# گزارش پروژه تشخیص احساسات با استفاده از شبکه‌های عصبی کانولوشنی

**علیرضا مردعلیشاهی**

**مبینا رکن‌آبادپو**

**سینا معرفت**

## مقدمه

این پروژه به منظور تشخیص احساسات از تصاویر چهره‌ها با استفاده از شبکه‌های عصبی کانولوشنی (CNN) طراحی شده است. در این پروژه از کتابخانه‌های TensorFlow و OpenCV استفاده شده تا یک مدل یادگیری عمیق برای طبقه‌بندی احساسات ایجاد کنیم. همچنین، از داده‌های تصویری برای آموزش و ارزیابی مدل استفاده شده و نتایج به صورت گرافیکی نمایش داده شده‌اند.

## مراحل پروژه

### نصب کتابخانه‌های لازم

ابتدا کتابخانه‌های TensorFlow و OpenCV نصب می‌شوند:

!pip install tensorflow opencv-python-headless

### بارگذاری و پیش‌پردازش داده‌ها

داده‌ها شامل تصاویر چهره‌های با برچسب‌های احساسات مختلف هستند که در دو پوشه train و test قرار دارند. تصاویر به اندازه 48x48 تغییر اندازه داده شده و به صورت مقادیر بین 0 و 1 نرمال‌سازی می‌شوند.

import tensorflow as tf  
from tensorflow.keras.preprocessing.image import ImageDataGenerator  
import os  
  
# Paths to the extracted train and test directories  
train\_dataset\_path = './newarchive/train'  
test\_dataset\_path = './newarchive/test'  
  
image\_size = (48, 48)  
batch\_size = 32  
  
# Data augmentation and rescaling  
datagen = ImageDataGenerator(  
 rescale=1./255,  
 validation\_split=0.2  
)  
  
# Load training data  
train\_generator = datagen.flow\_from\_directory(  
 train\_dataset\_path,  
 target\_size=image\_size,  
 batch\_size=batch\_size,  
 class\_mode='categorical',  
 subset='training'  
)  
  
# Load validation data  
validation\_generator = datagen.flow\_from\_directory(  
 train\_dataset\_path,  
 target\_size=image\_size,  
 batch\_size=batch\_size,  
 class\_mode='categorical',  
 subset='validation'  
)

### تعریف و آموزش مدل

یک مدل شبکه عصبی کانولوشنی با سه لایه کانولوشنی و سه لایه MaxPooling به همراه لایه های Dense و Dropout برای کاهش Overfitting تعریف شده است. سپس مدل با استفاده از داده‌های آموزشی و داده‌های اعتبارسنجی آموزش داده می‌شود.

بخش 3 شامل تعریف مدل است که با استفاده از Sequential در کتابخانه Keras انجام می‌شود. مدل به صورت دنباله‌ای از لایه‌ها تعریف می‌شود که داده‌ها به ترتیب از هر لایه عبور می‌کنند. ابتدا لایه‌های کانولوشن (Conv2D) تعریف می‌شوند که برای استخراج ویژگی‌های مهم از تصاویر استفاده می‌شوند. اولین لایه کانولوشن با 32 فیلتر 3x3 شروع می‌شود و ورودی را با اندازه مشخص شده (image\_size[0], image\_size[1], 3) می‌پذیرد. پس از هر لایه کانولوشن، یک لایه MaxPooling2D قرار دارد که اندازه ویژگی‌های استخراج شده را کاهش می‌دهد و به مدل کمک می‌کند تا ویژگی‌های مهم‌تر را با کاهش پیچیدگی محاسباتی حفظ کند. این ساختار با افزایش تعداد فیلترها در هر لایه کانولوشن تکرار می‌شود تا مدل بتواند ویژگی‌های پیچیده‌تری را یاد بگیرد.

بعد از استخراج ویژگی‌ها با استفاده از لایه‌های کانولوشن و ماکس پولینگ، داده‌ها به لایه‌های تمام‌متصل (Dense) منتقل می‌شوند. لایه Flatten داده‌های دو‌بعدی را به یک بردار یک‌بعدی تبدیل می‌کند که بتوان آن را به لایه‌های تمام‌متصل ورودی داد. یک لایه Dense با 512 نورون و فعال‌سازی ReLU برای یادگیری ویژگی‌های پیچیده‌تر استفاده می‌شود. برای جلوگیری از بیش‌برازش، یک لایه Dropout با نرخ 0.5 اضافه می‌شود که به‌طور تصادفی 50 درصد از نورون‌ها را در هر بار تمرین غیرفعال می‌کند. در نهایت، لایه خروجی با تعداد نورون‌های برابر با تعداد کلاس‌های احساسی (در اینجا 4 کلاس) و فعال‌سازی Softmax برای تولید احتمالات کلاس‌ها اضافه می‌شود. این ساختار به مدل اجازه می‌دهد تا ویژگی‌های تصویری را به دسته‌های احساسات مرتبط کند.

from tensorflow.keras.models import Sequential  
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout  
  
