**Model Training Code (CNN for Fake Image Detection)**

import tensorflow as tf

from tensorflow.keras.models import Sequential

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

from tensorflow.keras.preprocessing.image import ImageDataGenerator

import pandas as pd

# Load the dataset and labels (assuming dataset is pre-generated and saved in a specific structure)

df = pd.read\_excel('path\_to\_dataset/fake\_colorized\_image\_detection\_dataset.xlsx')

# Image DataGenerator for loading and augmenting data

datagen = ImageDataGenerator(rescale=1./255, validation\_split=0.2)

# Load real and fake images from the dataset directory (update paths as necessary)

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

'dataset\_directory',

target\_size=(224, 224),

batch\_size=32,

class\_mode='binary', # Real vs. Fake

subset='training'

)

validation\_generator = datagen.flow\_from\_directory(

'dataset\_directory',

target\_size=(224, 224),

batch\_size=32,

class\_mode='binary',

subset='validation'

)

# Define CNN model

model = Sequential([

Conv2D(32, (3, 3), activation='relu', input\_shape=(224, 224, 3)),

MaxPooling2D(2, 2),

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

MaxPooling2D(2, 2),

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

MaxPooling2D(2, 2),

Flatten(),

Dense(512, activation='relu'),

Dense(1, activation='sigmoid') # Binary classification (Real or Fake)

])

# Compile the model

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

# Train the model

history = model.fit(

train\_generator,

epochs=10,

validation\_data=validation\_generator

)

# Save the trained model

model.save('fake\_colorized\_image\_detection\_model.h5')

print("Model training completed and saved.")