Image captioning

Deep learning school, part 2

Основной **целью** было написание кода для задачи Image captioning. Для этого нужно было почитать статьи, изучить подходы, выбрать датасет и фреймворк для написания кода.

Image captioning - это текстовое описание картинки, т.е. на входе имеем картинку, на выходе - текст. Одной из особенностей решения задачи является стык CV и NLP: описание картинки можно получить так:

- извлечь CNN-признаки с разных слоёв с помощью предобученной сети
- эти признаки подать в RNN
- добавить attention для контекста (чтобы новое генерируемое слово было точнее, нужен контекст)
- после RNN взять feed-forward сеть (обычный fc, например) и линейный слой для получения распределения по словам

Основные идеи довольно хорошо изложены здесь:
https://towardsdatascience.com/image-captioning-in-deep-learning-9cd23fb4d8d2
https://downloads.hindawi.com/journals/cin/2020/3062706.pdf
https://arxiv.org/abs/1502.03044

В качестве фреймворка мною был выбран **tensorflow** с целью глубже с ним ознакомиться, а также более простым способом сёрвить модели (tf-serving для docker).

В качестве датасета выбор пал на **coco2017**:
https://www.kaggle.com/aishwr/coco2017
Это размеченный датасет на английском, в котором каждая картинка имеет порядка 5 разных меток. train и val содержат 591753 и 25014 картинок (с учётом пятикратного повтора).

Эксперименты.

1. **Модели**

Encoder принимает на вход CNN-признаки (предобученной inception-V3) для attention, преобразует их в нужную размерность post_processingчерез dense слой. CNN-признаки с предпоследнего слоя идут в RNN для инициализации стейтов. Attention берёт state и признаки от энкодера и для каждого таймстемпа выдаёт контекстный вектор, который позволяет выдать новое слово.

Decoder принимает на вход target (т.е. векторизованный caption), attention-признаки и state самого декодера. Target нужен для teaching-forcing. State нужен для того, чтобы модель была стриминговой, а также корректного инференса (т.е. который бы совпадал с тренировкой). Изначально state получается путём dense-преобразования CNN-признаков с предпоследнего слоя.

Выбор моделей делал на небольшом датасете. Выбрав модель, обучал на большом датасете.

Пробовал GRUCell vs LSTMCell. Особых отличий по качеству нет, по скорости ожидаемо GRUCell быстрее. Решил всё ж LSTMCell оставить, т.к. его чаще берут.

