Описание результатов работы

0.1 Анализ данных

В качестве данных для обучения (+валидации) и проверки работоспособности модели были взяты архивы из LibriTTS: train-clean-100.zip и test-clean.zip соответственно. Архивы содержат в себе директории с названиями-числами, которые соответствуют id спикеров. В каждой такой папке лежат аудиозаписи wav, в которых спикер проговаривает предложение, а также их транскрипции txt. Train-clean-100 содержит в себе 251 директорий, а test-clean - 40. Анализ данных показал что в обучающей выборке train-clean-100 всего 125 спикеров женского пола и 126 - мужского, а в тестовой выборке - по 20 представителей каждого пола. При этом количество аудиозаписей с женским голосом для обучающей выборки = 17635, а с мужским = 15611 (в тестовой: 2930 женских и 1907 мужских). Для тренировочных данных это свидетельствует о сбалансированности выборок, но для тестовых - это не совсем так.

Информацию о полах спикеров по их id можно узнать из файла libritts_speakerinfo.txt. Извлечение массива с путями до аудиофайла для каждого спикера, а также получение массива меток было произведено с помощью скрипта src/data/prepare_data.py.

Для обучения модели я решила извлечь признаки из мелспектрограмм аудиозаписей, взяв их как среднее по первой оси транспонированного выхода функции librosa.feature.melspectrogram. Преобразовав таким образом тензор к вектору размера 128, мы потеряем информацию о частотно-временных зависимостях записи, однако и из таких признаков получится извлечь достаточно данных для решения задачи бинарной классификации. Извлечение признаков более сложного вида заняло бы гораздо больше времени.

0.2 Выбор модели

Я решила построить классификатор на основе сверточной нейросети - её структура должна помочь сохранить зависимость/последовательность 128ми мел-бендов, т.е. принимая на вход признак она будет обучаться, понимая созависимую, непрерывную природу его составляющих бинов. В структуре нейросети, помимо сверточных слоев, я решила использовать слои с пулингом (для понижения размерности и более быстрого обучения сети), а также активации ReLU и линейные слои с активацией сигмоидой в конце.

```
Model(
(blocks): Sequential(
(0): Conv1d(1, 32, kernel_size=(3,), stride=(2,), padding=(1,), bias=False)
(1): BatchNorm1d(32, eps=1e-05, momentum=0.1, affine=False, track_running_stats=True)
(2): ReLU(inplace=True)
(3): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
(4): Conv1d(32, 64, kernel_size=(3,), stride=(2,), padding=(1,), bias=False)
(5): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=False, track_running_stats=True)
(6): ReLU(inplace=True)
(7): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
(8): Conv1d(64, 128, kernel_size=(3,), stride=(1,), padding=(1,), bias=False)
(9): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
(10): Flatten(start_dim=1, end_dim=-1)
(11): Linear(in_features=512, out_features=32, bias=True)
(12): Linear(in_features=32, out_features=1, bias=True)
(13): Sigmoid()
))
```

Можно было бы построить классификатор с помощью случайного леса, пробуя улучшать его варьированием гиперпараметров (grid search, random search), а также алгоритмами бустинга (адаптивный,

градиентный). Несмотря на неумение работать с последовательностями данных, я думаю что случайный лес (на грамотно подобранном наборе признаков и параметров) справился бы с задачей не хуже, чем получившаяся нейронная сеть.

