دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

تشخیص لبخند

درس: یادگیری عمیق

استاد: دکتر محمد کیانی

شیدا عابدپور

۴۰۰۳۶۲۳۰۲۵

آبان ۱۴۰۳

مقدمه

در این پروژه قصد بر آن است تا بتوان به کمک روش Transfer Learning، لبخند زدن انسان را تشخیص داد. در ابتدا لازم است چهره‌های دیتاست GENKI4K تشخیص و برش داده شوند تا شبکه بهتر بتواند عمل کند.

۱- تشخیص چهره

مدل RetinaFace به‌عنوان یکی از پیشرفته‌ترین و دقیق‌ترین مدل‌ها برای شناسایی چهره‌ها شناخته می‌شود. در این پروژه، از این مدل برای تشخیص چهره‌ها و همچنین انجام alignment تصاویر استفاده شد. بر اساس نتایج به‌دست‌آمده، این مدل در تصاویری که چهره‌ها به‌وضوح قابل شناسایی بودند و نیازی به برش نداشتند، قادر به شناسایی چهره‌ها نبود. از مجموع ۴۰۰۰ تصویر، مدل موفق به شناسایی چهره‌ها در ۳۸۴۶ تصویر شد، در حالی که در باقی تصاویر، چهره‌ها به‌طور خودکار و به‌گونه‌ای طبیعی کراپ شده بودند. نمونه‌ای از تشخیص چهره توسط این مدل در تصویر ۱-۱ و نمونه‌ای از عدم تشخیص چهره در تصویر ۱-۲ قابل مشاهده است.

این فرایند در محیط کولب و با استفاده از GPU مدل T4 انجام شد که مدت زمان لازم برای تکمیل آن حدود ۸ دقیقه بود و نتایج خوبی به دنبال داشت.

تصویر ۱-۱

تصویر ۱-۲

۲- تشخیص لبخند

مدل‌های DeepFace در زمینه شناسایی و تحلیل چهره‌ها کاربرد دارند و بر روی مجموعه‌های داده مختلفی آموزش داده شده‌اند. این مدل‌ها، مانند مدل‌های FaceNet، VGGFace، و سایر مدل‌های مشابه، برای انجام وظایف مختلفی مانند شناسایی چهره، تایید چهره، و پیش‌بینی ویژگی‌های چهره (مثل سن، جنسیت و احساسات) استفاده می‌شوند. با توجه به نزدیکی پیش‌بینی احساسات توسط این مدل‌ها به پروژه فعلی، یعنی تشخیص لبخند، از این مدل‌ها به عنوان مدل پایه در روش یادگیری انتقالی استفاده شد.

یادگیری انتقالی می‌تواند به دو روش انجام شود:

۱. **تنظیم دقیق مدل پیش‌آموزش دیده (Fine-tuning)**: در این روش، از یک مدل پیش‌آموزش دیده استفاده می‌شود که روی یک مجموعه داده بزرگ آموزش دیده است و سپس این مدل برای انجام یک کار جدید تنظیم می‌شود. این کار با آزادسازی (unfreeze) برخی از لایه‌های مدل و آموزش مجدد آن‌ها بر روی داده‌های جدید انجام می‌شود.
۲. **استخراج ویژگی‌ها (Feature Extraction)**: در این روش، از مدل پیش‌آموزش دیده برای استخراج ویژگی‌های مفید از داده‌های جدید استفاده می‌شود. مدل پیش‌آموزش دیده به عنوان یک استخراج‌کننده ویژگی عمل می‌کند و این ویژگی‌ها به یک مدل دیگر داده می‌شوند تا پیش‌بینی‌ها را انجام دهد.

یکی از چالش‌های استفاده از مدل‌های deepface، عدم امکان پیاده‌سازی به روش اول در تنسورفلو است، بنابراین از مدل پایه به عنوان یک استخراج‌کننده ویژگی استفاده شد و سپس این ویژگی‌ها به یک شبکه عصبی کوچک جهت پیش‌بینی نهایی داده شد.

```python
# Function to generate embeddings
def generate_embeddings(image_paths, labels):
    embeddings = []

    # Process each image and extract embeddings
    for path in image_paths:
        # Read and preprocess image
        img = cv2.imread(path)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)  # Convert BGR to RGB
        img = cv2.resize(img, (160, 160))  # Resize to FaceNet input size
        img = img.astype("float32") / 255.0  # Normalize image

        # Extract embeddings using DeepFace
        result = DeepFace.represent(img_path=path, model_name="VGG-Face", enforce_detection=False)
        if result:
            embeddings.append(np.array(result[0]["embedding"]))
        else:
            print(f"Face not detected in {path}")

    return np.array(embeddings)
```

باید توجه داشت استفاده از یادگیری انتقالی احتمال overfit را افزایش می‌دهد. جهت پیشگیری، باید نکات زیر را در نظر داشت:

۱- **افزایش داده‌ها(Data Augmentation)**: در زمینه شناسایی چهره، شامل تغییرات مختلفی در تصاویر چهره است که به مدل کمک می‌کند تا ویژگی‌های مختلف چهره را بهتر یاد بگیرد. می‌توان این روش را بر روی داده‌های آموزشی(train) اعمال کرد تا احتمال بیش‌برداز کمتر شود.

```python
def augment_images(image_paths, labels, augment_count=3):
    # Create a generator for data augmentation
    data_gen = ImageDataGenerator(
        rotation_range=20,        # Rotate images by a random degree
        width_shift_range=0.2,    # Shift images horizontally
        height_shift_range=0.2,   # Shift images vertically
        shear_range=0.2,          # Shear images
        zoom_range=0.2,           # Zoom in/out images
        horizontal_flip=True      # Flip images horizontally
    )

    aug_labels = []
    for idx, (path, label) in enumerate(zip(image_paths, labels)):
        img = cv2.imread(path)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)  # Convert BGR to RGB
        img = cv2.resize(img, (160, 160)) / 255.0  # Resize and normalize
        for i in range(augment_count):
            aug_img = data_gen.random_transform(img)  # Apply augmentation
            aug_labels.append(label)  # Save the corresponding label

    return np.array(aug_labels)
```

