پروژه اول درس مباحث ویژه

تشخیص لبخند

استاد:

دکتر کیانی

تهیه کننده:

رضا اعلایی

دانشکده مهندسی کامپیوتر پاییز۱۴۰۳

فهرست

شرح پروژه

در این پروژه، ما قصد داریم به کمک بهره‌گیری از یک مدل از پیش آموزش داده شده، یک مدل هوش مصنوعی برای تشخیص لبخند پیاده‌سازی کنیم.

دیتاست به کار رفته در این پروژه، دیتاست genki۴k است که شامل تصاویری در دو دسته‌ی smile و non smile است.

بستری که برای اجرا کردن پروژه استفاده شده، google colab هست.

اقدامات کلی پروژه

روند کلی این کار به شرح زیر است.

ابتدا به کمک یک face detector آماده، چهره‌های هر عکس را استخراج میکنیم و با crop کردن، در یک بخش جدید ذخیره‌شان فقط تصاویر چهره را ذخیره میکنیم. Face detector استفاده شده در این مورد retinaface است.(توضیحات مربوط به انتخاب هر قسمت در بخش‌های بعد توضیح داده شده است.)

پس از ذخیره تصاویر برش خورده، به کمک الگوریتم‌های alignment تصاویر را align میکنیم.

در مرحله بعد، فایل hdf۵. مدل انتخاب شده‌مان را دانلود میکنیم و در نهایت به آموزش مدل و fine tune کردن آن با دیتاهای خود میپردازیم.

تشخیص چهره

برای این کار از retinaface استفاده شده است. پس از استفاده از مدل‌های مختلف تشخیص چهره، مانند yunet، dlib(با upsampling و بدون آن)، opencv و retina، دقت اشخیص در دو الگوریتم yunet و retina بیشتر بود و چون امکان پردازش با gpu وجود داشت، retina به عنوان تشخیص‌دهنده چهره به کار رفت.

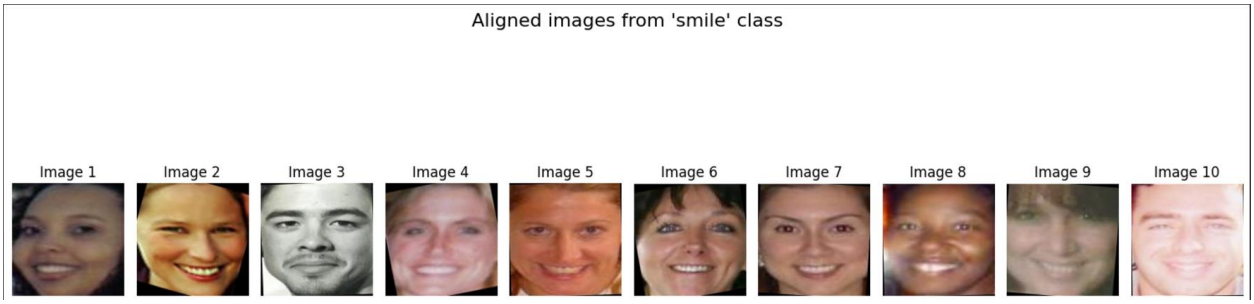
البته دقت بالای retina معایبی هم در این دیتاست داشت. مثلا برخی چهره‌ها از تصاویر موجود در قاب عکس داخل تصویر تشخیص داده میشد که باعث کم‌کیفیت شدن دیتاست میشد. لین مشکلات در مرحله بعد حل شد.

تراز کردن عکس‌ها

در اینجا به کمک لایبرری face-alignment ابتدا لندمارک‌های تصویر برش خورده‌ی چهره را پیدا کرده و سپس، به کمک تابع align_face (که دستی پیاده‌سازی شده) و لندمارک‌های تصویر، آن را تراز میکنیم.

با انجام این کار هم تصاویری که کاملا از روبه رو نیستند صاف میشوند و هم چهره‌های ریز تشخیص داده شده در بخش قبل، به دلیل پیدا نکردن landmarkهای صورت توسط الگوریتم حذف میشوند که به بهتر شدن کیفیت داده‌ها کمک زیادی میکند.

نمونه عکس‌های تراز شده:



مدل از پیش آموزش داده شده

در اینجا از مدل زیر استفاده شده است.

https://github.com/meng۱۹۹۴۴۱۲/Smile_Detection.git

پس از بررسی چندین مدل که در زمینه تشخیص لبخند آموزش داده شده بودند، این مدل برای پایه‌ی آموزش مدل جدید انتخاب شد. معیارهایی که در نظر گرفته شد برای این کار عبارتند از:

دقت نهایی مدل، دیتاستی که مدل با آن آموزش دیده(دیتاست نسبتا خوب با حجم تصاویر تقریبا زیاد).

