

Face Recognition Using OpenCV

Face Recognition Using Deep Learning and Machine Vision in Python by OpenCV Library.



عنوان گزارش کار: تشخیص چهره با

استفاده از OpenCV

رشته تحصیلی: مهندسی تکنولوژی نرم افزار

کامپیوتر

استاد راهنما: دکتر مهدی رنجبر حسنی

دانشجو: دانیال کمالی

Work Report Title: Face Recognition
Using OpenCV

Field of Study: Computer Software
Technology Engineering

Supervisor: Dr. Mahdi Ranjbar Hassani

Student: Danial Kamali

Table of Contents:**فهرست مطالب:**

برپایی محیط مجازی و نصب وابستگی های لازم	3
کشف چهره و تشخیص چهره با استفاده از OpenCV - مرحله آموزش دادن سیستم	5
ایجاد دیتابیس برای تشخیص چهره	5
ضبط چهره ها	9
آموزش تشخیص دهنده	13
تشخیص چهره با استفاده از SQLite - واکنشی داده از OpenCV	17
پیاده سازی و نتایج	21
نتایج (تصاویر)	22
کارهای مرتبط	25
منابع	35

Setup Virtual Environment and Install Necessary Dependencies

برپایی محیط مجازی و نصب وابستگی‌های لازم

در این بخش ابتدا توضیح خواهیم داد که چرا کتابخانه **OpenCV** برای تشخیص چهره انتخاب شده است.

دلیل انتخاب کتابخانه **Open Computer Vision Library** (یا همان **OpenCV**) رایگان و متن‌باز بودن آن است که این کتابخانه را به یکی از محبوب‌ترین کتابخانه‌های پردازش تصویر و یادگیری ماشین در بین دانشجویان، محققان و توسعه دهندگان تبدیل کرده است. **OpenCV** کتابخانه‌ای چندسکویی (**Cross-platform**) است و توسط سیستم عامل‌های ویندوز، لینوکس، **iOS**، **MacOS**، **Android** پشتیبانی می‌شود. همچنین دارای رابط برنامه نویسی به زیان‌های **C**، **Java**، **Python**، **C++** و متلب می‌باشد.

در این پروژه از **Python 3.7.4** و **opencv-python 4.1.2** استفاده شده است. تشخیص چهره و کشف چهره (**Face Detection**) که در بیشتر موقع در کنار یکدیگر استفاده می‌شوند دارای دو مفهوم متفاوت هستند. کشف چهره یعنی سیستم قادر است چهره را در عکس یا فیلم شناسایی کند و تشخیص چهره در واقع تعیین می‌کند که این چهره متعلق به کیست.

نحوه برپایی **virtualenv** (محیط مجازی) و نصب وابستگی‌های لازم:

در اینجا مقدمات اولیه سیستم تشخیص چهره در پایتون را نصب خواهیم کرد. دلیل نصب محیط مجازی تعدد کتابخانه‌ها مورد استفاده بوده و اینکه در مواقعی این اجزا جزیی از کتابخانه استاندار نیستند و باید به صورت جداگانه اقدام به نصب آن‌ها کرد.

برای شروع کار اقدام به نصب **virtualenv** و سایر وابستگی‌ها در پایتون می‌کنیم. راحت‌ترین شیوه نصب استفاده از **pip** در پایتون است. برای نصب **virtualenv** توسط **pip** از کد

استفاده می‌کنیم و در صورت موفقیت آمیز بودن عملیات در ترمینال پیغام `virtualenv opencvenv` `Installing setuptools, pip, wheel...done.` نمایش می‌بادد.

حال آمده‌ی نصب وابستگی‌های لازم هستیم که عبارتند از:

OpenCV	.1
OpenCV-contrib	.2
SQLite	.3
Numpy	.4
Pillow	.5

برای نصب وابستگی‌ها از دستورات زیر استفاده می‌شود:

```
Pip3 install opencv-python
Pip3 install opencv-contrib-python
Pip3 install Pillow
```

لازم به ذکر است که SQLite به صورت پیش‌فرض در Python 3 در دسترس است بنابراین نیاز به نصب ندارد، همچنین numpy در هنگام نصب opencv-python بطور خودکار نصب می‌شود.

حال که تمام وابستگی‌های لازم نصب شده‌اند، ما آمده‌ی ساخت سیستم تشخیص چهره با استفاده از OpenCV هستیم.

برای نصب وابستگی‌های لازم روش ساده‌تری نیز وجود دارد که در صورت استفاده از Visual Studio و نصب بودن Python در آن می‌توان از مسیر

Tools > Python > Python Environment

اقدام به نصب افزونه‌ها کرد. که در این پروژه از این روش استفاده شد.

Face Detection and Face Recognition Using OpenCV – Training

کشف چهره و تشخیص چهره با استفاده از **OpenCV** – آموزش دادن سیستم

در بخش قبلی نصب پیش‌نیازها توضیح داده شد، در این بخش از گزارش کار تمرکز بر روی چگونگی نوشتتن کدهای لازم برای پیاده‌سازی ضبط و آموزش برنامه‌ی تشخیص چهره است.

مراحل را می‌توان به صورت زیر دسته‌بندی کرد:

1. ایجاد دیتابیس برای تشخیص چهره
2. ضبط چهره‌ها
3. آموزش تشخیص دهنده

ایجاد دیتابیس برای تشخیص چهره:

در ابتدا باید یک دیتابیس به منظور ذخیره نام مطابق هر چهره ایجاد شود. از **SQLite 3** برای این منظور استفاده می‌شود.

در پوشه پروژه فایلی به نام **create_database.py** ایجاد کرده و کدنویسی دیتابیس انجام می‌شود.

سپس اسکریپ (فایل آغازگر) پایتون را به کمک دستور

```
python3 create_database.py
```

اجرا می‌کنیم، این عمل موجب ایجاد یک فایل دیتابیس به نام **database.db** در مسیر فعلی (پوشه پروژه) می‌شود.

دیتابیس حاوی یک جدول به نام **users** با دو ستون به نام **id** و **name** است.

create_database.py

```
1. import sqlite3
2. conn = sqlite3.connect('database.db')
3. c = conn.cursor()
4. sql = """
5. DROP TABLE IF EXISTS users;
6. CREATE TABLE users (
7.         id integer unique primary key autoincrement,
8.         name text
9. );
10. """
11. c.executescript(sql)
12. conn.commit()
13. conn.close()
```

create_database.py

1. فراخوانی کتابخانه `sqlite3`
2. ایجاد ارتباط بین دیتابیس `sqlite3` به نام `database.db` با پایتون
3. ایجاد شیء از `Cursor` به منظور دادن دسترسی اجرا به پایتون در یک جلسه دیتابیس
4. اسکریپت اجرا دیتابیس در پایتون
5. دستور مربوط به `SQLite` بوده و شرط وجود جدول را بررسی می‌کند.
6. ساخت جدول جدید به نام `users`
7. ساخت فیلد `id` جدول به عنوان کلید و با مقدار افزایشی خودکار از نوع عدد صحیح
8. ساخت فیلد `name` از نوع متن
9. بستن دستورات ایجاد جدول
10. پایان اسکریپت `SQLite`
11. اجرا چندین عبارت `SQL` توسط دستور
12. پایان تراکنش با سیستم مدیریت پایگاه داده
13. بستن ارتباط با دیتابیس در پایتون

به محض ساخت دیتابیس می‌توان از نرم‌افزاری مانند

DB Browser for SQLite

برای دیدن ساختار دیتابیس استفاده کرد، همانند شکل زیر:

	id	name
1	1	Danial
2	2	Danial
3	3	Danial
4	4	Danial
5	5	Danial
6	6	Danial
7	7	Alireza
8	8	Alireza
9	9	Alireza
10	10	Alireza
11	11	Tom Hanks
12	12	Tom Hanks
13	13	Tom Hanks
14	14	Tom Hanks
15	15	Tom Cruise
16	16	Tom Cruise
17	17	Tom Cruise
18	18	Tom Cruise
19	19	Dwayne Johns...
20	20	Dwayne Johns...