۲- **Drop out**: می‌توان با استفاده از آن در لایه‌های Dense مدل طبقه‌بندی، احتمال بیش‌برداز را کاهش داد.

```python
# Build a neural network classifier
classifier = tf.keras.Sequential([
    tf.keras.layers.Input(shape=(512,)),  # Input layer for Facenet512 embeddings
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Dense(1, activation='sigmoid')  # Binary output for smile/non-smile
])
```

۳- **تنظیم ضریب یادگیری(Learning Rate)**: با توجه به نتایج، باید ضریب یادگیری را کوچک انتخاب کرد. در این پروژه مقدار 10^{-5} مناسب بود.

```python
learning_rate = 1e-5
optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate)

# Compile the model with the custom learning rate
classifier.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['accuracy'])
```

گزارشات نهایی

با توجه به دقت مدل retina-face، از آن به عنوان مدل تشخیص چهره و align کردن تصاویر استفاده شد. این فرایند در محیط colab صورت گرفت و ۸ دقیقه به طول انجامید.(در فایل face_detection.ipnyb قابل مشاهده است.)

مدل‌های موجود در Deepface به عنوان مدل پایه در یادگیری انتقالی انتخاب شدند، زیرا دیتاست آن‌ها بر روی چهره‌ها می‌باشد و همچنین پیش‌بینی احساسات به تشخیص لبخند نزدیک است، بنابراین می‌توانند ویژگی‌های مرتبط و بهتری استخراج کنند. فرایند استخراج ویژگی‌ها در محیط colab بین ۳۰ تا ۴۰ دقیقه به طول انجامید. تعداد لایه‌ها، میزان دراپ اوت و ضریب یادگیری در مدل طبقه‌بندی نهایی با توجه به نتایج تنظیم شد تا شبکه کمتر دچار overfit شود.

از مدل‌های deepface، مدل VGG-Face و مدل Facenet512 بررسی شدند، نتایج به شرح زیر است:

در مدل Facenet512 :

دقت داده‌های train بین ۸۲ تا ۸۵ درصد گزارش شد.

دقت داده‌های test بین ۷۶ درصد تا ۸۰ درصد گزارش شد.

در مدل VGG-Face:

دقت داده‌های train حدود ۹۸ درصد گزارش شد.

دقت داده‌های test بین ۹۰ تا ۹۲ درصد گزارش شد.

مدل نهایی

با توجه به نتایج، از مدل VGG-Face جهت استخراج ویژگی از تصاویر استفاده شد، سپس ویژگی‌ها به شبکه عصبی زیر داده شد. دقت داده‌های آموزشی نزدیک ۹۸ درصد، داده‌های ارزیابی بین ۹۰ تا ۹۱ درصد گزارش شد.

```python
# Build a neural network classifier
classifier = tf.keras.Sequential([
    tf.keras.layers.Input(shape=(4096,)),  # Input layer for VGG-Face embeddings
    tf.keras.layers.Dense(1024, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(1, activation='sigmoid')  # Binary output for smile/non-smile
])

learning_rate = 1e-5 * 5
optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate)

# Compile the model with the custom learning rate
classifier.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
history = classifier.fit(X_train_embeddings, y_train,
                         epochs=10,
                         batch_size=32,
                         validation_data=(X_val_embeddings, y_val))
```

```
Epoch 1/10
90/90 [==============================] - 10s 101ms/step - loss: 0.6801 - accuracy: 0.5684 - val_loss: 0.6425 - val_accuracy: 0.6844
Epoch 2/10
90/90 [==============================] - 8s 85ms/step - loss: 0.5425 - accuracy: 0.7976 - val_loss: 0.3746 - val_accuracy: 0.8844
Epoch 3/10
90/90 [==============================] - 9s 102ms/step - loss: 0.3090 - accuracy: 0.8851 - val_loss: 0.2432 - val_accuracy: 0.8969
Epoch 4/10
90/90 [==============================] - 8s 88ms/step - loss: 0.2213 - accuracy: 0.9174 - val_loss: 0.2066 - val_accuracy: 0.9375
Epoch 5/10
90/90 [==============================] - 9s 100ms/step - loss: 0.1811 - accuracy: 0.9326 - val_loss: 0.1962 - val_accuracy: 0.9156
Epoch 6/10
90/90 [==============================] - 8s 91ms/step - loss: 0.1432 - accuracy: 0.9538 - val_loss: 0.1948 - val_accuracy: 0.9250
Epoch 7/10
90/90 [==============================] - 9s 101ms/step - loss: 0.1155 - accuracy: 0.9622 - val_loss: 0.1934 - val_accuracy: 0.9250
Epoch 8/10
90/90 [==============================] - 7s 76ms/step - loss: 0.0936 - accuracy: 0.9726 - val_loss: 0.2010 - val_accuracy: 0.9187
Epoch 9/10
90/90 [==============================] - 9s 101ms/step - loss: 0.0720 - accuracy: 0.9833 - val_loss: 0.2098 - val_accuracy: 0.9312
Epoch 10/10
90/90 [==============================] - 8s 93ms/step - loss: 0.0539 - accuracy: 0.9861 - val_loss: 0.2133 - val_accuracy: 0.9219
25/25 [==============================] - 0s 15ms/step - loss: 0.2736 - accuracy: 0.9013
Test Accuracy: 90.13%
```