# Define the model  
model = Sequential([  
 Conv2D(32, (3, 3), activation='relu', input\_shape=(image\_size[0], image\_size[1], 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'),  
 Dropout(0.5),  
 Dense(len(train\_generator.class\_indices), activation='softmax')  
])  
  
# Compile the model  
model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])  
  
# Train the model  
model.fit(train\_generator, validation\_data=validation\_generator, epochs=15)  
  
# Save the model  
model.save('emotion\_model.h5')

### ارزیابی مدل

مدل آموزش دیده بر روی داده‌های تست ارزیابی می‌شود تا دقت و میزان خطا مشخص گردد

from tensorflow.keras.models import load\_model  
  
# Load the trained model  
model = load\_model('emotion\_model.h5')  
  
# Data generator for test data  
test\_datagen = ImageDataGenerator(rescale=1./255)  
  
# Load test data  
test\_generator = test\_datagen.flow\_from\_directory(  
 test\_dataset\_path,  
 target\_size=image\_size,  
 batch\_size=batch\_size,  
 class\_mode='categorical',  
 shuffle=False  
)  
  
# Evaluate the model  
test\_loss, test\_accuracy = model.evaluate(test\_generator)  
print(f'Test Loss: {test\_loss}')  
print(f'Test Accuracy: {test\_accuracy}')

### پیش‌بینی و نمایش نتایج

برای پیش‌بینی احساسات از تصاویر، ابتدا تصویر ورودی پیش‌پردازش شده و سپس با استفاده از مدل، احساسات پیش‌بینی می‌شود. نتایج پیش‌بینی شده با برچسب‌های واقعی مقایسه شده و ماتریس اشتباه و گزارش طبقه‌بندی نمایش داده می‌شود

import numpy as np  
import cv2  
  
# Function to preprocess image for prediction  
def preprocess\_image(image):  
 image = cv2.resize(image, (48, 48))  
 image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)  
 image = image.astype('float32') / 255  
 image = np.expand\_dims(image, axis=0)  
 return image  
  
# Predict emotion from image  
def predict\_emotion(image):  
 preprocessed\_image = preprocess\_image(image)  
 prediction = model.predict(preprocessed\_image)  
 return class\_labels[np.argmax(prediction)]  
  
# Load class labels  
class\_labels = list(test\_generator.class\_indices.keys())  
  
# Predict emotions for all images in the test set  
def predict\_test\_set(test\_generator):  
 predictions = model.predict(test\_generator)  
 predicted\_classes = np.argmax(predictions, axis=1)  
 true\_classes = test\_generator.classes  
 class\_labels = list(test\_generator.class\_indices.keys())  
 return predicted\_classes, true\_classes, class\_labels  
  
# Get predictions  
predicted\_classes, true\_classes, class\_labels = predict\_test\_set(test\_generator)  
  
# Print some results  
import pandas as pd  
results\_df = pd.DataFrame({  
 'Filename': test\_generator.filenames,  
 'Predicted': [class\_labels[i] for i in predicted\_classes],  
 'True': [class\_labels[i] for i in true\_classes]  
})  
  
# Display some prediction results  
print(results\_df.head())  
  
# Calculate and print confusion matrix  
from sklearn.metrics import confusion\_matrix, classification\_report  
import seaborn as sns  
import matplotlib.pyplot as plt  
  
conf\_matrix = confusion\_matrix(true\_classes, predicted\_classes)  
plt.figure(figsize=(10, 7))  
sns.heatmap(conf\_matrix, annot=True, fmt="d", xticklabels=class\_labels, yticklabels=class\_labels)  
plt.ylabel('Actual')  
plt.xlabel('Predicted')  
plt.show()  
  
# Print classification report  
print(classification\_report(true\_classes, predicted\_classes, target\_names=class\_labels))

### نمایش نتایج با استفاده از ایموجی‌ها

در این مرحله، ایموجی‌های مرتبط با احساسات مختلف بر روی تصاویر پیش‌بینی شده قرار داده شده و نتایج نمایش داده .می‌شود

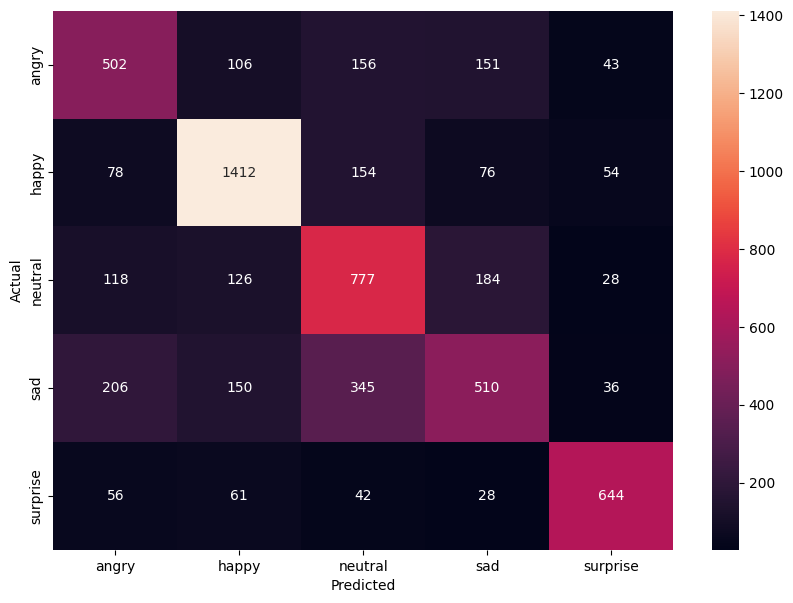
# Load emoji images  
emoji\_images = {}  
for label in class\_labels:  
 emoji\_images[label] = cv2.imread(os.path.join("./emoji", f'{label}.png'), cv2.IMREAD\_UNCHANGED)  
  