ضبط چهره‌ها:

حال باید مجموعه داده‌ها را برای تشخیص چهره آماده کرد. ما از فایل

Haarcascade_frontalface_default.xml

که در گیت‌هاب و در مسیر

Opencv/data/haarcascades

وجود دارد استفاده می‌کنیم. لینک فایل در گیت‌هاب:

<https://github.com/opencv/opencv/tree/master/data/haarcascades>

فایل را پس از دانلود در مسیر (پوشه) پروژه قرار داده و بعد از آن فایلی به نام `record_face.py` را در مسیر پروژه ساخته و کدنویسی می‌کنیم.

کد مربوط به فایل وقتی اجرا شود ابتدا درخواست وارد کردن نام برای چهره را می‌کند، سپس از `haarcascade` به منظور پیدا کردن چهره‌ی موجود در تصویر دوربین استفاده می‌کند و به دنبال 3 نمونه هر یک با فاصله‌ی 100 میلی‌ثانیه می‌گردد.

هنگامی که 3 نمونه‌ی چهره یافت شد، داده‌ی نمونه‌ها را درون پوشه‌ای به نام `dataset` که درون مسیر اصلی پروژه ایجاد شده ذخیره می‌کند.

در قدم بعدی قصد آموزش تشخیص دهنده به منظور شناسایی چهره را داریم.

record_face.py

```
1. import cv2
2. import numpy as np
3. import sqlite3
4. import os
5. conn = sqlite3.connect('database.db')
6. if not os.path.exists('./dataset'):
7.     os.makedirs('./dataset')
8. c = conn.cursor()
9. face_cascade =
    cv2.CascadeClassifier('haarcascade_frontalface_default
    .xml')
10. cap = cv2.VideoCapture(0)
11. uname = input("Enter your name: ")
12. c.execute('INSERT INTO users (name) VALUES (?)',
(uname,))
13. uid = c.lastrowid
14. sampleNum = 0
15. while True:
16.     ret, img = cap.read()
17.     gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
18.     faces = face_cascade.detectMultiScale(gray, 1.3, 5)
19.     for (x,y,w,h) in faces:
20.         sampleNum = sampleNum+1
21.
        cv2.imwrite("dataset/User."+str(uid)+". "+str(sampleNum
        )+".jpg",gray[y:y+h,x:x+w])
22.         cv2.rectangle(img, (x,y), (x+w, y+h), (255,0,0),
2)
23.         cv2.waitKey(100)
24.         cv2.imshow('img',img)
25.         cv2.waitKey(1);
26.         if sampleNum > 2:
27.             break
28. cap.release()
29. conn.commit()
30. conn.close()
31. cv2.destroyAllWindows()
```

record_face.py

1. فراخوانی کتابخانه `opencv`
2. فراخوانی کتابخانه `numpy` با نام `np` (کتابخانه محاسبات علمی، شامل آرایه `n`-بعدی)
3. فراخوانی کتابخانه `sqlite3`
4. فراخوانی کتابخانه `os` (توابع تعامل با سیستم عامل)
5. اتصال به دیتابیس `SQLite` به نام `database.db`
6. بررسی شرط وجود نداشتن مسیر (پوشه) `dataset`
7. ساخت مسیر `dataset` اگر وجود نداشت
8. ایجاد شیء از `Cursor` به منظور دادن دسترسی اجرا به پایتون در یک جلسه دیتابیس
9. استفاده از فایل `haarcascade_frontalface_default.xml` و ساخت شیء `videoCapture`
10. ساخت شیء از `videoCapture`
11. نمایش متن و گرفتن مقدار وارد شده
12. فراخوانی مقدار از پایتون در دیتابیس
13. برگرداندن مقداری برای فیلد خود افزایشی (`id` (AUTO_INCREMENT))
14. مقداردهی اولیه عدد نمونه
15. اجرا حلقه تا زمان صحیح بودن
16. ضبط فریم به فریم (`ret` مقدار برگشتی از فریم دوربین را دریافت می‌کند)
17. عملیات بر روی فریم (تبدیل تصویر به مقیاس خاکستری)
18. عملیات تشخیص
19. تشکیل مستطیل اطراف شیء تشخیص داده شده
20. افزایش یک واحد مقدار `sampleNum`
21. استفاده از `id` برای ذخیره نمونه ضبط شده
22. تشکیل مستطیل و تعیین رنگ
23. ایجاد تاخیر 100 میلی ثانیه برای نمایش پنجره
24. نمایش تصویر در پنجره
25. ایجاد تاخیر 1 میلی ثانیه
26. شرط: اگر مقدار `sampleNum` بیش از 2 شد

record_face.py

27. خروج از حلقه
28. رهاسازی (خاتمه) ضبط
29. پایان تراکنش با سیستم مدیریت پایگاه داده
30. بستن ارتباط با دیتابیس در پایتون
31. از بین بردن (بستن) پنجره‌ها

آموزش تشخیص دهنده (Recognizer)

سه متد را برای شناسایی چهره ارائه می‌کند:

- Eigenfaces
- Fisherfaces
- Local Binary Patterns Histograms (LBPH)

هر سه متد، تشخیص چهره را به وسیله‌ی مقایسه چهره با چهره‌های موجود آموزش داده شده انجام می‌دهند. در مجموعه آموزش (Training Set)، الگوریتم چهره‌ها را تدارک دیده و به آن می‌گوییم که به کدام چهره تعلق دارد.

Eigenfaces و Fisherfaces یک توصیف ریاضی را از مهم‌ترین ویژگی‌های مجموعه‌ی آموزش را به‌طور کلی می‌یابند.

LBPH هر چهره در مجموعه‌ی آموزش را به‌طور جداگانه و مستقل تجزیه و تحلیل می‌کند. روش LBPH تا حدودی ساده‌تر است، به این معنا که هر تصویر را در مجموعه داده‌های محلی مشخص می‌کنیم و هنگامی که یک تصویر ناشناخته جدید ارائه می‌شود، ما همان تجزیه و تحلیل (analysis) را روی آن انجام می‌دهیم و نتیجه را با هر یک از تصاویر موجود در مجموعه داده مقایسه می‌کنیم.

در این پروژه از تشخیص چهره LBPH برای مقاصد خود بهره می‌بریم. برای انجام این کار یک فایل به نام `trainer.py` در پوشه پروژه ایجاد کرده و کدنویسی می‌کنیم. فایل را با دستور `python trainer.py` اجرا کرده، این کار موجب ایجاد فایلی به نام `recognizer` درون پوشه‌ی `trainingData.yml` می‌شود.

trainer.py

```
1. import os
2. import cv2
3. import numpy as np
4. from PIL import Image
5. recognizer = cv2.face.LBPHFaceRecognizer_create()
6. path = 'dataset'
7. if not os.path.exists('./recognizer'):
8.     os.makedirs('./recognizer')
9. def getImagesWithID(path):
10.    imagePaths = [os.path.join(path,f) for f in
11.        os.listdir(path)]
12.    faces = []
13.    IDs = []
14.    for imagePath in imagePaths:
15.        faceImg = Image.open(imagePath).convert('L')
16.        faceNp = np.array(faceImg,'uint8')
17.        ID = int(os.path.split(imagePath)[-1].
18.            split('.')[1])
19.        faces.append(faceNp)
20.        IDs.append(ID)
21.        cv2.imshow("training",faceNp)
22.        cv2.waitKey(10)
23.    return np.array(IDs), faces
24. IDs, faces = getImagesWithID(path)
25. recognizer.train(faces,IDs)
26. recognizer.save('recognizer/trainingData.yml')
27. cv2.destroyAllWindows()
```

trainer.py

1. فراخوانی کتابخانه `os`
2. فراخوانی کتابخانه `opencv`
3. فراخوانی کتابخانه `numpy` با نام `np`
4. فراخوانی کتابخانه تصویر (`Python Image Library`)
5. تعیین شیء از صفات تشخیص چهره `LBPH` به نام `recognizer`
6. تعامل با فایل در مسیر `dataset`
7. شرط: اگر مسیر `recognizer` وجود نداشت
8. ساخت مسیر (پوشش) `recognizer`
9. گرفتن تصویر با `ID`
10. پیوستن به یک یا چند مولفه مسیر به طور هوشمند، مقداردهی لیست `faces`
11. ایجاد لیست (آرایه) `IDs`
12. ایجاد لیست `faceImg`
13. برای لیست مسیر تصویر
14. باز کردن تصویر و تبدیل، نتیجه در متغیر `faceImg`
15. ایجاد آرایه (`unsigned integer (0 to 255)`)
16. تقسیم نام یک مسیر، نتیجه در متغیر `ID`
17. افزودن مقداری `faceNp` به انتهای `faces`
18. افزودن مقدار `ID` به انتهای `IDs`
19. متند نمایش تصویر (پنجره به طور خودکار اندازه تصویر می‌شود)
20. تاخیر `10` میلی ثانیه
21. بازگرداندن مقدار آرایه `IDs` و `faces`
22. گرفتن تصویر با `ID` از مسیر (`Path` (متغیر `Path`))
23. آموزش تشخیص دهنده
24. ذخیره نتیجه در مسیر `recognizer` و فایل `trainingData.yml`
25. از بین بردن (بستن) پنجره‌ها

در بخش دوم بطور خلاصه ما سه عمل اصلی به شرح زیر را انجام دادیم:

1. ایجاد `create_database.py` به منظور ایجاد دیتابیس و جدول
2. ایجاد `record_face.py` به منظور دریافت تصاویر چهره‌ها و ضبط نام مربوطه آن در دیتابیس.
3. ایجاد `trainer.py`, استفاده از تشخیص دهنده چهره‌ی OpenCV LBPH برای آموزش مجموعه داده که فایل `trainingData.yml` را خروجی می‌دهد که از آن برای تشخیص چهره استفاده خواهیم کرد.

برنامه تشخیص چهره تقریباً کامل شده است و آنچه که باید انجام دهیم تشخیص آن چهره‌ها و واکشی داده از SQLite است.

Face Recognition OpenCV – Fetching Data From SQLite

تشخیص چهره با استفاده از **OpenCV** – واکشی داده از **SQLite**

در این بخش تمرکز بر روی چگونگی نوشتن کدهای لازم برای تشخیص چهره و واکشی اطلاعات مربوط به کاربر از دیتابیس **SQLite** است.

برای آشنایی بیشتر با دلایل انتخاب و نحوه کار **LBPH** به منابع این گزارش رجوع کنید.

تشخیص چهره و واکشی داده از **SQLite**

حال ما از فایلی که در طول آموزش برای تشخیص (**Training to Recognize**) تهیه کرده‌ایم استفاده خواهیم کرد تا چهره کسی که روپروری دوربین قرار دارد را تشخیص دهیم. در حال حاضر محیط مجازی فعال شده و وابستگی‌های لازم نصب شده است.

حال به سراغ ایجاد یک فایل به نام **detector.py** می‌رویم و شروع به کدنویسی می‌کنیم.
سپس فایل را با استفاده از دستور **python3 detector.py** اجرا کرده و در نتیجه سیستم شروع به انجام فرآیند تشخیص چهره می‌کند.

detector.py

```
1. import cv2
2. import numpy as np
3. import sqlite3
4. import os
5. conn = sqlite3.connect('database.db')
6. c = conn.cursor()
7. fname = "recognizer/trainingData.yml"
8. if not os.path.isfile(fname):
9.     print("Please train the data first")
10.    exit(0)
11. face_cascade =
12.     cv2.CascadeClassifier('haarcascade_frontalface_default
13.     .xml')
14. cap = cv2.VideoCapture(0)
15. recognizer = cv2.face.LBPHFaceRecognizer_create()
16. recognizer.read(fname)
17. while True:
18.     ret, img = cap.read()
19.     gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
20.     faces = face_cascade.detectMultiScale(gray, 1.3, 5)
21.     for (x,y,w,h) in faces:
22.         cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 3)
23.         ids,conf = recognizer.predict(gray[y:y+h,x:x+w])
24.         c.execute("select name from users where id =
25.         (?) ;", (ids,))
26.         result = c.fetchall()
27.         name = result[0][0]
28.         if conf < 50:
29.             cv2.putText(img, name, (x+2,y+h-5),
30.             cv2.FONT_HERSHEY_SIMPLEX, 1, (150,255,0),2)
31.         else:
32.             cv2.putText(img, 'No Match', (x+2,y+h-5),
33.             cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255),2)
34.         cv2.imshow('Face Recognizer',img)
35.         k = cv2.waitKey(30) & 0xff
36.         if k == 27:
37.             break
38. cap.release()
39. cv2.destroyAllWindows()
```

detector.py

1. فراخوانی کتابخانه `opencv`
2. فراخوانی کتابخانه `numpy` با نام `np`
3. فراخوانی کتابخانه `sqlite3`
4. فراخوانی کتابخانه `os`
5. اتصال به دیتابیس `SQLite` به نام `database.db`
6. ایجاد شیء از `Cursor` به منظور دادن دسترسی اجرا به پایتون در یک جلسه دیتابیس
7. ذخیره مسیر فایل ایجاد شده از آموزش در `fname`
8. شرط: اگر فایل `fname` وجود نداشت
9. نمایش پیغام
10. خروج (عدد 0 برای نشان دادن موفقیت)
11. استفاده از فایل `haarcascade_frontalface_default.xml` و ساخت شیء
12. ساخت شیء از `videoCapture`
13. تعیین شیء از صفات تشخیص چهره `LBPH` به نام `recognizer`
14. خواندن `fname` توسط شیء `recognizer`
15. اجرا حلقه تا زمان صحیح بودن
16. ضبط فریم به فریم (`ret` مقدار برگشتی از فریم دوربین را دریافت می‌کند)
17. عملیات بر روی فریم (تبديل تصویر به مقیاس خاکستری)
18. عملیات تشخیص
19. تشکیل مستطیل اطراف شیء تشخیص داده شده
20. تشکیل مستطیل و تعیین رنگ
21. با توجه به یک مدل آموزش دیده، مجموعه جدید از داده‌ها را پیشینی می‌کند.
22. اجرا تعامل با دیتابیس و انتخاب نام براساس `ID`
23. متد `fetchall` تمام یا باقی‌مانده ردیف نتایج را گرفته و لیستی را برمی‌گرداند.
24. قرار دادن مقدار لیست دو بعدی `result` در `name`
25. شرط: اگر مقدار `conf` کوچک‌تر از 50 بود

detector.py

26. قرار دادن نام در موقعیت معین شده
27. در غیر این صورت:
28. قرار دادن عبارت **No Match** در موقعیت معین شده
29. نمایش تصویر
30. تاخیر 30 میلی ثانیه، مقداردهی k
31. شرط: اگر k برابر با 27 بود
32. خروج از حلقه
33. رهاسازی (خاتمه) ضبط
34. از بین بردن (بستن) پنجره ها

Implementation and Results:

پیاده سازی و نتایج:

در مراحل قبل به ساخت پایگاه داده و آموزش سیستم با استفاده از چهره های ضبط شده پرداختیم، با اجرا فایل `detector.py` یکسری فرآیندها برای تشخیص چهره صورت می پذیرد که در ادامه توضیح داده می شوند.

در ابتدا سیستم تشخیص چهره وجود فایل دیتابیس را بررسی می کند، سپس فایل `haarcascade_frontalface_default.xml`

را به منظور انجام عملیات تشخیص چهره مورد استفاده قرار می دهد.

