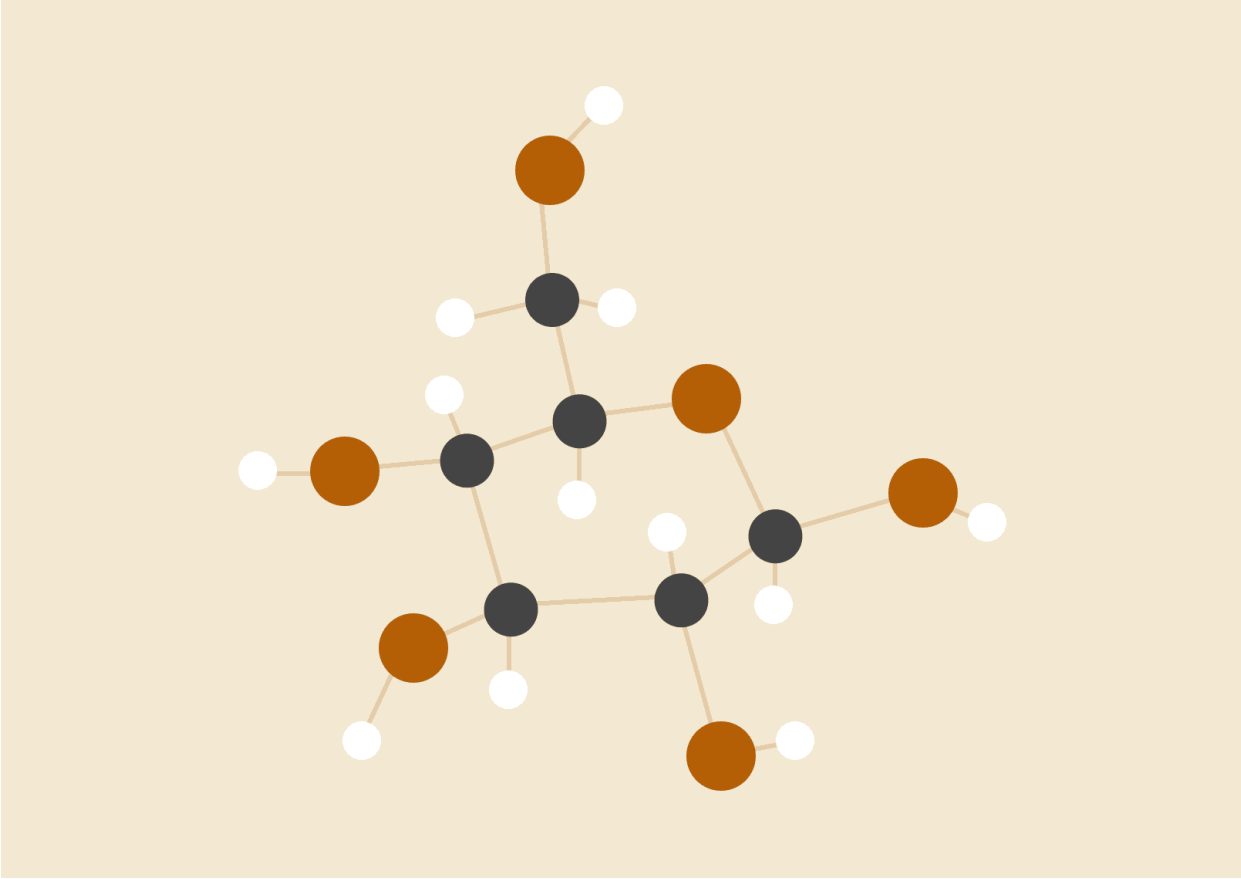
# Глубинное обучение в обработке звука, ДЗ-2, Keyword spotting

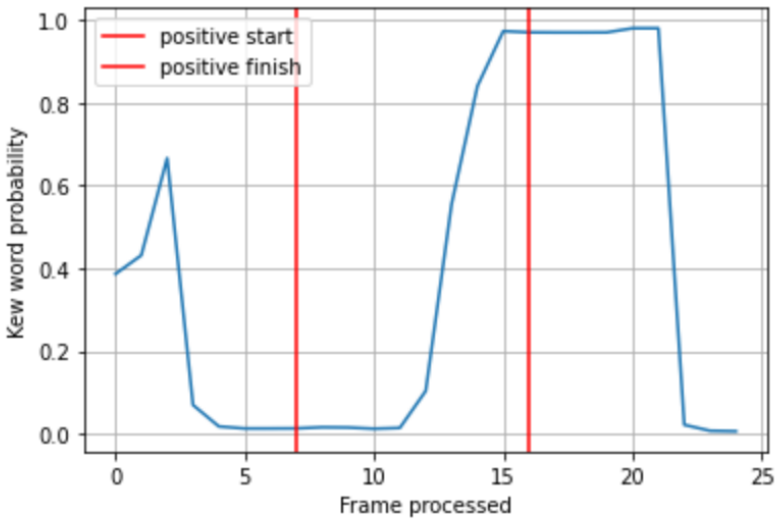
*Петрович Сергей, группа БПМИ181*



21.11.2021

## Part 1. Streaming

Для начала была реализована стриминговая версия модели для KWS. У нее есть обычный и стриминговый режим: в обычном режиме она работает в точности так же, как и бейзлайн-версия, а в стриминговом режиме, соответственно, возможность последовательной обработки входного потока фреймов. В условии требовалось поддерживать буфер для входных фреймов и для выходов GRU. На вход стриминговому режиму передается какой-то кусок входной дорожки, далее предсказание делается по новому куску и сохраненному на данный момент буферу. После этого буфер обновляется. Для того, чтобы изначально заполнить буфер, делается initial вызов (для того, чтобы первый кусок был хотя бы размера свертки). Для демонстрации работы я взял два отрицательных примера из случайного батча и один положительный, а затем склеил их таким образом, чтобы положительный пример оказался в центре. Ожидаем, что модель будет выдавать вероятности около 1 где-то в середине дорожки, а в остальных местах выходная вероятность будет существенно ниже.




На картинке выше представлен график выдаваемой вероятности в эксперименте. В целом, картинка соотносится с ожиданиями. Больше всего смущает только плато высоких вероятностей, продолжающихся до середины отрицательного примера. Такое поведение можно объяснить тем, что на этом промежутке в буфере еще находятся скрытые состояния, содержащие положительные таймстемпы, поэтому модель продолжает на них срабатывать, пока буфер полностью от них не очистится.

## Part 2. Speedup and compression

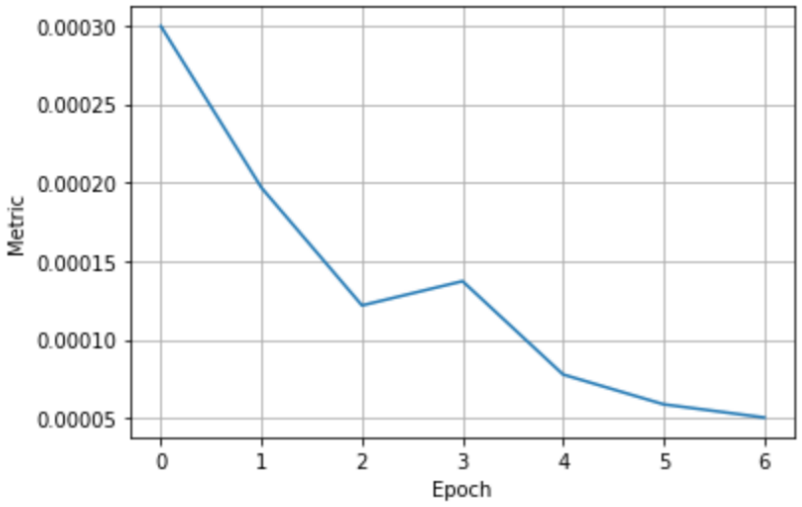
Для оценки размера модели я реализовал функцию, которая считает размер файл с моделью, а не как функция в условии. Более того, все задание я выполнял на CPU для лучшей воспроизводимости результатов и большей технической свободы (вроде бы квантизация в Pytorch пока не очень ладит с GPU). Так как модель довольно маленькая, то даже на центральном процессоре все считается довольно быстро (по крайней мере у меня локально на ноутбуке).

Вычислительную эффективность я считал на одном батче из валидационного сплита. Для базовой модели получились такие значения метрик при качестве около 2.5e-5:

1. Размер модели из условия: 0.2687187194824219 Mb
2. Размер файла с моделью: 0.27574634552001953 Mb
3. MACs значение для одного батча: 119527424.0

**Дистилляция 1**

Сжимать и ускорять модель я начал с дистилляции модели, так как таким образом можно хорошо улучшить метрики как по размеру модели, так и по скорости работы. Самое простое и логичное, что я попробовал - это дистиллировать ту же самую модель, но со всеми скрытыми размерностями в 2 раза меньше бейзлайн-модели:




```
END OF EPOCH 6: au_fa_fr = 5.0300458561275e-05
```

Дистиллировать модель до нужного качества получилось всего лишь за 7 эпох. При дистилляции использовалась температура 1. и начальным learning rate 1e-3. Уверен, что если подкрутить гиперпараметры обучения, то можно добиться еще более быстро сходимости, но хватило и этого.

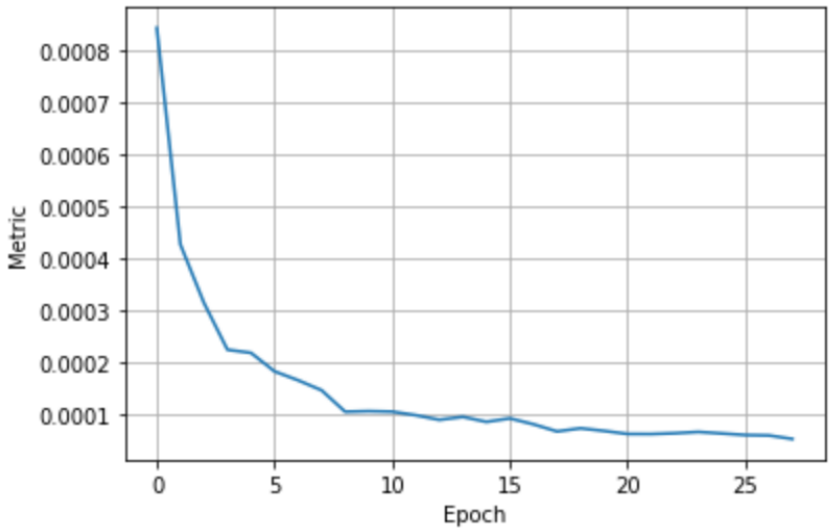
При этом получились следующие метрики сжатия и эффективности:

1. Размер модели из условия: 0.06893539428710938 Mb
2. Размер файла с моделью: 0.07616138458251953 Mb
3. MACs значение: 35613696.0
4. Compression rate по размеру модели из условия: 3.898124066183388
5. Compression rate по размеру файла с моделью: **3.620553211204468**
6. Speedup rate: **3.3562207078984443**

После того, как все легко получилось с такой дистилляцией, я решил, что можно дистиллировать еще сильнее.

**Дистилляция 2**

На этот раз я заменил все скрытые размерности на 16 (вместо базовых 64), а также в 2 раза уменьшил число выходных каналов свертки (с 8 на 4). В результате процесс оптимизации занял больше эпох, но все равно получилось добиться требуемого качества (также использовалась температура 10):




```
END OF EPOCH 27: au_fa_fr = 5.373776596800111e-05
```

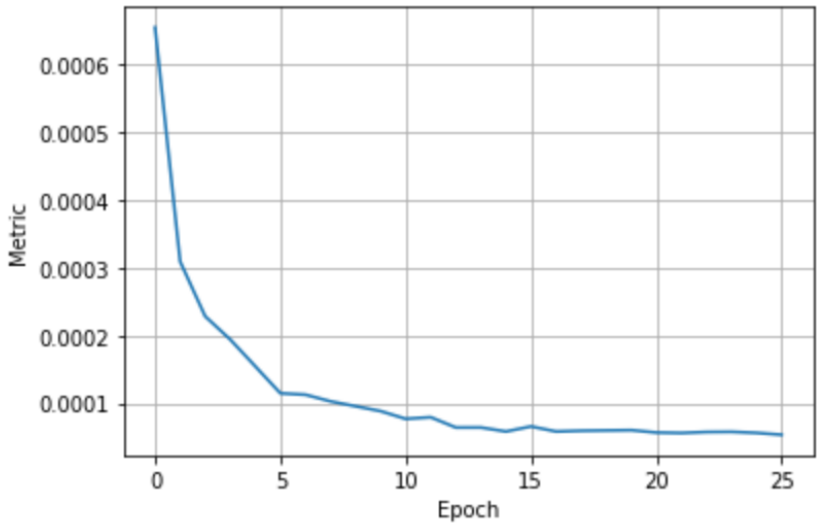
После такого метрики сжатия и ускорения стали выглядеть совсем хорошо:

1. Размер модели из условия: 0.025478363037109375 Mb
2. Размер файла с моделью: 0.03270435333251953 Mb
3. MACs значение: 19321856.0

4. Compression rate по размеру модели из условия: 10.546938164395868
5. Compression rate по размеру файла с моделью: **8.431487475578106**
6. Speedup rate: **6.186125390852721**

**Дистилляция 3**

Далее я попробовал еще больше дожать модель - и у меня получилось! Уменьшил число каналов свертки еще в 2 раза, поставил размер скрытых состояний 12 вместо 16 и обучал с температурой 30. В результате получилось доучить модель до приемлемого качество за 26 эпох:




```
END OF EPOCH 25: au_fa_fr = 5.3427453493782784e-05
```

1. Размер модели из условия: 0.011951446533203125 Mb
2. Размер файла с моделью: 0.022477149963378906 Mb
3. MACs значение: 9431040.0
4. Compression rate по размеру модели из условия: 22.484200446856047
5. Compression rate по размеру файла с моделью: **12.26785183928041**
6. Speedup rate: **12.673832790445168**

**Динамическая квантизация с базовой моделью**

После этого я попробовал квантизовать модель. Попробовал я сначала это сделать для базовой модели для линейного и GRU слоев, так как только эти слои поддерживаются динамической квантизацией. У квантизованной модели получилось качество около 2.6e-5, а также неплохо улучшились метрики сжатия и скорости:

1. Размер модели из условия: 0.003082275390625 Mb
2. Размер файла с моделью: 0.0834512710571289 Mb
3. MACs значение: 20477952.0
4. Compression rate по размеру модели из условия: 87.1819306930693
5. Compression rate по размеру файла с моделью: **3.304279755442546**
6. Speedup rate: 5.836883688368837

На этом эксперименте я сначала обрадовался сжатию почти в 90 раз, а потом расстроился, что в домашке очередной баг. Именно после этого я стал замерять еще и размер файла модели, так как функция в условии, вероятно, не умеет в квантизованные веса. Как измерить ускорение на квантизованных моделях я так и не придумал. Можно было измерять время в секундах на один батч, но, как правильно замечено преподавателями, это совсем бессмысленно за счет всяких низкоуровневых оптимизаций. Но теоретически ускорение не должно изменяться при квантизации, поэтому в качестве ускорения имею в виду ускорение исходной модели (не квантизованной).

**Динамическая квантизация с самой дистиллированной моделью**

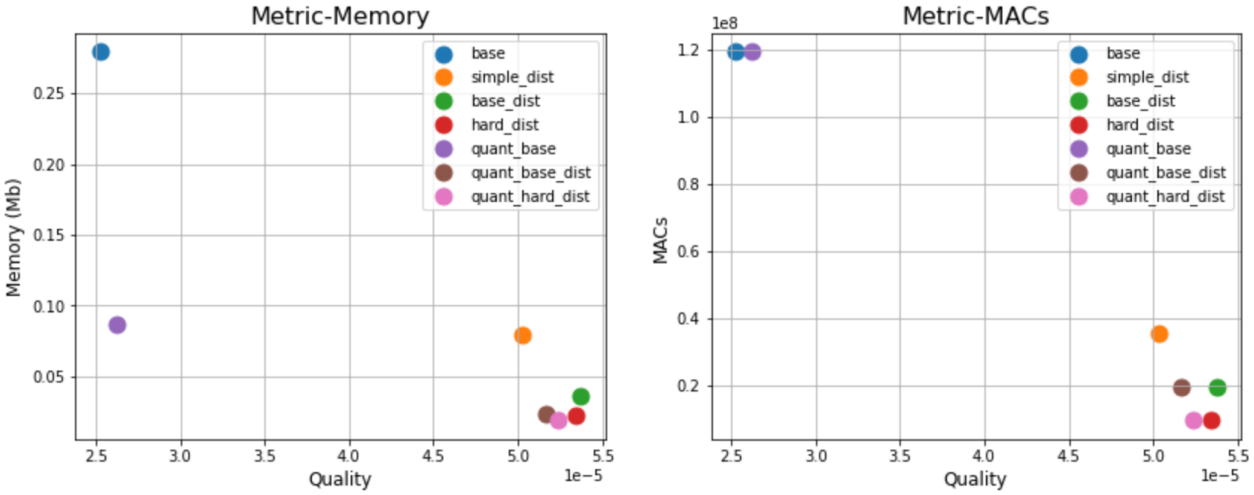
После предыдущего эксперимента, я заметил, что квантизация несильно влияет на качество модели, поэтому применил ее к лучше своей дистилляции. В результате получилась модель с качеством около 5.23e-5 против 5.34e-5 у исходной модели, что оказалось довольно интересным результатом. Но особенно хорошими получились метрики сжатия модели:

1. Размер файла с моделью: 0.00077056884765625 Mb
2. MACs значение: 9431040.0 (у исходной модели)
3. Compression rate по размеру файла с моделью: **17.90901207804274**
4. Speedup rate: **12.673832790445168** (у исходной модели)

## Conclusion

В результате работы был реализован стриминговый режим KWS-модели (хоть и не идеальный). Также базовая модель была сжата в **17.90901207804274** раз и ускорена в **12.673832790445168** раз. Для этого была использована дистилляция и встроенная динамическая квантизация из Pytorch. Также я пытался квантизовать свертку с помощью статической квантизации, но сделать это в итоге не получилось.

**График с результатами разных конфигураций**




**Мир, если бы домашки по DLA выдавались без багов…**