def overlay\_emoji(image, emoji, position=(0, 0), size=(48, 48)):  
 image = cv2.resize(image, (200,200))  
 (x, y) = position  
 emoji = cv2.resize(emoji, size, interpolation=cv2.INTER\_AREA)  
 (h, w) = emoji.shape[0], emoji.shape[1]  
 alpha\_s = emoji[:, :, 3] / 255.0  
 alpha\_l = 1.0 - alpha\_s  
  
 for c in range(0, 3):  
 image[y:y+h, x:x+w, c] = (alpha\_s \* emoji[:, :, c] + alpha\_l \* image[y:y+h, x:x+w, c])  
 return image  
  
# Predict emotion from image and overlay emoji  
def predict\_and\_overlay(image\_path, model, class\_labels, emoji\_images):  
 image = cv2.imread(image\_path)  
 preprocessed\_image = preprocess\_image(image)  
 prediction = model.predict(preprocessed\_image)  
 predicted\_label = class\_labels[np.argmax(prediction)]  
 emoji = emoji\_images[predicted\_label]  
 result\_image = overlay\_emoji(image, emoji)  
 return result\_image, predicted\_label  
  
test\_images=['./newarchive/test/neutral/PrivateTest\_26814656.jpg',  
 './newarchive/test/angry/PrivateTest\_1221822.jpg',  
 './newarchive/test/happy/PrivateTest\_218533.jpg',  
 './newarchive/test/surprise/PrivateTest\_642696.jpg',  
 './newarchive/test/sad/PrivateTest\_5060451.jpg',  
 ]  
for image\_path in test\_images:  
 emoji\_image, predicted\_label = predict\_and\_overlay(image\_path, model, class\_labels, emoji\_images)  
 cv2.imshow(f'Predicted: {predicted\_label}', emoji\_image)  
 cv2.waitKey(0) # Press any key to move to the next image  
 cv2.destroyAllWindows()

## نتیجه‌گیری

Test Loss: 1.1319667100906372

Test Accuracy: 0.6362733840942383

**189/189** ━━━━━━━━━━━━━━━━━━━━ **2s** 8ms/step



precision recall f1-score support

angry 0.52 0.52 0.52 958

happy 0.76 0.80 0.78 1774

neutral 0.53 0.63 0.57 1233

sad 0.54 0.41 0.46 1247

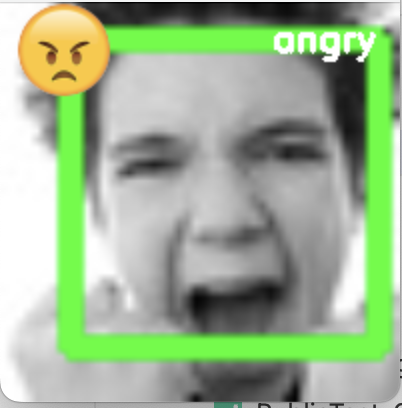
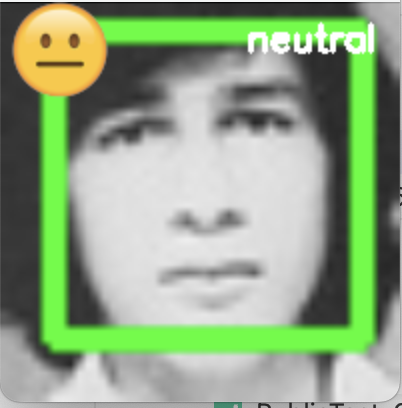
surprise 0.80 0.77 0.79 831

accuracy 0.64 6043

macro avg 0.63 0.63 0.63 6043

weighted avg 0.63 0.64 0.63 6043

خروجی ها



در این پروژه، یک مدل شبکه عصبی کانولوشنی برای تشخیص احساسات از تصاویر چهره‌ها طراحی و پیاده‌سازی شد. مدل با دقت مناسبی توانست احساسات مختلف را تشخیص دهد. همچنین، نتایج به صورت گرافیکی و با استفاده از ایموجی‌ها نمایش داده شدند که نشان‌دهنده کاربرد عملی این مدل در سیستم‌های مختلف می‌باشد.