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 48 x48 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 Dropout و Dense 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 3 3x 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

```python
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

```python
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()
```

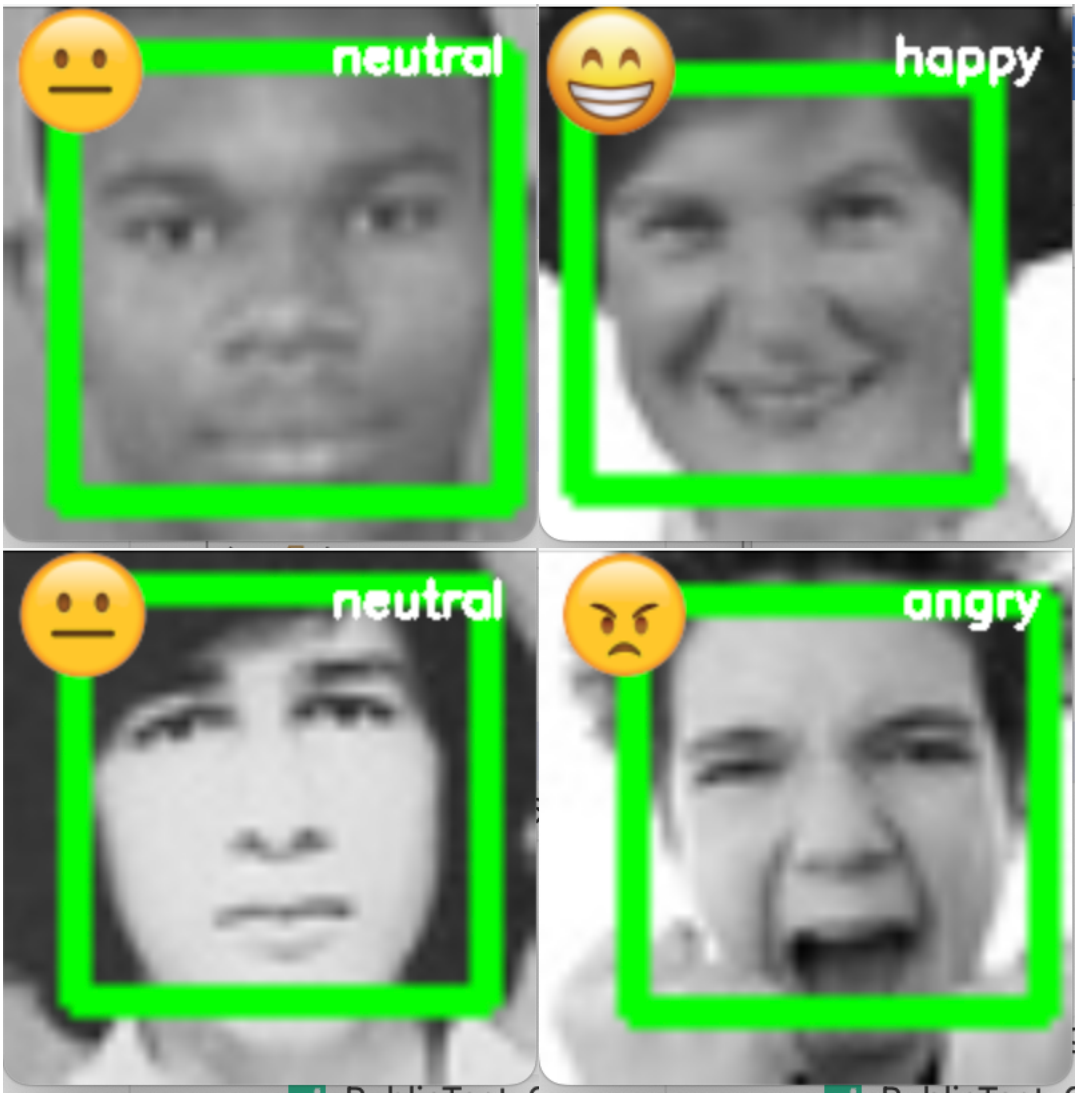
نتیجه‌گیری

```
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.