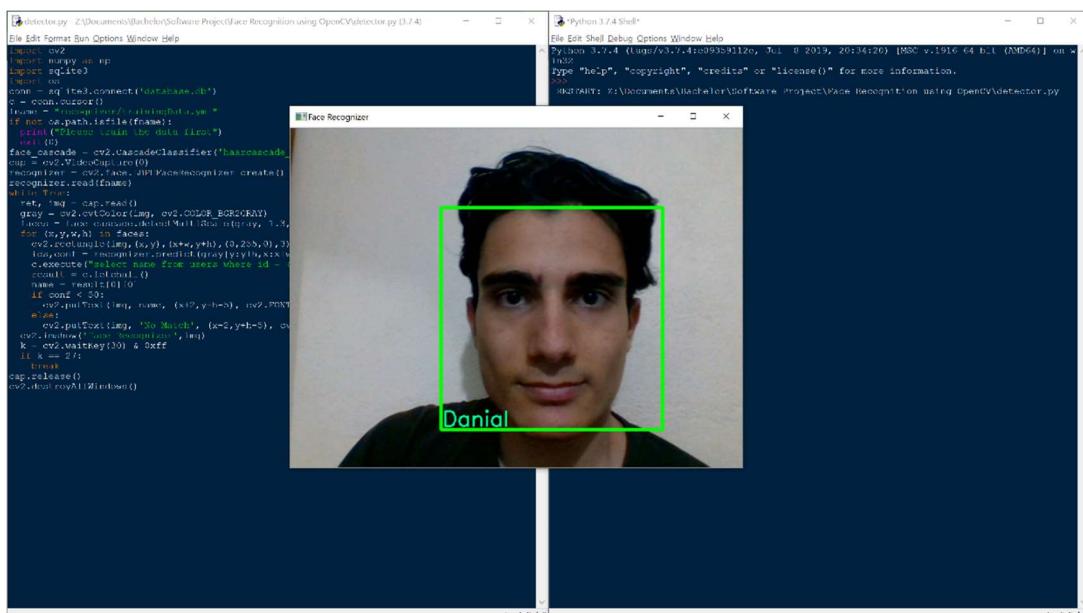
در مرحله بعد سیستم با استفاده از دوربین موجود (وبکم لپ تاپ) فرآیند شناسایی چهره ها را آغاز می کند، به دنبال آغاز این فرآیند سیستم با رویت کردن هر چهره اطراف آن یک مربع ترسیم می کند و با نمونه برداری از چهره های مشاهده شده و مقایسه آنها با پایگاه اطلاعاتی سعی در شناسایی صاحب آن چهره دارد.

اگر که شناسایی چهره با موفقیت همراه بود اطلاعاتی که در هنگام ضبط چهره شخص به دیتابیس داده شده در پایین کادر به نمایش در می آید، در اینجا تنها اطلاعات ذخیره شده برای هر چهره موجود یک عنوان شامل نام و نام خانوادگی او است، و اگر چهره تشخیص داده شده شناسایی نشد عبارت **No Match** نمایش می یابد.

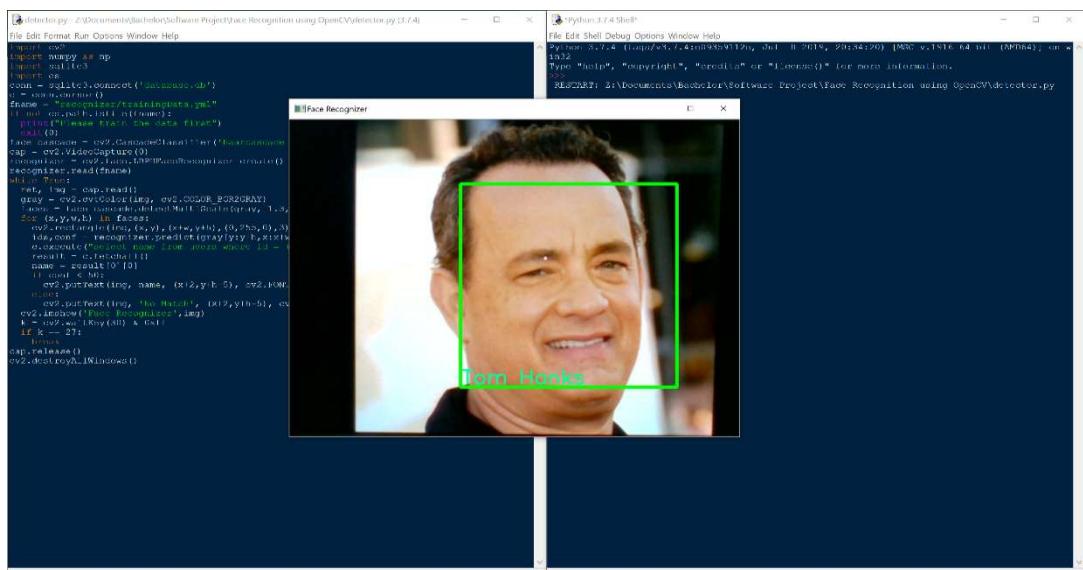
در ادامه تصاویری از نحوه تشخیص چهره توسط برنامه آورده شده است. در این پروژه برای تشخیص چهره از دوربین ویکم **(HD Web Camera 0.9MP (1280×720** لپتاپ استفاده شده.

نتائج (تصاویر):

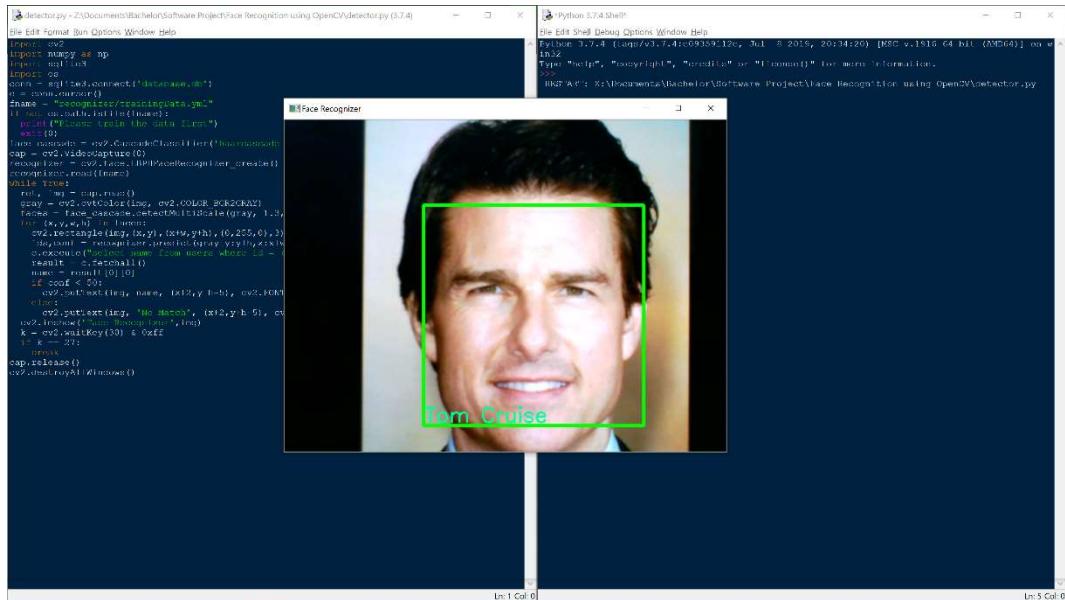
تشخیص چهره یک نمونه واقعی:



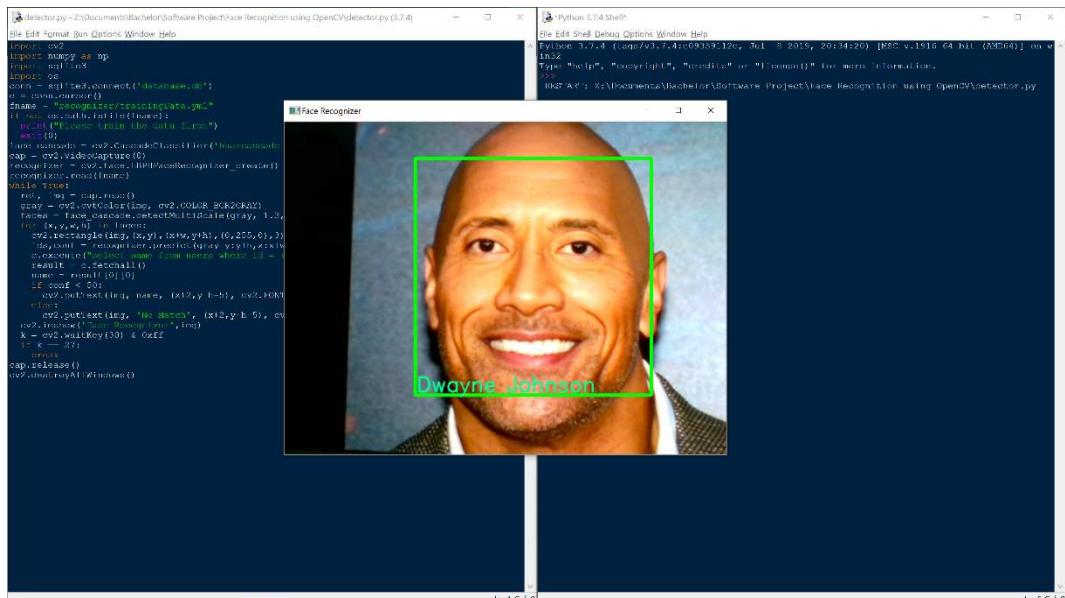
تشخیص چهره بر روی تصاویر:



Tom Hanks

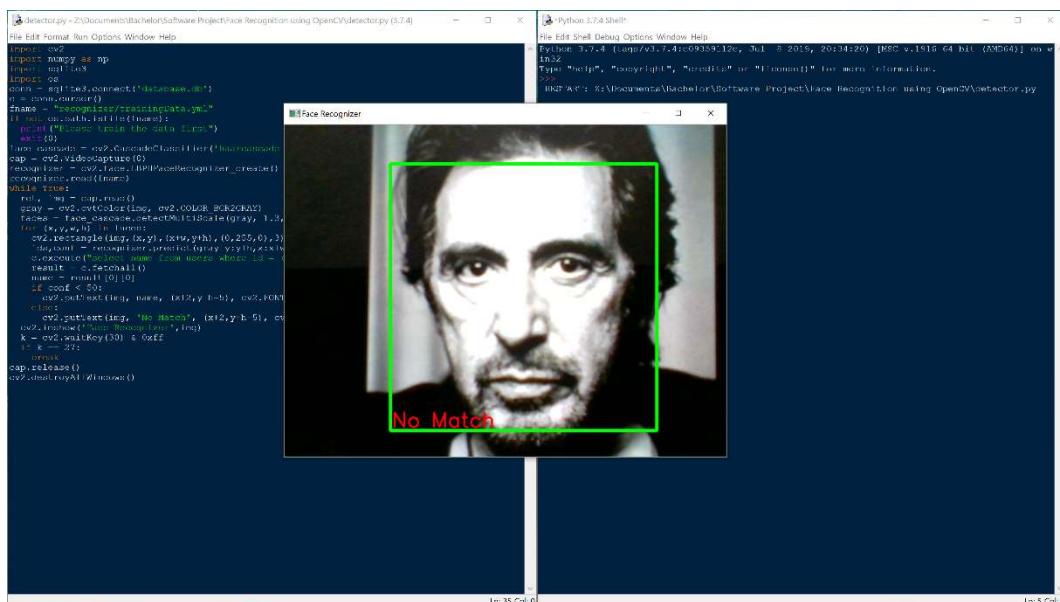
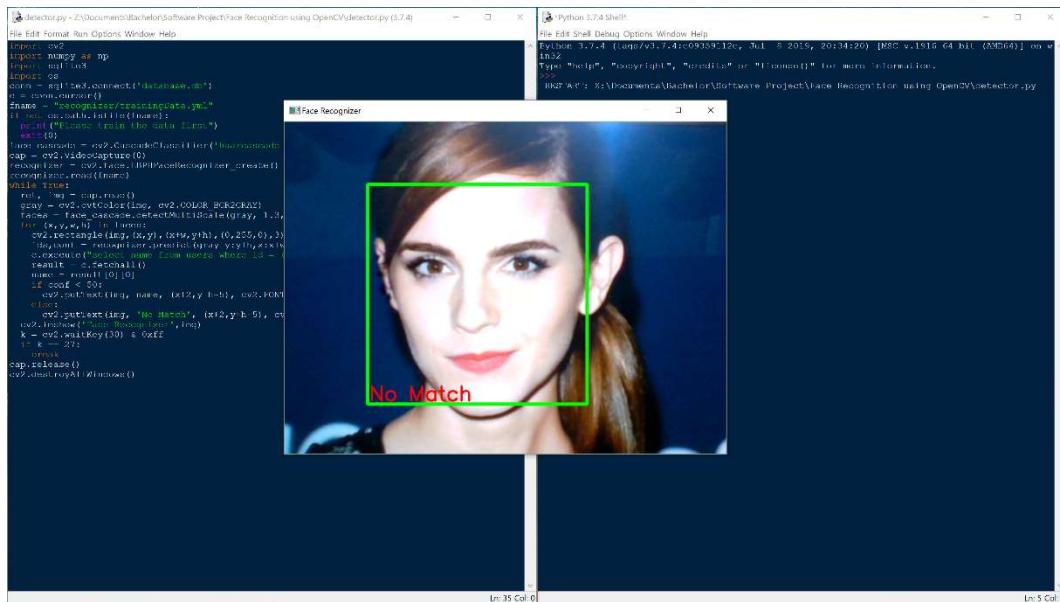


Tom Cruise



Dwayne Johnson

تشخیص چهره تصاویر ناشنا:



Related Works:

کارهای مرتبط:

بخش کارهای مرتبط شامل چکیده (Abstract) مقالات علمی مرتبط با موضوع و حوزه فعالیت این پژوهه است.

[1] Low Resolution Face Recognition Using a Two-Branch Deep Convolutional Neural Network Architecture

تشخیص چهره وضوح پایین با استفاده از یک معماری شبکه عصبی کانولوشن^۱ عمیق دو انشعابی

در اینجا یک روش بدیع برای تشخیص چهره وضوح پایین با استفاده از شبکه‌های عصبی کانولوشن عمیق (DCNNs) پیشنهاد می‌شود. معماری پیشنهادی از دو بخش DCNN تشکیل شده است تا تصاویر چهره با وضوح بالا و پایین را در یک فضای مشترک با تحولات غیرخطی ترسیم کنند. بخش مربوط به تبدیل تصاویر با وضوح بالا از ۱۴ لایه تشکیل شده است و بخش دیگر که تصاویر چهره با وضوح پایین را به فضای مشترک نگاشت می‌کند شامل یک شبکه ۵ لایه با وضوح فوق العاده متصل به یک شبکه ۱۴ لایه است. فاصله بین ویژگی‌های تصاویر با وضوح بالا و پایین، پردازش پسرو (backpropagated) برای آموزش شبکه‌ها است. روش پیشنهادی در مجموعه داده‌های FERET، LFW و MBGC مورد ارزیابی قرار گرفته و با روش‌های رقبای پیشرفته مقایسه شده است. ارزیابی‌های تجربی گستره نشان می‌دهد که روش ارائه شده به طور قابل توجهی عملکرد تشخیص را به ویژه برای تصاویر کاوشگر صورت با وضوح بسیار کم (۰.۵٪ در دقت تشخیص) بهبود می‌بخشد. علاوه بر این، می‌تواند یک تصویر با

^۱Convolution به معنای همتابی، پیچش - نوعی محاسبه پردازش تصویر که توسط یک ماتریس شرح داده می‌شود، با این فرض که می‌خواهید جزئیات دقیقی را از یک تصویر بیرون بکشید. یک روش برای انجام این کار، افزودن تفاوت میان هر سلول تصویری و همسایگانش است. اگر سلول‌های تصویری را به عنوان اعدادی در نظر بگیرید که روشنایی آنها را بیان می‌کنند، می‌توانید ماتریس همتابی را به کار ببرید.

