Задача

Необходимо написать классификатор картинок открытых (class=1)/закрытых (class=0) глаз, используя известную обучающую выборку. Решение будет проверяться на отложенной тестовой выборке (из того же распределения).

Особенности задачи — выборка не содержит меток классов открытых или закрытых глаз. Полная либо частичная разметка выборки, unsupervised кластеризация и т.д. - все это является частью задачи.

Решение должно быть реализовано на Python. Можно использовать любые библиотеки, но желательно не добавлять тяжеловесные зависимости без необходимости.

Текущий подход

Supervised classification

В тренировочном датасете 3600 изображений с глазами.

Простая сверточная нейросеть, 66 тыс параметров.

```
Grayscale image 24 x 24 > Conv2d(1, 24, 3) > LeakyReLU > Po
ol(2) >
                        > Conv2d(24, 48, 3) > LeakyReLU > P
ool(2) >
                        > Conv2d(48, 128, 3) > Dropout(0.1)
> Fc(2)
```

При обучении использованы данные MRL eye dataset: http://mrl.cs.vsb.cz/eyedataset (http://mrl.cs.vsb.cz/eyedataset). Использована случайная подвыборка из 20000 изображений. Аугментации - RandomFlip, CLAHE, ShiftScaleRotate. Изначальный размер MRL выборки (около 80 000 изображений) был уменьшен, чтобы выровнять распределения (в MRL глаза всего 37 человек), в тренировочном - намного больше.

Была вручную полностью размечена тренировочная выборка (заняло менее часа, сильно ускорено с помощью бейзлайнового простого решения). Разметка приложена в файле `assets.csv` . Бейзлайновое решение описано ниже.

Размечать данные **необязательно** для решения этой задачи. В некоторых случаях разметка может быть очень дорогой, или в принципе недоступной. Конкретно в этой задаче она очень удобна, т.к. позволяет насчитывать метрики, и проверять простейшие решения.

Золотое правило -- всегда сначала надо проверять самые тупые идеи - дают бейзлайн, позволяют анализировать данные и возникающие сложности.

Плюсы:

Очень быстро работает, можно запускать инференс на CPU и на мобилках. Локальная скорость около 800 картинок /сек. Простая модель, мало параметров,

легко тренится в условиях небольшого количества данных.

Минусы:

Нуждается в усилении (см. дальнейшие шаги).

Качество на валидации `acc=0.91, f1=0.91` . Это не очень много, можно сделать горадо лучше, но требуется больше времени.

Запуск решения

```
python3 inference.py /путь/до/папки
```

Решения за две строчки

Это бейзлайновые решения, они работают хуже, но их можно было получить практически бесплатно.

opencv + haar cascade

В opencv есть предобученные детекторы landmarks на признаках Хаара. Как бейзлайн можно взять его. Идея в том, что он не срабатывает на закрытые глаза (нет зрачков).

In [ ]:
```
import os
import random
from glob import glob

from tqdm import tqdm
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import cv2 as cv
```

In [ ]:
```
eye_cascade = cv.CascadeClassifier('haarcascade_eye_tree_eyeglasses
```

In [ ]:
```
def classify_haarcascade(img, scale=1.06):
    eyes = eye_cascade.detectMultiScale(img, scale, 0, minSize=(4,4
    return len(eyes) > 0
```

Такой классификатор работает очень быстро (более 4000 картинок/сек), и почти не требует мозга.

Метрики на полном train:

```
acc: 0.83694
pre: 0.81257
rec: 0.84046
f1 : 0.82628
```

Что значительно проигрывает сетям (часто ломается в каких-то сложных случаях).

supervised classification pretrained model-zoo

Есть примитивный нейросетевой классификатор, обученный как раз на MRL eye dataset: https://docs.openvinotoolkit.org/latest/omz_models_model_open_closed_eye_0001.html (https://docs.openvinotoolkit.org/latest/omz_models_model_open_closed_eye_0001.html)

Если вытащить ONNX граф и сконвертировать его в torch, то получится также простой классификатор, с нулем мозга.

Точность сравнимая с opencv, f1 и acc порядка `0.81` .

В этой модели гораздо меньше параметров, и она очень быстро работает. Есть квантизация.

Что еще было проверено

- различные архитектуры простейших сетей
- взятые первые слои предобученных сетей с дотренировкой (resnext, vgg), показали себя сравнимо с EyeNet, но при этом гораздо более вычислительно трудоемки
- попытки написать классификатор на HOG признаках, качество ниже, однако на tSNE видно, что классы можно хорошо разделить (см. `index.png`)

дальнейшие шаги

- В первую очередь, найти больше данных. Есть еще датасеты с глазами.

  Больше данных - можно увеличивать модель

  Более того, можно найти много отдельных классов - например, много открытых глаз можно найти в каком-нибудь Celeb Dataset и разных facial expression датасетах.
- Поработать со сложными случаями (очки, блики, темная кожа, низкий контраст)
- Больше подбора архитектуры
- Сейчас думаю слабые места еще с нормализацией (ее нет), не и

для резерча еще можно попробовать

- из классических методов EAR + landmarks по типу https://iopscience.iop.org/article/10.1088/1742-6596/1529/5/052015/pdf (https://iopscience.iop.org/article/10.1088/1742-6596/1529/5/052015/pdf)

Проблемы - низкое разрешение

- На случай, если разметка недоступна, или слишком дорога, можно посмотреть в сторону distance learning, простейший способ уменьшить количество разметки - вытащить вектор картинки любым предобученным backbone, разметить

несколько сотен глаз, затем вссе закинуть в kNN, посмотреть, что получится
- Также в сторону few shot classification, 1-shot
- Можно попробовать обойтись вообще без разметки, подбор feature extract -> кластеризация
- single class svm (хотя по опыту он почти никогда не работает)
- автоэнкодеры, VAE
- эмбеддинги по типу Arcface, self-supervised векторизация с последующим разделением
- ... еще очень много идей, к сожалению, у меня уже закончилось время

In [ ]: