Бинарная классификация на открытые/закрытые глаза

Успенская Наталья

June 2021

1 Этапы работы

1. Поиск open source датасетов и текущих решений (статей/кода) для данной задачи
2. Разметка 10% от присланной обучающей выборки = 400 изображений, в дальнешем будет упоминаться как vl_labeled_test_dataset
3. Unsupervised
4. Semi-Supervised
5. Supervised

2 Найденные датасеты

- `https://www.kaggle.com/shehzadhanif/eyes-open-closed-dataset` датасет размера 47.2 тыс чб 83×83 изображений, разметка - открыт/закрыт, нашла много ошибок (причем для "сложных"примеров прикрытых глаз), изначально собиралась его использовать для обучения, но из-за множественных ошибок отказалась
- `http://parnec.nuaa.edu.cn/_upload/tpl/02/db/731/template731/pages/xtan/ClosedEyeDatabases.html` датасет размера 4846 чб 24×24 изображений, разметка - открыт/закрыт, далее open_source_dataset. Такого же размера изображения, как и в неразмеченном датасете (похожее распределение данных на присланную неразмеченную выборку)

3 Кратко статьи/решения данной задачи

В основном DL методы для решения данной задачи используют отношение между шириной и высотой глаза, долю открытого зрачка и т.п, которые определяются по facial landmarks. Но детекция особых точек применяется к изображению всего лица, к вырезанному глазу не нашла обученного решения (хотя можно найти часть большой сетки, которая смотрит на глаза, читала в issues dlib), можно еще было взять датасет глаз с сегментацией зрачка, белка и тд, а по ним определять точки, но времени проверить это было немного, решила начать с очевидных решений.

4 Unsupervised

4.1 kmeans

быстрый baseline KMeans, зафиченный на оставшихся 90% от присланной обучающей выборки, показал на vl_labeled_test_dataset EER слишком большой ≈ 0.3.

4.2 deepcluster

Нашла в топах на unsupervised image classification `https://paperswithcode.com/task/unsupervised-image-classification` эту модель `https://github.com/facebookresearch/deepcluster`. Архитектура представлена на Рис. 4.2, Convnet экстрактит фичи, на которых потом применяется кластеризация (дефолт KMeans).

Обучала на том же разбиении данных, что и в 4.1 (backbone=alexnet). В предсказаниях наблюдался всегда сильный перекос в сторону какого-то одного лейбла В issues к гитхабу писали, что датасет порядка от 1 до нескольких тысяч слишком маленький для этой модели и авторы не уверены в ее качестве на малых данных (у нас 4к).

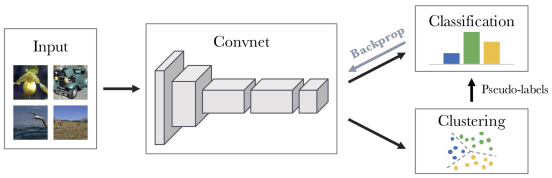

Рис. 1: deepcluster architecture

5 Semi-Supervised

Поделила open_source_dataset на train/val в пропорции 0.9 : 0.1, для тестовой выборки использовала размеченный мной vl_labeled_test_dataset.

Использовала ResNet18, предобученную на ImageNet, поменяв последний полносвязный слой на полносвязный слой с выходом размерности 2 (кол-во классов), и дообучала на нормализованном датасете, вычислив mean и std, с разными аугментациями, параметрами, лоссами, оптимизаторами, с lr_scheduler.

Лучшие результаты на 14 эпохе: val accuracy = 0.9742, val EER = 0.0243, val accuracy = 0.9739, val EER = 0.0281.

с аугментацией:

```
train_transform = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(256),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.4973, 0.4973, 0.4973],
                         std=[0.1962, 0.1962, 0.1962]),
    ])
```

с параметрами обучения:

```
batch_size = 128
criterion = torch.nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(),
                      lr=0.001,
                      momentum=0.9,
                      weight_decay=0.001)
```

Идея улучшить результат таким образом:

- обучить ResNet18 на open_source_dataset (способ выше)
- разметить этой моделью неразмеченную обучающую выборку без части, размеченной мной для тестовой выборки
- поправить при необходимости ошибки в разметке
- объединить open_source_dataset и target_dataset
- обучить новую модель на расширенном датасете (open_source_dataset + target_dataset поделила на train и val, vl_labeled_test_dataset - test).

После присваивания лейблов неразмеченной выборке и объединения датасетов, стала проверять на дубликаты 6 (очень похожие выборки).

6 Проверка на дубликаты объединенного дадасета и создание финального датасета

https://www.pyimagesearch.com/2020/04/20/detect-and-remove-duplicate-images-from-a-dataset-for-deep-learning/

Неразмеченная выборка для обучения полностью содержится в open_source_dataset, а 846 изображений – похоже неизвестная мне тестовая выборка :). Поэтому делю 4000 изображений из open_source_dataset, совпадающих с неразмеченная выборкой, на train/val в пропорции 0.9 : 0.1, в качестве test беру эти 846 изображений и никак не использую для обучения, далее приведу метрики на val и test для разных моделей 7.

```
3687652782579442175: ['labeled_extended/open/Phil_Mickelson_0001_R.jpg',
 'labeled_extended/open/000404.jpg'],
3704329831505459288: ['labeled_extended/open/Robert_Downey_Jr_0001_R.jpg',
 'labeled_extended/open/001241.jpg'],
17554145942999309564: ['labeled_extended/open/003573.jpg',
 'labeled_extended/open/Rick_Carlisle_0001_L.jpg'],
9279282971668432120: ['labeled_extended/open/000857.jpg',
 'labeled_extended/open/Asif_Hanif_0001_R.jpg'],
7160939684412014320: ['labeled_extended/open/Asmaa_Assad_0001_L.jpg'],
2205987915398365667: ['labeled_extended/open/Sue_Johnston_0001_R.jpg',
 'labeled_extended/open/002232.jpg'],
7208051277530671104: ['labeled_extended/open/Anders_Ebbeson_0001_L.jpg',
 'labeled_extended/open/001248.jpg'],
6502045679449349692: ['labeled_extended/open/Ann_Veneman_0001_L.jpg',
 'labeled_extended/open/002271.jpg'],
1947528605757478944: ['labeled_extended/open/002467.jpg',
 'labeled_extended/open/Zinedine_Zidane_0001_L.jpg'],
9458551878104849120: ['labeled_extended/open/003152.jpg'

[26] print(f'Number of unique imgs = {len(hashes)}')

     Number of unique imgs = 4846
```

Рис. 2: Уникальных изображений 4846/8846

Для чистоты эксперимента выложу код обучения для воспроизводимости результатов без использования тестовой выборки и финальное разбиение датасета на train/val/test. `https://drive.google.com/file/d/11g_p8c0XT_Yn-0xUcqGWlX_FM8KksouE/view?usp=sharing` – ссылка на итоговый датасет. Классы в каждой папке сбалансированы. Примеры изображений представлены на Рис. 6

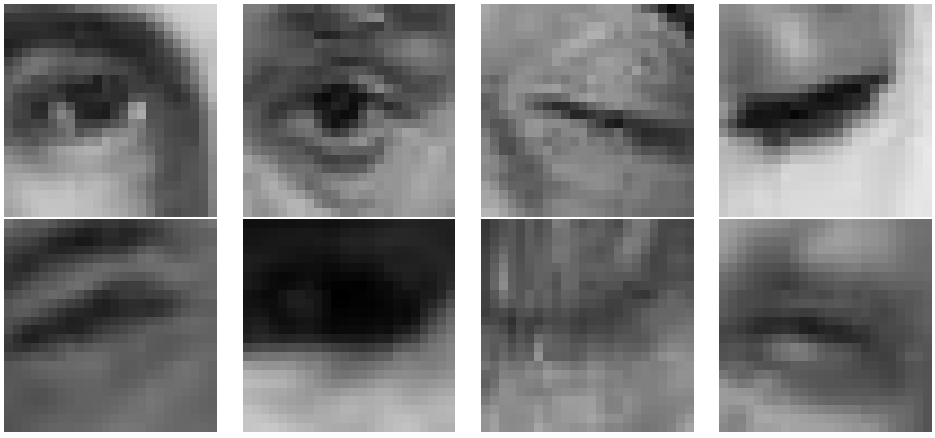

Рис. 3: Вверху простые примеры, внизу – сложные. Два столбца слева соответствуют открытом глазу, два справа – закрытому.

7 Supervised

Датасет, описанный в предыдущей части 6 используется для обучения 5 архитектур, представленных в Таб. 7. В Таб. 7 приведены метрики на лучших весах после подбора различных гиперпараметров обучения, аугментаций.

metrics	resnet18	resnet34	mobilenet_v3_large	shufflenet_v2_x1_0	**wide_resnet50_2**
val accuracy	0.9726	0.9701	0.9601	0.9800	**0.9850**
test accuracy	0.9752	0.9728	0.9704	0.9728	**0.9835**
val EER	0.0196	0.0203	0.0380	0.0152	**0.0152**
test EER	0.0212	0.0212	0.0284	0.0236	**0.0154**

Таблица 1: Метрики для различных архитектур

8 Финальное решение

Из Таб. 7 видно, что лучшие результаты у архитектуры WideResnet-50-2. В финальном решении используется данная архитектура с весами, на которых получены результаты в Таб. 7 на 19 эпохе.

Аугментация:

```
train_transform = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(256),
    transforms.RandomHorizontalFlip(p=0.2),
    transforms.RandomRotation(degrees=(-45, 45)),
    transforms.RandomPerspective(distortion_scale=0.7, p=1, interpolation=2, fill=0),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.4976, 0.4976, 0.4976],
                         std=[0.1970, 0.1970, 0.1970]),
    ])
```

Оптимизация:

```
criterion = torch.nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(),
                      lr=0.0008,
                      momentum=0.9,
                      nesterov=True,
                      weight_decay=0.002)
```