به نام خدا

گزارش پروژه میان ترم
پردازش تصویر

استاد: آقای دکتر کیانی

نویسنده: علی ناظری
شماره دانشجویی: ۹۸۳۶۱۳۰۵۶

تاریخ: ۱۴۰۲/۲/۲۹

فهرست

مقدمه

در این پروژه یک مدل بدست آمد که بتواند لبخند را تشخیص دهد. داده ها شامل ۴۰۰۰ عکس با label می‌باشند؛ عکس هایی تکی از افراد هستند. در بخش اول پروژه باید فقط صورت افراد شناسایی و جدا شود سپس در بخش دوم ویژگی ها استخراج و در بخش آخر مدل پیاده سازی شود و ارزیابی شود.

بخش اول

نیاز است تا صورت افراد جدا از بخش های دیگر باشند تا عملیات یادگیری با ویژگی هایی که استخراج می شود کیفیت بالاتری داشته باشد. برای تشخیص چهره افراد از یک مدل آماده استفاده شده است. این مدل haarcascades_frontalface_default نام دارد و در فایل ها موجود است. با دادن هر عکس به این مدل درصورتی که یک چهره تشخیص داده شده بود عکس بریده می شد و ذخیره می شد در غیر این صروت با تغییر تنظیمات مدل این عمل تکرار می شد تا یک چهره پیدا کند(فایل get_faces.py). در تعداد ۴۸تا از تصاویر چهره ای تشخیص داده نشد. سپس تمامی عکس ها بررسی شدند که در تمام آن ها به درستی چهره تشخیص داده شده با شد در اینجا هم عده ای از تصاویر خراب بودند. سپس به صورت در ستی با drag & drop این کار انجام شد و تصاویر ذخیره و جایگذاری شدند(فایل select by hand.py). در پایان ۴۰۰۰ عکس فقط از صورت افراد بدست آمد.

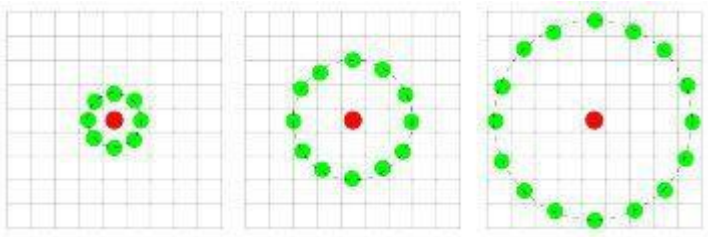
بخش دوم

داده ها که عکس ها هستند آماده است. حالا باید ویژگی هایی که برای تشخیص لبخند کمک می کنند استخراج شوند. همان طور که در کلاس مطرح شده بود دو ویژگی HOG و LBP برای این کار مناسب هستند. در ویژگی HOG زاویه ها در عکس مشخص می شود و LBP بافت تشخیص داده می شود. تنظیمات برای هر کدام از آن ها به صورت زیر می باشد:

```
hog_features = hog(img, orientations=30, pixels_per_cell=(16, 16),cells_per_block=(1, 1), visualize=False, channel_axis=-1) # HOG
lbp_c = LocalBinaryPatterns(24, 2) # number of points, radius
```

ابتدا یک نمونه از کلاس LocalBinaryPatterns ساخته می شود و ورودی ها داده می شوند سپس برای هر عکس تابع این کلاس صدا زده می شود. در این تابع ابتدا LBP بدست می آید سپس روی ماتریس خروجی histogram زده می شود و در پایان normalize می شود. نرمال سازی تمام اعداد را بین ۰ و ۱ می آورد.

در عکس زیر نحوه کار کردن تعداد neighbor points و radius مشاهده می شود. مقدار radius تعداد پیکسل ها از مرکز است و neighbor points تعداد نقاطی که برای انجام این محاسبات در این محدوده در نظر گرفته می شود. اگر مانند مورد استفاده شده در نظرگرفته شود تعداد نقاط همسایگی ۲۴ و شعاع ۲ است که یعنی تمامی نقاط در شعاع ۲ با مقدار مرکزی مقایسه می شوند و یک رشته باینری می سازند و سپس این کار دوباره با حرکت این محدوده انجام می شود و در نهایت با تبدیل این مقادیر به عدد حقیقی یک ماتریس دیگر بدست می آید که با استفاده از histogram تبدیل به یک vector می شود و این مقادیر ویژگی های LBP می باشند.



HOG به این صورت بد ست می آید که gradient عکس محا سبه شده و سپس از این زاویه های بد ست آمده gradient histogram محاسبه می شود و در نهایت normalize و flat می شود. در تابعی که استفاده شده ورودی اول عکس مورد نظر است ورودی دوم تعداد شیب ها(gradiant) است و تعداد پیکسل ها برای هر سلول و تعداد سلول های هر بلاک ا ست و آخرین ورودی تعداد کانال ها را م شخص می کند که با بقدار ۱- م شخص می شود که از سه کانال ت شکیل شده ا ست. تمامی عملیات محا سبه زاویه ها و محا سبه histogram و نرمال سازی و تخت کردن به صورت یکجا در این تابع انجام می شود.

در عکس زیر این عملیات بصری سازی شده است.

در HOG ۱۰۸۰ ویژگی بدست آمد و در LBP ۲۶ ویژگی سپس این دو برای هر عکس به هم متصل شدند و ماتریس ویژگی ها را بوجود آوردند و با خواندن فایل labels عملیات استخراج ویژگی و تگ ها به پایان می رسد.

بخش سوم

در این بخش با استفاده است کتابخانه scikitlearn ماتریس ویژگی ها و تگ ها به دو قسمت train و test تقسیم شدند و با مدل svm با kernel های مختلف یادگیری انجام شد. در kernel = rbf مدل دقت بالایی داشت و این دقت هم در تست و هم در ماتریس یادگیری قابل مشاهده بود. ولی هنگامی که عکسی خارج از این دیتاست داده می شود نمی توانست آن را تشخیص دهد در نتیجه می توان گفت که این مدل overfit شده بود. دومین Kernel که بیشترین دقت را داشت linear بود با دقت ۸۲ که بر خلاف مدل rbf می توانست لبخند را در تصاویر دیگری هم شناسایی کند پس این مدل انتخاب شد.

در پایان با استفاده از کتابخانه pickle مدل با نام finalized_smile_detection_SVC_model ذخیره شد.

سپس در آخر با استفاده از metrics های کتابخانه sklearn این مدل ارزیابی شد.

```
Score =  <bound method ClassifierMixin.score of SVC(kernel='linear', random_state=13)>
Confusion Matrix =
[[368  79]
 [ 98 455]]
Accuracy =  0.823
F1 =  0.8371665133394665
Precision =  0.8520599250936329
Recall =  0.8227848101265823
```

همچنین در آخرین بلاک کد می توانید مسیر عکس دلخواهی را وارد کنید و پیشبینی مدل را ببینید.

فایل ها و فولدر ها

by hand : folder

فایلی که عکس هایی که به صورت دستی صورت آن ها بریده شدند در آن ذخیره شده اند.

---

newimg : folder

شامل تصاویر نهایی که صورت آن ها جدا است(۴۰۰۰ تصویر)

---

labels.txt

تگ های تصاویر

---

get_face.py

جداکردن صورت افراد با مدل آماده

---

select by hand.py

جداکردن صورت افراد به صورت دستی

---

svm_model.ipynb

استخراج ویژگی ها و یادگیری

---

finalized_smile_detection_SVC_model

مدل ذخیره شده نهایی

---

photo.jpg

عکسی از نویسنده برای تست

---

haarcascade_frontalface_default.xml

مدل تشخیص چهره آماده

منابع

https://machinelearningmastery.com/save-load-machine-learning-models-python-scikit-learn/

https://pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/

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

https://www.geeksforgeeks.org/face-detection-using-cascade-classifier-using-opencv-python/

https://scikit-learn.org/stable/modules/svm.html

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

https://scikit-image.org/docs/stable/api/skimage.feature.html#skimage.feature.hog

https://scikit-image.org/docs/stable/auto_examples/features_detection/plot_hog.html#sphx-glr-auto-examples-features-detection-plot-hog-py

https://www.researchgate.net/figure/An-example-of-HOG-descriptor-from-a-facial-image-a-Facial-image-b-Facial-image-with_fig2_320246762

https://scikit-image.org/docs/stable/auto_examples/features_detection/plot_local_binary_pattern.html