DEEP FAKE DETECTION PROJECT

Fake\_real\_image\_training:

from keras.models import Sequential

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

from tensorflow.keras.preprocessing.image import ImageDataGenerator # type: ignore

# Define the CNN architecture

model = Sequential([

    Conv2D(32, (3, 3), activation='relu', input\_shape=(150, 150, 3)),

    MaxPooling2D((2, 2)),

    Conv2D(64, (3, 3), activation='relu'),

    MaxPooling2D((2, 2)),

    Conv2D(128, (3, 3), activation='relu'),

    MaxPooling2D((2, 2)),

    Flatten(),

    Dense(128, activation='relu'),

    Dense(1, activation='sigmoid')

])

# Compile the model

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

# Data preprocessing

train\_datagen = ImageDataGenerator(rescale=1./255)

train\_directory = 'real\_and\_fake\_face\_detection'

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

    train\_directory,

    target\_size=(150, 150),

    batch\_size=32,

    class\_mode='binary'

)

# Train the model

model.fit(

    train\_generator,

    steps\_per\_epoch=int(train\_generator.samples/train\_generator.batch\_size),

    epochs=10

)

# Save the trained model

model.save('fake\_real\_classifier.h5')

Fake\_real\_image\_detection:

import numpy as np

from keras.preprocessing import image

from keras.models import load\_model

# Load the pre-trained model

model = load\_model('fake\_real\_classifier.h5')

def preprocess\_image(img\_path):

    img = image.load\_img(img\_path, target\_size=(150, 150))

    img\_array = image.img\_to\_array(img)

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

    return img\_array

def predict\_image\_real\_or\_fake(img\_path):

    img\_array = preprocess\_image(img\_path)

    result = model.predict(img\_array)

    if result[0][0] > 0.5:

        print("Real Image")

    else:

        print("Fake Image")

# Path to the image you want to classify

image\_path = 'C:\\mounika.T\\real\_and\_fake\_face\_detection'

# Predict whether the image is real or fake

predict\_image\_real\_or\_fake('C:\\mounika.T\\real\_and\_fake\_face\_detection\\real\_and\_fake\_face\\training\_fake\\easy\_1\_1110.jpg')

Output:





