Абаполов Филипп

Отчет по выполнению тестового задания в VisionLabs

20 Ноября 2020 г.

Задание

Необходимо натренировать классификатор открытых/закрытых глаз, используя заданную обучающую выборку. Решение будет проверяться на тестовой выборке, которая тебе недоступна. Обе выборки были взяты из одной базы изображений. Решение должно быть реализовано в python в виде функции-обертки для классификатора следующего вида is_open = openEyeCheck(inpIm), где inpIm - полный путь к изображению глаза, is_open - результат классификации (1 - открыт, 0 - закрыт). Обучающая выборка доступна по ссылке:
https://drive.google.com/file/d/122BgFHJG8Kgn1E_bkT1Lu8I1Cf10glVn/view?usp=sharing

Необходимо прислать эту функцию со списком необходимых для запуска решения библиотек. Рядом с основной функцией могут лежать вспомогательные файлы (например, веса сети). Помимо функции решения ждем от тебя короткий отчет (2-3стр.) с описанием задачи, мотивацией и кратким изложением выбранного метода, промежуточными результатами (например анализ метода для разных значений параметров, тренировочные / валидационные ошибки, ...), результатами на валидационной выборке в виде EER (Equal error rate) и сложных/простых примеров картинок, выводами и ссылками на литературу.

Итоговое решение о принятии на работу будем принимать по совокупности характеристик твоего резюме, нашего интервью, решения тестовой задачи (в частности, результата на тестовой выборке) и отчета.

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 и остается на твоё усмотрение.

План ресерча

- Изучение решений self-supervised learning
- Изучение решений self-supervised learning state-of-the-art
- Полная разметка данных и supervised learning

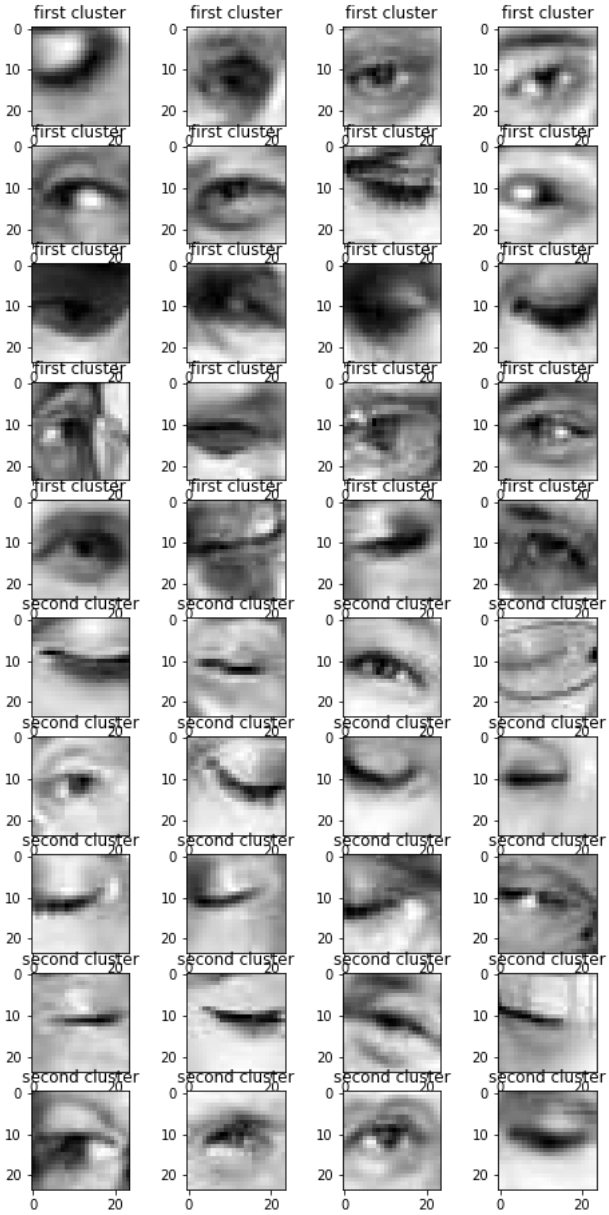
Изучение решений self-supervised learning

В данном разделе self-supervised я называю фактически semi-supervised, так как даже если и получится разделять на два кластера, но нужно будет разметить какой кластер относится к какому классу.