0.3 Описание экспериментов

Для подбора оптимальных гиперпараметров я решила воспользоваться фреймворком ray tune. Алгоритм будет случайным образом варьировать набор параметров { batch_size, lr, epoches } - размер батча, скорость обучения и количество эпох. Было запущено 20 испытаний.

```
== Status ==
Current time: 2022-02-15 17:05:50 (running for 00:37:59.98)
Memory usage on this node: 4.2/12.7 GiB
Using AsyncHyperBand: num_stopped=17
Bracket: Iter 16.000: -0.13493975903614458 | Iter 8.000: -0.15042117930204574 | Iter 4.000: -0.17349397590361446 |
      Iter 2.000: -0.19822213039885173 | Iter 1.000: -0.3971119133574007
Resources requested: 0/2 CPUs, 0/0 GPUs, 0.0/6.74 GiB heap, 0.0/3.37 GiB objects
Result logdir: /root/ray_results/DEFAULT_2022-02-15_16-27-50
Number of trials: 20/20 (20 TERMINATED)

|Trial name| status    | loc               |      lr      |batch_size|epoch| loss     |  accuracy  | training_iteration
| 00000  | TERMINATED | 172.28.0.2:9552  | 0.00168294  |    12 |  20 | 0.166065 |  0.936241 |                  20 |
| 00001  | TERMINATED | 172.28.0.2:9551  | 1.23473e-05 |    16 |  10 | 0.428916 |  0.809323 |                  10 |
| 00002  | TERMINATED | 172.28.0.2:9663  | 0.00013657  |    16 |  20 | 0.180723 |  0.929925 |                  16 |
| 00003  | TERMINATED | 172.28.0.2:9757  | 0.00399305  |     8 |  20 | 0.146811 |  0.942857 |                  20 |
| 00004  | TERMINATED | 172.28.0.2:9803  | 0.00392563  |    16 |  10 | 0.154217 |  0.939699 |                  10 |
| 00005  | TERMINATED | 172.28.0.2:9881  | 9.5095e-05  |     8 |  10 | 0.370638 |  0.836842 |                   1 |
| 00006  | TERMINATED | 172.28.0.2:9919  | 1.5825e-05  |     6 |  20 | 0.601986 |  0.721955 |                   1 |
| 00007  | TERMINATED | 172.28.0.2:9958  | 1.8633e-05  |     4 |  20 | 0.504813 |  0.778947 |                   1 |
| 00008  | TERMINATED | 172.28.0.2:10002 | 0.000791431 |    16 |  20 | 0.13253  |  0.945113 |                  20 |
| 00009  | TERMINATED | 172.28.0.2:10074 | 7.34712e-05 |     8 |  10 | 0.389892 |  0.824211 |                   1 |
| 00010  | TERMINATED | 172.28.0.2:10114 | 0.00157494  |    16 |  10 | 0.236145 |  0.902406 |                   2 |
| 00011  | TERMINATED | 172.28.0.2:10154 | 1.20615e-05 |     4 |  10 | 0.608303 |  0.723459 |                   1 |
| 00012  | TERMINATED | 172.28.0.2:10198 | 0.00621972  |     6 |  20 | 0.133574 |  0.948421 |                  20 |
| 00013  | TERMINATED | 172.28.0.2:10246 | 2.23019e-05 |     4 |  10 | 0.512034 |  0.750526 |                   1 |
| 00014  | TERMINATED | 172.28.0.2:10290 | 3.67047e-05 |     4 |  20 | 0.427798 |  0.801654 |                   1 |
| 00015  | TERMINATED | 172.28.0.2:10334 | 0.000469409 |     4 |  10 | 0.185319 |  0.924511 |                   8 |
| 00016  | TERMINATED | 172.28.0.2:10457 | 5.52389e-05 |    16 |  10 | 0.568675 |  0.744361 |                   1 |
| 00017  | TERMINATED | 172.28.0.2:10491 | 0.00104873  |     4 |  10 | 0.148014 |  0.940602 |                  10 |
| 00018  | TERMINATED | 172.28.0.2:10543 | 9.61764e-05 |    12 |  20 | 0.435018 |  0.806767 |                   1 |
| 00019  | TERMINATED | 172.28.0.2:10581 | 0.000116182 |    12 |  10 | 0.404332 |  0.824361 |                   1 |


(func pid=10491) Finished Training
2022-02-15 17:05:50,508 INFO tune.py:636 -- Total run time: 2280.63 seconds (2279.96 seconds for the tuning loop)
      .
Best trial config: {'lr': 0.0007914312635613735, 'batch_size': 16, 'epochs': 20}
Best trial final validation loss: 0.13253012048192772
Best trial final validation accuracy: 0.9451127648353577
```

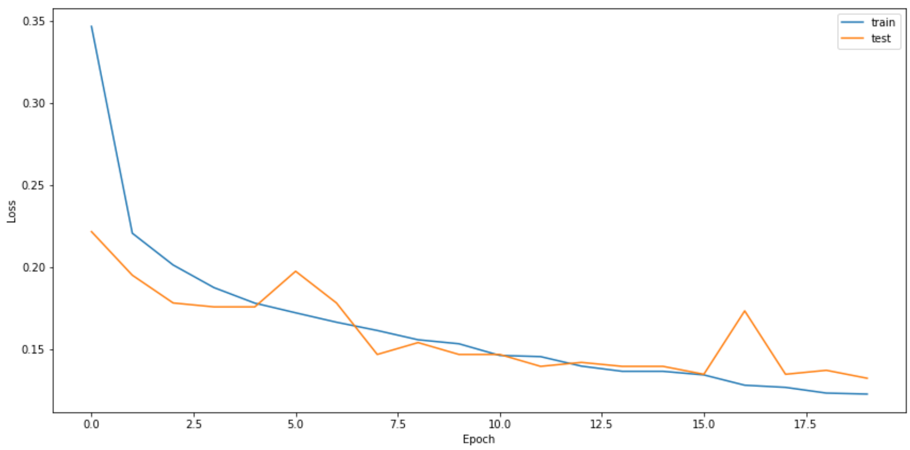
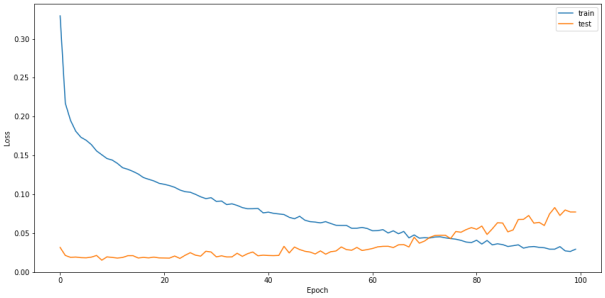


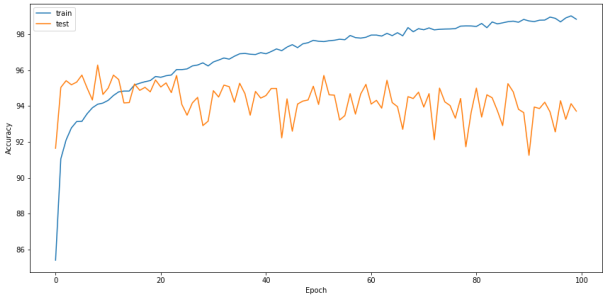
Рис. 1: Значение бинарной кросс-энтропии оптимальной модели в зависимости от эпохи обучения

По результатам экпериментов была получена и сохранена оптимальная конфигурация гиперпараметров: {'lr': 0.0007914312635613735, 'batch_size': 16, 'epoches': 20}. Так как изначально из соображений

быстродействия количество эпох бралось небольшим, я решила увеличить его до 100 и посмотреть как себя ведут на протяжении обучения бинарная кросс-энтропия и точность (accuracy) модели:



Значение функции потерь (BCE) оптимальной модели в зависимости от эпохи обучения



Точность (accuracy) модели в зависимости от эпохи обучения

Рис. 2: Модель с оптимальными lr и batch_size на 100 эпохах обучения

Видим, что уже с самого начала обучения модель показывает себя хорошо на тестовой выборке, но начиная с какого-то момента, лосс начинает расти - модель начинает переобучаться на тренировочных данных. Таким образом, имеет смысл обучать модель не более сорока эпох.

0.4 Выводы

Итак, была построена сверточная нейросетевая модель для классификации мужских и женских голосов. Классификатор показывает неплохой результат (около 94% accuracy) на тестовой выборке, однако ему есть куда расти и как улучшаться:

1. При извлечении признаков из-за усреднения мелспектрограммы для аудиозаписей было потеряно много данных о связи частот с громкостью и временем, которые могли бы быть полезны при дальнейшем анализе нейросети: возможно стоит подумать о другом наборе признаков, с большим размером/размерностью.

2. Саму структуру модели можно модифицировать: текущая последовательность слоев может быть неоптимальной, стоит попробовать поизменять количество каналов/размерность скрытых линейных слоев. Вдогонку к предыдущему пункту: если увеличивать размерность признаков, то нужно как учитывать связи по нескольким направлениям сразу - например, брать не одномерные Conv1d слои, а Conv2d.

3. При проведении экспериментов по оптимизации модели стоит проводить большее количество испытаний, а также возможно ввести больше гиперпараметров (в т.ч. связанных со структурой самой CNN) и увеличить возможные границы варьирования существующих.

4. Наконец, можно построить классификатор не с помощью CNN: попробовать логистическую регрессию, MLP, случайный лес и сравнить результаты между собой.