اگرچه که معماری این مدل lenet است که معماری بسیار قدیمی میباشد، اما با درنظر گرفتن محدودیت سخت‌افزار، حتی با کمک google colab، و دقت نهایی مدل بنظر میرسد معماری بدی برای این کار نباشد.

اگرچه که به‌نظر میرسد به کمک معماری‌های غیر sequntial میتوان به دقت‌های بالاتر هم رسید.

معماری مدل به شکل زیر است:

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_1 (Conv2D)           (None, 28, 28, 20)        520

 activation_1 (Activation)   (None, 28, 28, 20)        0

 max_pooling2d_1 (MaxPoolin  (None, 14, 14, 20)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 14, 14, 50)        25050

 activation_2 (Activation)   (None, 14, 14, 50)        0

 max_pooling2d_2 (MaxPoolin  (None, 7, 7, 50)          0
 g2D)

 flatten_1 (Flatten)         (None, 2450)              0

 dense_1 (Dense)             (None, 500)               1225500

 activation_3 (Activation)   (None, 500)               0

 dense_2 (Dense)             (None, 2)                 1002

 activation_4 (Activation)   (None, 2)                 0

=================================================================
Total params: 1252072 (4.78 MB)
Trainable params: 1252072 (4.78 MB)
Non-trainable params: 0 (0.00 Byte)
```

آموزش مدل نهایی

در نهایت و برای آموزش مدل نهایی، ابتدا تعدادی لایه برای عمل train نهایی خود اضافه میکنیم.

همچنین با اینکه در اینجا خود لایه خروجی، دو کلاسه است، اما به دلیل تغییر لایه‌ها و حذف خروجی اصلی، باید حتما لایه خروجی دو کلاسه را اضافه کنیم.

حال مدل را با adam optimizer با نرخ یادگیری ۱e-۴ کامپایل می‌کنیم(با اجراهای مختلف نرخ یادگیری برابر با این مقدار بیشترین دقت train و validation و همچمین کمترین مقادیر loss را داشت). همچنین loss function این مدل به دلیل دو کلاسه بودن مسئله، binary crossentropy است.

دیتاهای تراز شده نیز در این مرحله به دو دسته‌ی train و validation تقسیم میشوند.

دیتاها همگی rescale میشوند، اما با اجراهای مختلف، تنظیمات دیگر برای Augmentation باعث افت شدید نتیجه‌ی نهایی مدل میشد.

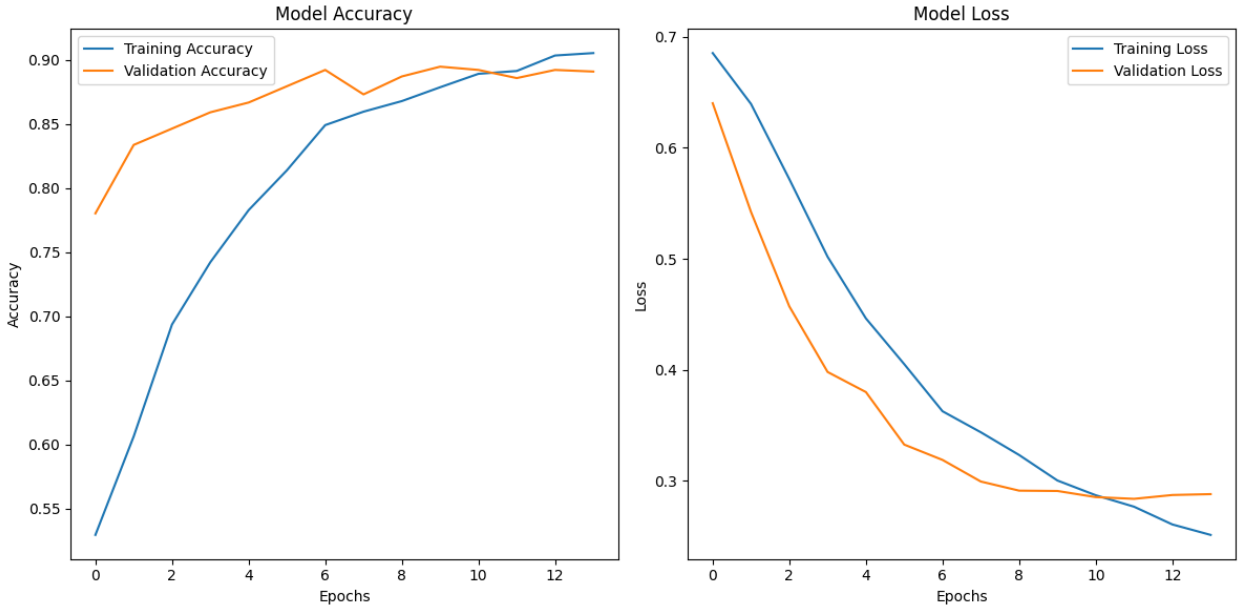
در نهایت و برای آموزش مدل، تعداد epochها ۲۵ در نظر گرفته شد اما امکان early stopping نیز اضافه شد تا در صورتی که مقدار val_loss در حال افزایش است جلوی آموزش بیشتر مدل گرفته شود تا دچار overfit نشویم.