Рабочий ноутбук этого раздела: clustering.ipynb Основной идеей этого подхода является кластеризация изображений.

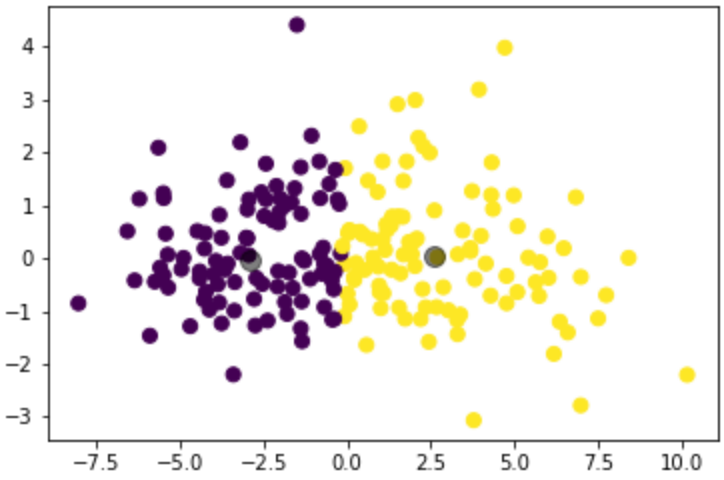
В начале мы размечаем 5% изображений для подсчета точности моделей.

1. K-means изображений (1, 24*24)


first cluster
second cluster

Результат: метод k-средних разделяет выборку с точностью 60% (f1=0.57, eer = 0.62)на маленькой тестовой выборке, так что на кросс валидации результат может измениться.

2. Гипотеза заключается в том, что понижение размерности методом главных компонент повысит качество к-средних.



Центры кластеров к-средних после понижения размерности до n_components=2 отмечены серым цветом. Точность на тестовой выборке составляет 60%(f1=0.57, eer = 0.62).

Гипотеза неверна, точность не изменилась.

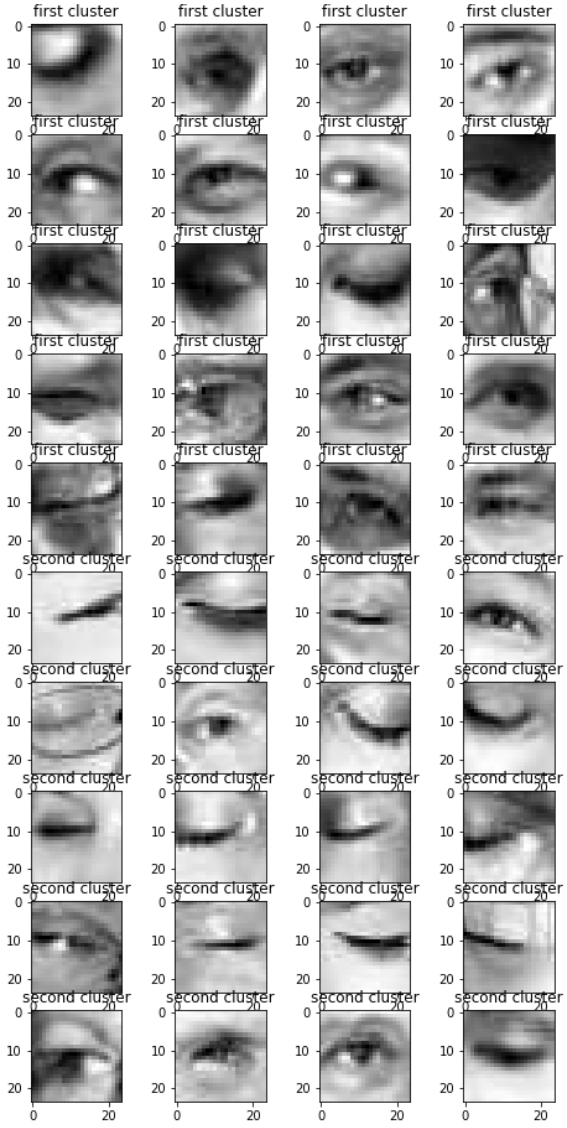
3. Глубокий подход

Линейный вариационный автоэнкодер:

Вариационный автоэнкодер переводит входные тензоры в latent-space размерностью (1, 100), причем вектора в скрытом пространстве имеют нулевое среднее и стандартное отклонение 1.

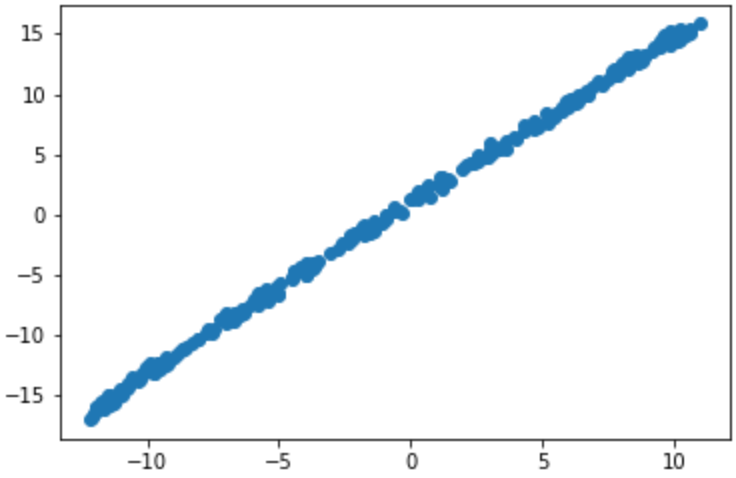
В данном решении ставится гипотеза 1, что метод к-средних может хорошо разделить вектора в скрытом пространстве.

Результат:

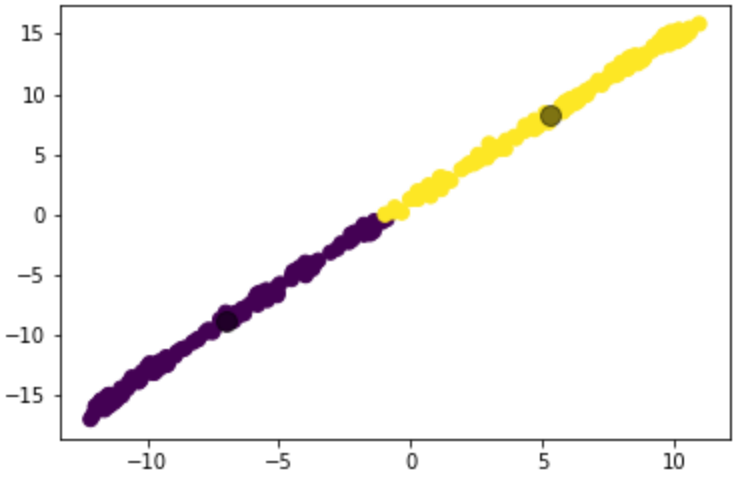


Точность VAE составляет 61% (f1 = 0.62, eer = 0.62)

Визуализация результата линейного автоэнкодера с помощью метода TSNE:



Хотя такой подход и не принят, но мы попробуем разделить результат tsne на кластеры:



Точность составляет все те же 61% (f1 = 0.62, eer = 0.62) . Нас такой результат не устраивает.

Гипотеза 2: линейный автоэнкодер не может извлекать визуальные фичи из изображений, и возможно слои свертки повысят качество.

Сверточный вариационный автоэнкодер:

Слои энкодера:

```
self.begin_encoder = nn.Sequential(
    nn.Conv2d(1, 32, kernel_size=4, stride=2), # 32x11x11
    nn.BatchNorm2d(32),
    nn.LeakyReLU(),
    nn.Conv2d(32, 64, kernel_size=4, padding=1, stride=2), # 64x5x5
    nn.BatchNorm2d(64),
    nn.LeakyReLU(),
    nn.Conv2d(64, 128, kernel_size=3, stride=2), #128x2x2
    nn.BatchNorm2d(128),
    nn.LeakyReLU(),
    nn.Conv2d(128, 256, kernel_size=2, stride=1), #256x1x1
    nn.BatchNorm2d(256),
    nn.LeakyReLU(),
    nn.Flatten()
)
```

Идея батч-нормализации взята из статьи по semi-supervised learning.

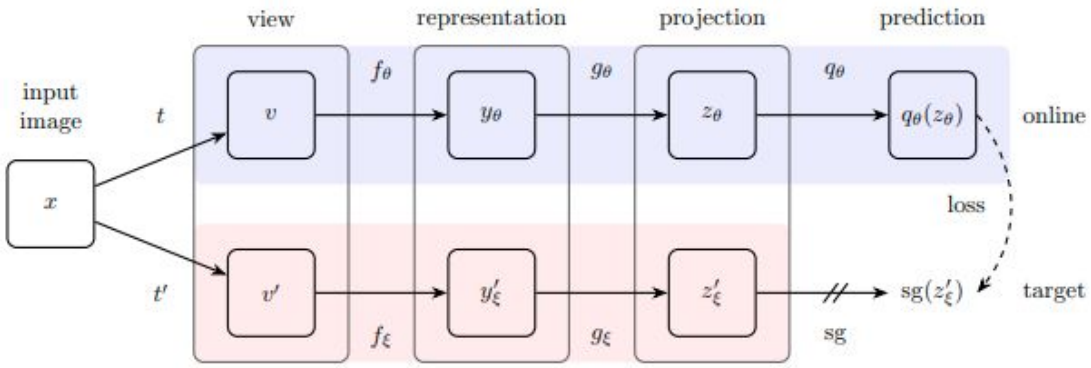
Точность сверточного автоэнкодера составляет 54%(f1 = 0.56, eer = 0.54), что можно считать случайным классификатором с точностью 50%.

Гипотеза 2 неверна.

Кластеризация векторов из нормального распределения не дала хороших результатов, гипотеза 1 неверна.

Изучение решений self-supervised learning state-of-the-art

В данном разделе я изучил SOTA архитектуры: Understanding self-supervised and contrastive learning with "Bootstrap Your Own Latent" (https://untitled-ai.github.io/understanding-self-supervised-contrastive-learning.html)
Такие архитектуры, как MoCo, BYOL используют фактически contrastive learning, суть которого заключается в сближении аугментаций одного изображения и отдаления от других объектов выборки. Архитектура BYOL:



Изучение и имплементация данных решений трудозатратно. Так как время было ограничено, было принято решение заняться другим подходом

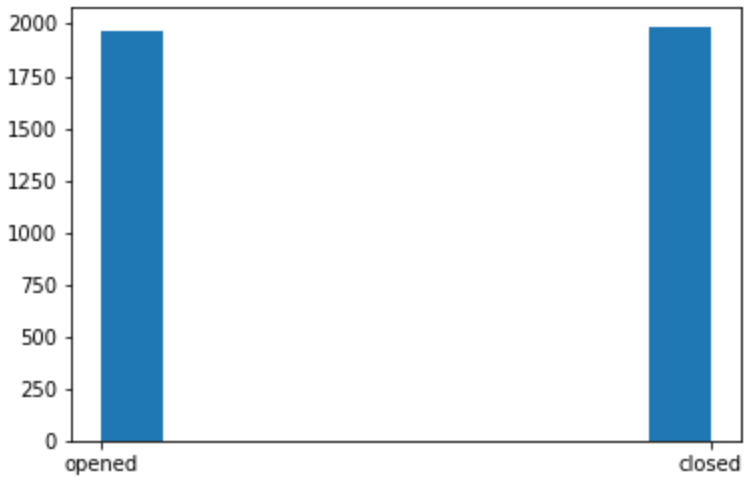
Полная разметка данных и supervised learning

Так как количество данных невелико(4000 изображений), основная идея заключается в том, что с помощью автоматизированного лейблинга и классической классификации изображений можно получить высокие результаты, так как backbone архитектур огромное количество.