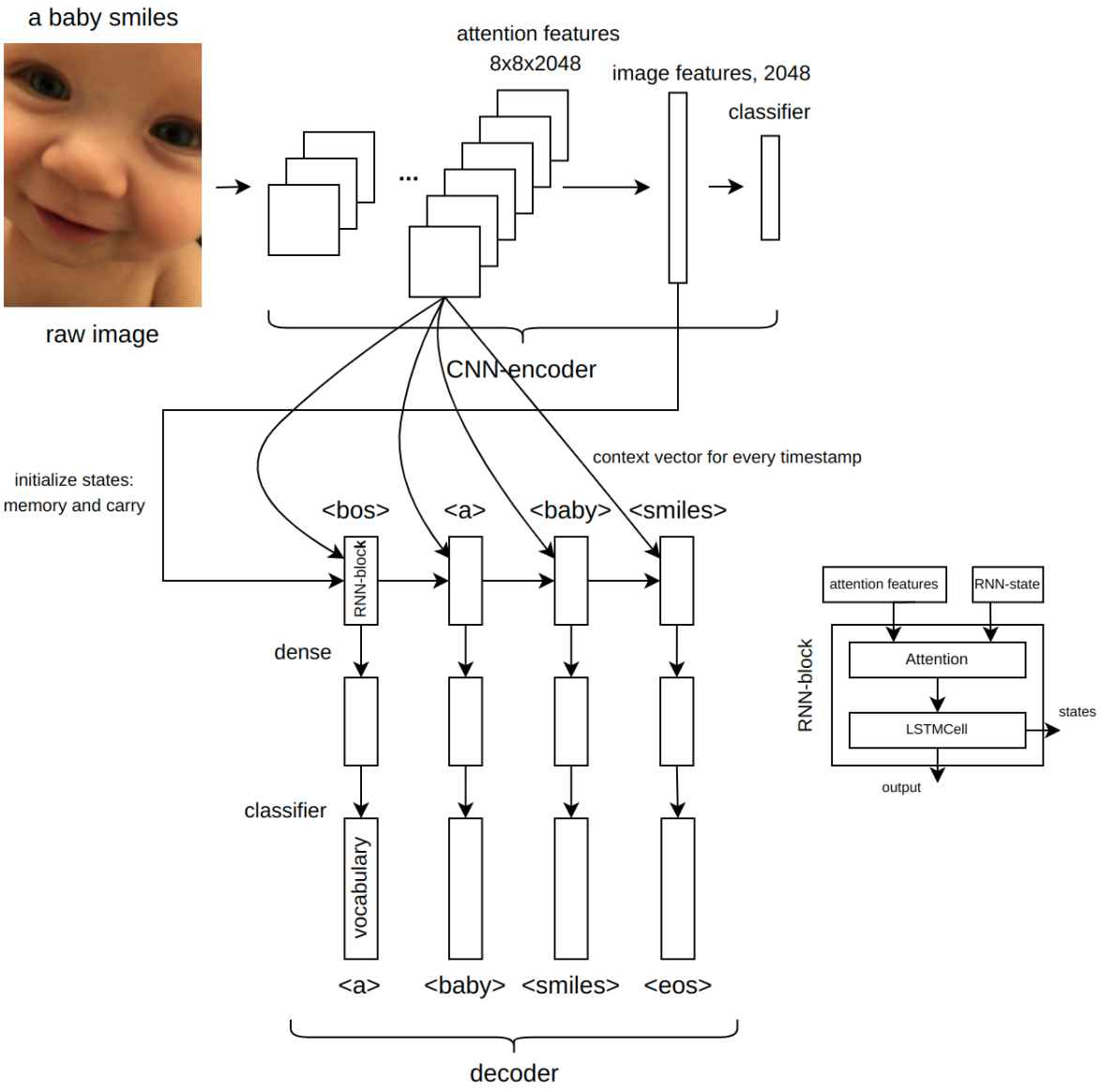
Пробовал добавить дополнительные слои к LSTMCell - профита не было, оставил один слой.

Добавление attention сильно улучшило качество.

Обучал эмбеддинги с нуля, использовал предоученные без finetuning и с finetuning. Предобученные эмбеддинги сильно улучшают качестов, последний вариант оказался лучшим, его и оставил.

Замечу, что в коде я использовал часть val-датасета для валидации, чтобы не тратить при обучении на это время. После нескольких запусков решил не считать на каждой эпохе валидацию, потому что нет доверия предлагаемым метрикам (пробовал только BLEU): модель может хорошо генерить текст по картинке, но текст будет отличен от того, что есть в val. Поэтому основной метрикой были глаза)

Итоговая модель:



2. **Данные**

Порядка 10000 картинок (с учётом кратности 5) достаточно, чтобы модель стала выдавать что-то вразумительное. При увеличении данных качество становится значительно выше. Решил увеличивать данные. При объёме ~100000 картинок качество уже неплохое, однако решил для робастности модели сделать датасет ещё больше: в итоге обучал модель на ~**590000** картинках.

Столкнулся с проблемой: изначально хотел заранее подготовить данные: извлечь CNN-признаки и сохранить их отдельно в файлы, а при обучении эти

файлы открывать. Однако реально сами файлы весят много (тот же порядок, что и картинки), а памяти не было достаточно ни на kaggle, ни на collab, поэтому решил изменить обучение с получением признаков “на ходу” (это позволило брать в разы больше данных для обучения, правда, замедлило обучение).

Обучал так: на каждой эпохе выбирал 50000 картинок и их прогонял. В рез-те вероятность выбора конкретной картинки p ≈ 50000 / 590000 = 5/59. За 45 эпох вероятность, что конкретная картинка не будет выбрана есть (1-p)^45 ≈ 0.02.

Также пробовал убирать teacher-forcing (aka tf): обучать сначала с tf, затем его частично убирать, сразу обучать частично с tf. Сложно сказать, какая модель выглядит лучше (кажется, что с tf, когда половину обучения было tf, вторую половину была смесь tf и greedy).

Итоги:

- были изучены подходы к решению задачи image captioning
- написан код на tensorflow для обучения моделей и демонстрации
- обучены несколько моделей, выбрана лучшая

Проект находится тут:
https://github.com/A-n-d-r-e-w-y/DLS_part_2/tree/main/Final_project_Image_Captioning