```
Layer 0: conv2d_1_input - Input shape: [(None, 28, 28, 1)], Output shape: [(None, 28, 28, 1)]
Layer 1: conv2d_1 - Input shape: (None, 28, 28, 1), Output shape: (None, 28, 28, 20)
Layer 2: activation_1 - Input shape: (None, 28, 28, 20), Output shape: (None, 28, 28, 20)
Layer 3: max_pooling2d_1 - Input shape: (None, 28, 28, 20), Output shape: (None, 14, 14, 20)
Layer 4: conv2d_2 - Input shape: (None, 14, 14, 20), Output shape: (None, 14, 14, 50)
Layer 5: activation_2 - Input shape: (None, 14, 14, 50), Output shape: (None, 14, 14, 50)
Layer 6: max_pooling2d_2 - Input shape: (None, 14, 14, 50), Output shape: (None, 7, 7, 50)
Layer 7: flatten_1 - Input shape: (None, 7, 7, 50), Output shape: (None, 2450)
Layer 8: dense_1 - Input shape: (None, 2450), Output shape: (None, 500)
Layer 9: activation_3 - Input shape: (None, 500), Output shape: (None, 500)
Layer 10: dense_2 - Input shape: (None, 500), Output shape: (None, 2)
Layer 11: custom_dense1 - Input shape: (None, 2), Output shape: (None, 32)
Layer 12: custom_drop1 - Input shape: (None, 32), Output shape: (None, 32)
Layer 13: custom_dense2 - Input shape: (None, 32), Output shape: (None, 32)
Layer 14: custom_dense3 - Input shape: (None, 32), Output shape: (None, 16)
Layer 15: custom_drop2 - Input shape: (None, 16), Output shape: (None, 16)
Layer 16: custom_dense4 - Input shape: (None, 16), Output shape: (None, 16)
Layer 17: custom_dense5 - Input shape: (None, 16), Output shape: (None, 4)
Layer 18: custom_dense6 - Input shape: (None, 4), Output shape: (None, 1)
```