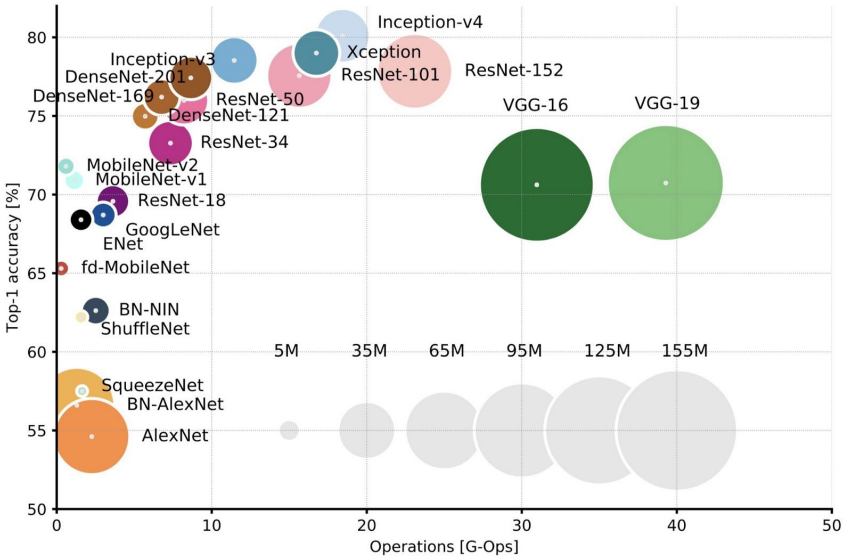
С помощью библиотеки pigeon разметка данных заняла всего 1-1.5 часа, что немного.

Размечено 3955 изображений из 4000, тк некоторые невозможно определить ни в одну ни в другую(например черные изображения).

Выборка получилась очень уравновешенная, что скорее всего говорит о хорошем качестве лейблинга.



Выбор mobilenet-v2 оправдан, модель имеет минимальное количество параметров, операций в отношении с точностью.

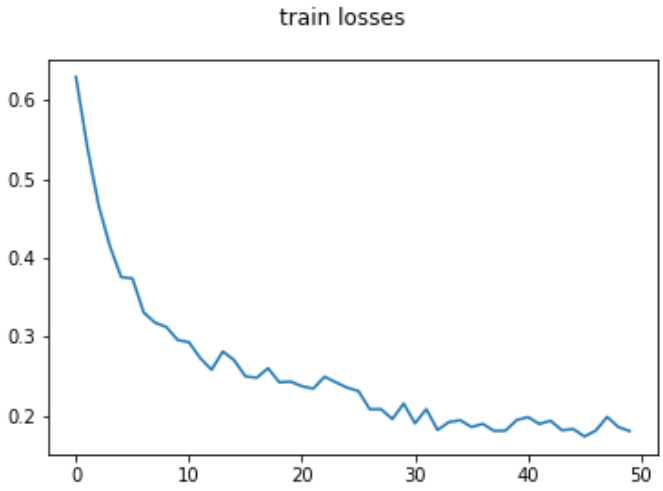


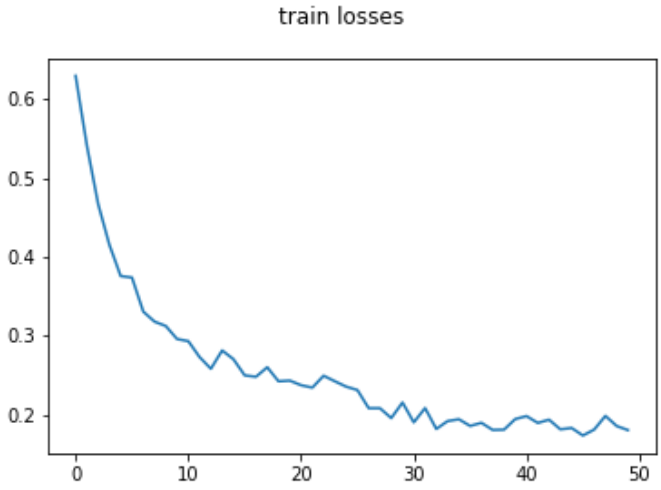
Результат на 50 эпохах:

acc: 0.930288

f1 : 0.931135

eer : 0.080473





ВЫВОДЫ

Существует множество sota архитектур для self-supervised learning’a, сложных в реализации, но показывающих хорошие результаты на экспериментах. В случае небольшого количества данных, как у нас, можно разметить их и использовать классические методы классификации.

Список литературы

Intuitively Understanding Variational Autoencoders
https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

Bootstrap Your Own Latent A New Approach to Self-Supervised Learning
https://arxiv.org/pdf/2006.07733.pdf

Understanding self-supervised and contrastive learning with "Bootstrap Your Own Latent" (BYOL)
https://untitled-ai.github.io/understanding-self-supervised-contrastive-learning.html