()

fig, ax **=** plt**.**subplots(ncols**=**4, figsize**=**(20,20))

**for** idx, img **in** enumerate(batch[0][:4]):

ax[idx]**.**imshow(img**.**astype(int))

ax[idx]**.**title**.**set\_text(batch[1][idx])

#Scale Data

data **=** data**.**map(**lambda** x,y: (x**/**255, y))

data**.**as\_numpy\_iterator()**.**next()

#Split Data

train\_size **=** int(len(data)**\***.7)

val\_size **=** int(len(data)**\***.2)

test\_size **=** int(len(data)**\***.1)

train\_size

train **=** data**.**take(train\_size)

val **=** data**.**skip(train\_size)**.**take(val\_size)

test **=** data**.**skip(train\_size**+**val\_size)**.**take(test\_size)

# Build Deep Learning Model

train

**from** tensorflow.keras.models **import** Sequential

**from** tensorflow.keras.layers **import** Conv2D, MaxPooling2D, Dense, Flatten, Dropout

model **=** Sequential()

model**.**add(Conv2D(16, (3,3), 1, activation**=**'relu', input\_shape**=**(256,256,3)))

model**.**add(MaxPooling2D())

model**.**add(Conv2D(32, (3,3), 1, activation**=**'relu'))

model**.**add(MaxPooling2D())

model**.**add(Conv2D(16, (3,3), 1, activation**=**'relu'))

model**.**add(MaxPooling2D())

model**.**add(Flatten())

model**.**add(Dense(256, activation**=**'relu'))

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

model**.**compile('adam', loss**=**tf**.**losses**.**BinaryCrossentropy(), metrics**=**['accuracy'])

model**.**summary()

#Train

logdir**=**'logs'

tensorboard\_callback **=** tf**.**keras**.**callbacks**.**TensorBoard(log\_dir**=**logdir)

hist **=** model**.**fit(train, epochs**=**20, validation\_data**=**val, callbacks**=**[tensorboard\_callback])

# Plot Performance

fig **=** plt**.**figure()

plt**.**plot(hist**.**history['loss'], color**=**'teal', label**=**'loss')

plt**.**plot(hist**.**history['val\_loss'], color**=**'orange', label**=**'val\_loss')

fig**.**suptitle('Loss', fontsize**=**20)

plt**.**legend(loc**=**"upper left")

plt**.**show()

fig **=** plt**.**figure()

plt**.**plot(hist**.**history['accuracy'], color**=**'teal', label**=**'accuracy')

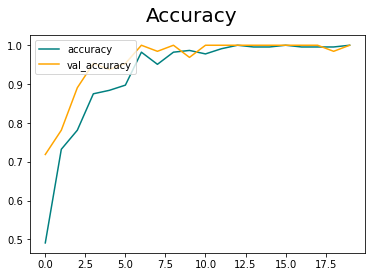
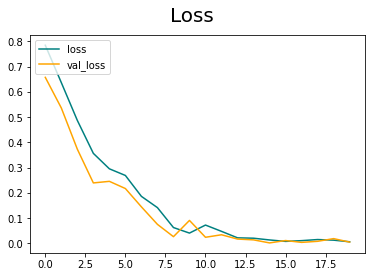
plt**.**plot(hist**.**history['val\_accuracy'], color**=**'orange', label**=**'val\_accuracy')

fig**.**suptitle('Accuracy', fontsize**=**20)

plt**.**legend(loc**=**"upper left")

plt**.**show()

**OUTPUT**



**RESULT**

Thus convolutional neural networks for human emotion image classification has been carried out successfully.