```
Epoch 1/25
127/127 [==============================] - 5s 21ms/step - loss: 0.6851 - accuracy: 0.5295 - val_loss: 0.6400 - val_accuracy: 0.7802
Epoch 2/25
127/127 [==============================] - 3s 20ms/step - loss: 0.6393 - accuracy: 0.6063 - val_loss: 0.5421 - val_accuracy: 0.8335
Epoch 3/25
127/127 [==============================] - 3s 20ms/step - loss: 0.5717 - accuracy: 0.6935 - val_loss: 0.4572 - val_accuracy: 0.8463
Epoch 4/25
127/127 [==============================] - 3s 21ms/step - loss: 0.5018 - accuracy: 0.7421 - val_loss: 0.3981 - val_accuracy: 0.8590
Epoch 5/25
127/127 [==============================] - 3s 20ms/step - loss: 0.4463 - accuracy: 0.7827 - val_loss: 0.3800 - val_accuracy: 0.8666
Epoch 6/25
127/127 [==============================] - 3s 23ms/step - loss: 0.4052 - accuracy: 0.8138 - val_loss: 0.3328 - val_accuracy: 0.8793
Epoch 7/25
127/127 [==============================] - 4s 30ms/step - loss: 0.3628 - accuracy: 0.8490 - val_loss: 0.3190 - val_accuracy: 0.8920
Epoch 8/25
127/127 [==============================] - 3s 20ms/step - loss: 0.3439 - accuracy: 0.8595 - val_loss: 0.2995 - val_accuracy: 0.8729
Epoch 9/25
127/127 [==============================] - 3s 20ms/step - loss: 0.3234 - accuracy: 0.8677 - val_loss: 0.2913 - val_accuracy: 0.8869
Epoch 10/25
127/127 [==============================] - 3s 20ms/step - loss: 0.3004 - accuracy: 0.8785 - val_loss: 0.2910 - val_accuracy: 0.8945
Epoch 11/25
127/127 [==============================] - 4s 34ms/step - loss: 0.2872 - accuracy: 0.8890 - val_loss: 0.2855 - val_accuracy: 0.8920
Epoch 12/25
127/127 [==============================] - 2s 19ms/step - loss: 0.2769 - accuracy: 0.8912 - val_loss: 0.2840 - val_accuracy: 0.8856
Epoch 13/25
127/127 [==============================] - 2s 19ms/step - loss: 0.2609 - accuracy: 0.9032 - val_loss: 0.2874 - val_accuracy: 0.8920
Epoch 14/25
127/127 [==============================] - 3s 20ms/step - loss: 0.2515 - accuracy: 0.9051 - val_loss: 0.2882 - val_accuracy: 0.8907
Epoch 14: early stopping
```

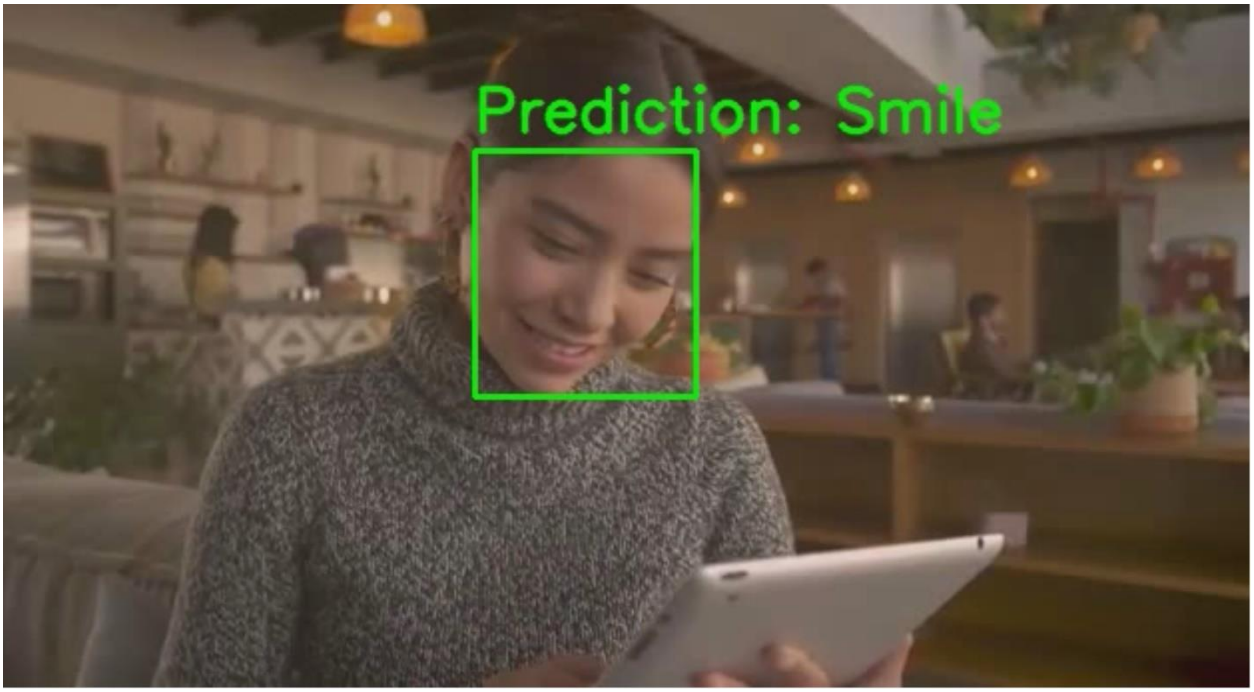
گزارش نهایی مدل و تست آن

در نهایت گزارش‌های نهایی مدل به شرح زیر است:



همچنین برای تست مدل، یک قطعه ویدئوی کوتاه بر طبق فریم جدا شد، هر فریم به مدل داده شد و نتیجه روی فریم نوشته شد. مدل هایی از خروجی در نهایت به شکل زیر است.

(ویدئو ورودی و نتیجه‌ی خروجی در فایل ارسالی موجود است)


Prediction: Smile

^