وضوح بالا از تصویر مربوط به کاوشگر با وضوح پایین که قابل مقایسه با روش‌های فوق العاده وضوح عالی از نظر کیفیت بصری است، بازسازی کند.

[2] Face-Specific Data Augmentation for Unconstrained Face Recognition

افزایش اطلاعات خاص چهره برای تشخیص چهره بدون محدودیت

دو مسئله به عنوان کلید توسعه سیستم‌های تشخیص چهره مؤثر شناسایی می‌شوند: حداقل رساندن تغییرات ظاهری تصاویر آموزش و به حداقل رساندن تغییرات ظاهری در تصاویر تست. اولی برای آموزش سیستم به هر نوع تغییر ظاهری که در نهایت با آن رو برو خواهد شد لازم دارد و اغلب با جمع آوری مجموعه‌های آموزشی گسترده با میلیون‌ها تصویر چهره مورد بررسی قرار می‌گیرد. حالت دوم شامل اشکال مختلف عادی سازی ظاهر برای از بین بردن عوامل مزاحم گیج کننده در زمان تست و ساده سازی مقایسه چهره‌های تست است. در اینجا تکنیک‌های بدیع افزایش داده چهره خاص کارآمد را توصیف می‌کنیم و نشان می‌دهیم که برای هر دو هدف ایده آل هستند. با استفاده از دانش چهره‌ها، شکل‌های سه بعدی و ظاهر آنها، موارد زیر را نشان می‌دهیم: (الف) می‌توانیم داده‌های آموزشی را برای تشخیص چهره با تنوع ظاهری خاص، به طور مصنوعی بهبود دهیم. (ب) این داده‌های آموزش مصنوعی می‌توانند به صورت آنلاین (درون خطی) تولید شوند، در نتیجه نیازهای انبوه ذخیره سازی مجموعه‌های آموزشی در مقیاس بزرگ را کاهش می‌دهد و آموزش را برای بسیاری از تغییرات ظاهری ساده می‌کند. سرانجام، (ج) تکنیک‌های سریع تقویت داده‌ها می‌توانند در زمان تست برای کاهش تغییرات ظاهری و بهبود نمایش چهره استفاده شوند. با همدیگر، با جدیدترین تکنیک‌های فنی، یک خط [لوله] شناسایی چهره بسیار مؤثر را توصیف می‌کند که در زمان انقیاد (**submission**)، نتایج پیشرفته‌ای را در چندین معیار بدست می‌آورد.

[3] Automatic Attendance System Using Deep Learning Framework

سیستم حضور و غیاب خودکار با استفاده از چارچوب یادگیری عمیق

حضور و غیاب در کلاس‌های بزرگ کار دشوار، تکراری و زمان ارزشمند کلاس را هدر می‌دهد. برای جلوگیری از این مشکلات، یک سیستم حضور و غیاب خودکار را با استفاده از چارچوب یادگیری عمیق پیشنهاد می‌شود. سیستم حضور و غیاب خودکار بر اساس پردازش تصویر شامل دو مرحله است: یافتن چهره و تشخیص چهره. یافتن و تشخیص چهره مسائل مورد تحقیق خوبی در حوزه بینایی رایانه هستند، اگرچه به دلیل تغییرات زیاد در موقعیت‌ها، شرایط مختلف نورپردازی و انسداد هنوز حل نشده‌اند. در این کار، از مدل پیشرفته کشف چهره برای یافتن چهره‌ها و یک معماری شناسایی جدید برای شناسایی چهره‌ها استفاده کرده‌ایم. شبکه تأیید چهره پیشنهادی از شبکه‌های پیشرفته سطحی‌تر است و به عملکرد تشخیص چهره مشابه‌ای دست یافته است. ما به 98.67% در LFW و 100% در داده‌های classroom دست یافته‌ایم. داده‌های کلاس برای اجرا (پیاده سازی) عملی شبکه کامل در طی این کار توسط ما ساخته شده است.

[4] Effect of subject's age and gender on face recognition results

تأثیر سن و جنسیت فرد بر نتایج تشخیص چهره

امروزه مکان‌های بیشتر و بیشتری به احراز هویت احتیاج دارند. تشخیص چهره یک فناوری کامل برای تحقیقات تأیید هویت است. صحبت تشخیص یک شاخص مهم برای ارزیابی الگوریتم‌های احراز هویت است. به منظور بهبود دقت در تأیید هویت، از چهره پیشرفته (قبلاتهیه شده) استفاده می‌شود. الگوریتم تشخیص خصوصیات یک روش مؤثر است، اما همچنین یک الگوریتم مفید برای بررسی عوامل مؤثر بر ویژگی‌های صورت است. بنابراین، بسیاری از محققان نتایج تشخیص را بر اساس نکات صورت، نور و سایر عوامل مورد مطالعه قرار می‌دهند. این مقاله همچنین یک مطالعه در مورد عوامل مؤثر در تشخیص چهره است، به طور عمدۀ با بررسی تأثیر سن و عوامل جنسیتی در نتایج احراز هویت و استفاده از روش یادگیری عمیق برای طبقه‌بندی ویژگی‌های صورت. نتایج شبیه سازی نشان می‌دهد که میانگین تشخیص به 83.73% می‌رسد. همچنین، این مقاله به بررسی تأثیر سن و جنسیت بر نتایج طبقه‌بندی می‌پردازد. نتایج نشان می‌دهد که اثر تشخیص در مردان میانسال در مبحث افراد مذکور کمتر از جوانان و سالمندان است. تأثیر سن در شناخت زنان اثر چندان زیادی ندارد. مردها نسبت به زنان میزان تشخیص بالاتری دارند.

[5] A Comprehensive Study on Center Loss for Deep Face Recognition

یک مطالعه جامع در مورد فقدان مرکز (تمرکز) تشخیص چهره عمیق

شبکه‌های عصبی کانولوشن عمیق (CNN) که با از بین رفتن بیشینه نرم‌افزاری (Softmax) آموزش دیده‌اند، در تعدادی از مشکلات شناخت مجموعه بسته (Close-Set) به موفقیت‌های چشمگیری رسیده‌اند، به عنوان مثال: تشخیص شی، تشخیص حرکت و غیره. بر خلاف کارهای مجموعه بسته، تشخیص چهره یک مجموعه باز (Open-Set) است که در آن کلاس‌های تست (اشخاص) معمولاً با آنهايي که در آموزش هستند متفاوت است. اين مقاله با توسعه از فقدان مرکز، به ویژگی‌های مجموعه باز تشخیص چهره می‌پردازد. به طور خاص، از دست دادن مرکز به طور همزمان يك مرکز برای هر کلاس ياد می گيرد، و فاصله بين ویژگی‌های عمیق تصاویر چهره و مراکز کلاس مربوطه را کاهش (penalize: حذف یا کاهش تاثیر یک پارامتر در معادله) می‌دهد. آموزش با فقدان مرکز، CNN‌ها را قادر می‌سازد ویژگی‌های عمیق را با دو خاصیت مطلوب استخراج کنند: تفکیک بین کلاس و فشردگی درون کلاس. بعلاوه، فقدان مرکز را از دو جنبه گسترش می‌دهیم. ابتدا، ما پارامترهای مشترکی را بین بیشینه نرم‌افزار و از دست رفتن مرکز اتخاذ می‌کنیم تا پارامترهای اضافی معرفی شده توسط مراکز را کاهش دهیم. دوم، ما مفهوم مرکز را از یک نقطه به یک منطقه در فضای تعییه تعمیم می‌دهیم، که بعداً به ما امکان می‌دهد تا تغییرات درون کلاس را به خود اختصاص دهیم. از دست رفتن مرکز پیش‌رفته، قدرت تمایزآمیز ویژگی‌های عمیق را به طور قابل توجهی افزایش می‌دهد. نتایج تجربی نشان می‌دهد که روش ما در چندین معیار مهم تشخیص چهره، از جمله Labeled Faces in the Wild و YouTube Faces 1، MegaFace Challenging و IJB-A Janus به دقت بالایی دست می‌یابد.

[6] Deep class-skewed learning for face recognition

یادگیری عمیق کلاس اریب (کج، منحرف) برای تشخیص چهره

مجموعه داده‌های چهره اغلب توزیع کلاس کاملاً اریب دارند، یعنی، کلاس‌های غنی (Rich) حاوی موارد زیادی هستند، در حالی که فقط محدود تصاویری به کلاس‌های فقیر (Poor) تعلق دارند. برای کاهش این مسئله، در این مقاله یادگیری کلاس اریب از دو جنبه بررسی می‌شود: تقویت ویژگی و نرمال سازی ویژگی. برای مقابله با مشکل توزیع بدون توازن، یک روش تقویت ویژگی جدید را با عنوان افزایش ویژگی‌های حاشیه‌ای بزرگ (LMFA) برای تقویت ویژگی‌های سخت و مساوی توزیع کلاس ارائه می‌شود، که منجر به مرزهای طبقه بندی متوازن بین طبقات غنی و فقیر می‌شود. با در نظر گرفتن شکاف توزیع بین ویژگی‌های آموزش و تست، یک عادی سازی ویژگی جدید به نام عادی سازی دامنه قابل انتقال (TDN) برای عادی سازی ویژگی‌های خاص دامنه برای اطاعت از توزیع یکسان گاوی‌ها (Gaussian: گوسی در ریاضیات) و تقویت تعییم ویژگی پیشنهاد شده است. تست‌های گسترده‌ای روی پنج مجموعه داده محبوب تشخیص چهره از جمله LFW، CFP، YTF و MegaFace و AgeDB تأثیرگذارد. ما به طور قابل توجهی و یا بهتر از روش‌های مدرن، که اثربخشی ویژگی‌های متعادل یادگیری کلاس پیشنهادی ماست، به نتایج قابل توجهی رسیده‌ایم.

[7] DLFace: Deep local descriptor for cross-modality face recognition

DLFace: توصیف کننده عمیق محلی برای تشخیص چهره متقابل

شناسایی چهره متقابل با هدف شناسایی چهره‌ها در بین روش‌های مختلف از جمله تطبیق طرح با عکس، تصاویر چهره با وضوح پایین با تصاویر با وضوح بالا و در نزدیکی تصاویر مادون قرمز با تصاویر روشنایی بصری انجام می‌شود به دلیل شکاف کیفیت ناشی از بافت، وضوح و تغییرات روشنایی. رویکردهای موجود یا از رویکردهای دست ساز استفاده می‌کنند که ویژگی توزیع اطلاعات ذاتی را نادیده می‌گیرد، یا از الگوریتم‌های مبتنی بر یادگیری عمیق در تصاویر جامع صورت که از اطلاعات محلی صورت استفاده نکرده‌اند. این مقاله، یک چارچوب یادگیری محلی توصیف کننده عمیق برای تشخیص چهره متقابل ارائه می‌دهد، که هدف آن یادگیری اطلاعات محلی تفکیک کننده و فشرده به طور مستقیم از تکه‌های (Patches) صورت خام است. برای از بین بردن شکاف کیفیت در سطح تکه‌های محلی، که پس از آن در شبکه‌های عصبی کانولوشن برای استخراج توصیفگر محلی عمیق یکپارچه شده است، ضریب متقابل جدیدی ارائه شده است. توصیفگر محلی پیشنهادی را می‌توان به راحتی در هر سیستم سنتی تشخیص چهره به کار برد، و از Fisherface به عنوان نمونه در مقاله استفاده می‌شود. تست‌های گسترده در شش مجموعه داده شناسایی چهره متقابل به طور گسترده استفاده می‌شود که نشان از برتری روش پیشنهادی نسبت به روش‌های مدرن دارد.

[8] Detecting and Mitigating Adversarial Perturbations for Robust Face Recognition

کشف و کاهش اختلالات مقابله برای تشخیص چهره قادرمند

مدل‌های مبتنی بر معماری شبکه عصبی عمیق (DNN) از قدرت بیان و ظرفیت یادگیری بالایی برخوردار هستند. با این حال، آنها در واقع یک روش جعبه سیاه هستند از آنجایی که تهیه فرمول ریاضی تابع‌هایی که در بسیاری از لایه‌های ارائه آموخته می‌شود کار ساده‌ای نیست. با درک این امر، بسیاری از محققان شروع به طراحی روش‌هایی برای استفاده بهینه از اشکالات الگوریتم‌های مبتنی بر یادگیری عمیق کرده‌اند که قابلیت اطمینان آنها را مورد سوال قرار داده و یکتایی‌های آنها را آشکار می‌کند. در این مقاله، سعی شده از سه جنبه مرتبط با قابلیت اطمینان DNN‌ها برای تشخیص چهره اقدام به حل شود: (الف) ارزیابی تأثیر معماری‌های عمیق برای تشخیص چهره از نظر آسیب پذیری در برابر تعرض‌ها، (ب) شناسایی یکتایی‌ها با توصیف رفتار پاسخ غیر طبیعی فیلتر در لایه‌های پنهان شبکه‌های عمیق و (ج) اصلاحات در خط [لوله] (Pipeline) پردازش برای کاهش مشکل. ارزیابی تجربی با استفاده از چندین DNN مبتنی بر منبع باز و سه پایگاه داده چهره در دسترس عموم نشان می‌دهد که عملکرد الگوریتم‌های تشخیص چهره مبتنی بر یادگیری عمیق در صورت وجود چنین تحریفاتی می‌تواند لطمه فراوانی بخورد. همچنین رویکردهای پیشنهادی را در چهار تحریف شبه تدریجی (Quasi-Imperceptible) موجود ارزیابی می‌کنیم: DeepFool، اختلالات عمومی مقابله، L2 و EAD (Elastic-Net). روش پیشنهادی با طراحی مناسب طبقه بندی با استفاده از پاسخ لایه‌های پنهان در شبکه، می‌تواند هر دو نوع تعرض را با دقت بسیار بالایی تشخیص دهد. سرانجام، اقدامات مقابله مؤثری برای کاهش تأثیر تعرض مقابله و بهبود قابلیت اطمینان کلی تشخیص چهره مبتنی بر DNN ارائه می‌شود.

[9] Learning Affective Video Features for Facial Expression Recognition via Hybrid Deep Learning

یادگیری موثر ویژگی‌های ویدیو برای تشخیص حالت چهره از طریق یادگیری عمیق ترکیبی

یکی از مهم‌ترین مسائل مهم تشخیص حالت چهره (FER) در توالی‌های ویدئویی، استخراج ویژگی‌های ویدئویی متمایز کننده فضا و زمان از تصاویر حالت چهره در توالی‌های ویدئویی است. در این مقاله، یک روش جدید از FER در توالی‌های ویدئویی از طریق یک مدل یادگیری عمیق ترکیبی پیشنهاد می‌شود. روش پیشنهادی برای اولین بار از دو شبکه عصبی کانولوشن عمیق (CNN)، شامل پردازش فضایی (Spatial: فاصله‌ای) CNN تصاویر استاتیک صورت و یک پردازش شبکه زمانی (موقعی) CN تصاویر جریان نوری استفاده می‌کند، که برای یادگیری جداگانه ویژگی‌های مکانی و زمانی سطح بالا در بخش‌های ویدیویی تقسیم شده. این دو CNN در مجموعه داده‌های حالت چهره ویدیویی هدف از یک مدل CNN از قبل آموزش دیده به خوبی تنظیم شده‌اند. سپس، ویژگی‌های مکانی و زمانی سطح بخش (Segment-Level) به دست آمده در یک شبکه ادغام (Fusion) عمیق ساخته شده با یک مدل باور (Belief) عمیق (DBN) یکپارچه شده است. این شبکه ادغام عمیق برای یادگیری مشترک ویژگی‌های فضایی و زمانی و متمایز ساز استفاده می‌شود. سرانجام، یک همبستگی متوسط بر روی ویژگی‌های سطح بخش DBN یاد گرفته شده در یک دنباله ویدیویی انجام می‌شود، تا یک ویژگی سراسری ویدیوی با طول ثابت را تولید کند. بر اساس بازنمایی ویژگی‌های ویدیویی سراسری، یک ماشین بردار پشتیبانی خطی (SVM) برای کارهای طبقه بندی حالت صورت استفاده می‌شود. انجام تست‌های گستردۀ در سه مجموعه داده عمومی حالت چهره مبتنی بر ویدئو، یعنی BAUM-1، RML و MMI، اثربخشی روش پیشنهادی را نشان می‌دهد، که نسبت به تکنولوژی‌های جدید عملکرد بهتری داشته است.

[10] Deep-Feature Encoding-based Discriminative Model for Age-invariant Face Recognition

مدل متمایز ساز مبتنی بر رمزگذاری عمیق برای تشخیص چهره تغییر ناپذیر با سن

تغییر صورت (سالخوردگی) یکی از مشکلات اساسی سیستم‌های تشخیص چهره به دلیل تغییرات زیاد درون شخصی ناشی از افزایش سن است. یک چالش بزرگ ایجاد یک نمایه از ویژگی‌های کارآمد، متمایز ساز و چارچوب مطابقت است که نسبت به تغییرات پیری صورت قدرتمند است. در این مقاله، یک مدل متمایز ساز مبتنی بر رمزگذاری با ویژگی‌های عمیق قدرتمند برای تشخیص چهره تغییر ناپذیر با سن پیشنهاد می‌شود. این روش با استفاده از یک مدل عمیق CNN از قبل آموزش دیده ویژگی‌های عمیق سطح بالا را یاد می‌گیرد. این ویژگی‌ها سپس با یادگیری یک رمزنامه (Codebook) کدگذاری می‌شوند، که هر یک از ویژگی‌ها را به یک کلمه رمز (Codeword) S-بعدی متمایز برای نمایش تصویر تبدیل می‌کند. با درج اطلاعات محلی در کل فرایند یادگیری، یک فرم بسته (Close-Form) راه حل برای هر دو مرحله به روزرسانی و رمزگذاری کد به دست می‌آید. از آنجایی که ویژگی‌های یک فرد در سنین مختلف باید دارای همبستگی‌های خاصی باشد، از تجزیه و تحلیل همبستگی کانونی برای ترکیب کردن جفت ویژگی‌های آموزش، برای دو سن مختلف استفاده می‌شود تا رمزنامه از نظر پیشرفت سن متمایز باشد. در مرحله تست، ویژگی‌های گالری و تصویر پرس و جو با استفاده از رمزنامه آموخته شده رمزگذاری می‌شوند. سپس از نقشه برداری خطی مبتنی بر رگرسیون (سیر قهقرایی، بسرفت) خطی برای تطبیق چهره استفاده می‌شود. روش خود را در سه مجموعه داده پیری چهره که به صورت عمومی در دسترس هستند، FGNET و MORPH Album 2 و Large Age-Gap (LAG) ارزیابی می‌کنیم. نتایج تجربی نشان می‌دهد که روش پیشنهادی بهتر از روش‌های مختلف جدید تشخیص چهره تغییر ناپذیر با سن است و از نظر دقت تشخیص، درجه 1 است.

References:

: منابع

[1] وبسایت فرادارس، آموزش پردازش تصویر با OpenCV توسط مهندس الهام شعبانی نیا

<https://faradars.org/courses/fvimg9405-opencv>

[2] OpenCV Face Recognition by Adrian Rosebrock

<https://www.pyimagesearch.com/2018/09/24/opencv-face-recognition/>

[3] Setup Virtual Environment and Install Necessary Dependencies

<https://www.pytorials.com/face-recognition-using-opencv-part-1/>

[4] Face Detection and Face Recognition Using OpenCV – Training

<https://www.pytorials.com/face-recognition-using-opencv-part-2/>

[5] Face Recognition OpenCV – Fetching Data From SQLite

<https://www.pytorials.com/face-recognition-using-opencv-part-3/>

[6] Face Recognition: Understanding LBPH Algorithm

<https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b>

[7] Read, Write and Display a video using OpenCV (C++ / Python)

<https://www.learnopencv.com/read-write-and-display-a-video-using-opencv-cpp-python/>

[8] Chapter 1. Basic Image Handling and Processing

<https://www.oreilly.com/library/view/programming-computer-vision/9781449341916/ch01.html>

[9] OpenCV API Reference, User Interface Documentation

https://docs.opencv.org/2.4/modules/highgui/doc/user_interface.html

[10] Questions and Code Guide <https://stackoverflow.com/>

[11] How to use Pillow, a fork of PIL

<https://www.pythonforbeginners.com/gui/how-to-use-pillow>

[12] How to Make Predictions with scikit-learn

<https://machinelearningmastery.com/make-predictions-scikit-learn/>

[13] Converting this 16-bit grayscale image to 'L' mode destroys it

<https://github.com/python-pillow/Pillow/issues/3011>

[14] Python Variables: Declare, Concatenate, Global & Local

<https://www.guru99.com/variables-in-python.html>

[15] Data types

<https://docs.scipy.org/doc/numpy-1.13.0/user/basics.types.html#array-types-and-conversions-between-types>

[16] Python Arrays https://www.w3schools.com/python/python_arrays.asp

[17] Erfan Zangeneh, Mohammad Rahmati and Yalda Mohsenzadeh. "Low Resolution Face Recognition Using a Two-Branch Deep Convolutional Neural Network Architecture". Expert Systems With Applications. 2019.

[18] Iacopo Masi, Anh Tuân Trần, Tal Hassner, Gozde Sahin and Gérard Medioni. "Face-Specific Data Augmentation for Unconstrained Face Recognition". International Journal of Computer Vision. 2019.

[19] Pinaki Ranjan Sarkar, Deepak Mishra and Gorthi R. K. Sai Subhramanyam. "Automatic Attendance System Using Deep Learning Framework". Machine Intelligence and Signal Analysis. 2019.

[20] Shifeng wu and Dahu Wang. "Effect of subject's age and gender on face recognition results". J. Vis. Commun. Image R. 2019.

[21] Yandong Wen, Kaipeng Zhang, Zhifeng Li and Yu Qiao. "A Comprehensive Study on Center Loss for Deep Face Recognition". International Journal of Computer Vision. 2019.

[22] Pingyu Wang, Fei Sua, Zhicheng Zhaoa, Yandong Guo, Yanyun Zhao and Bojin Zhuang. "Deep class-skewed learning for face recognition". Neurocomputing. 2019.

[23] Chunlei Peng, Nannan Wang, Jie Li and Xinbo Gao. "DLFace: Deep local descriptor for cross-modality face recognition". Pattern Recognition. 2019.

[24] Gaurav Goswami, Akshay Agarwal, Nalini Ratha, Richa Singh and Mayank Vatsa. "Detecting and Mitigating Adversarial Perturbations for Robust Face Recognition". International Journal of Computer Vision. 2019.

[25] Shiqing Zhang, Xianzhang Pan, Yueli Cui, Xiaoming Zhao and Limei Liu. "Learning Affective Video Features for Facial Expression Recognition via Hybrid Deep Learning". IEEE Access. 2019.

[26] M. Saad Shakeel and Kin-Man Lam. "Deep-Feature Encoding-based Discriminative Model for Age-invariant Face Recognition". Pattern Recognition. 2019.