СОФИЙСКИ УНИВЕРСИТЕТ
„СВ. КЛИМЕНТ ОХРИДСКИ“

SOFIA UNIVERSITY
ST. KLIMENT OHRIDSKI

Курсов проект

по

Откриване на знания в текст

на тема

„Image captioning“

Изготвили:
Елена Тодорова Захариева – ФН: 25751, Спец. ИИОЗ
Калоян Иванов Спиридонов – ФН: 25750, Спец. ИИОЗ

Преподавателски колектив:
Проф. Иван Койчев
Д-р Преслав Наков
Д-р Георги Георгиев

Дата: 05.07.2019

Гр. София

Съдържание

Резюме

В основата на задачата за генериране на описание по изображение, е проблемът за разбирането на самото изображение – елементарен за човек, но комплексен за машина. В този документ предсатвяме работата си върху проект, целящ да се справи с тази задача именно по метод, близък до този на човека. Използвали сме многослойна невронна мрежа, чиито входове можем условно да разделим на две – конволюционна невронна мрежа (CNN), на която подаваме изображение и рекурентна невронна мрежа LSTM, на която подаваме текст, с цел да получим цяло описание в края.

Въведение

Както споменахме, за човек, озаглавяването и описанието на снимка не е трудност – достатъчно е само да погледне изображението. За компютрите, обаче, този проблем е оставал неразрешен до появата и развитието на изкуствения интелект и Deep learning алгоритмите.

За да обясним степента на важност на задачата, нека разгледаме възможните приложения на комютърното описание на снимки. Първото, и най-очевидно приложение, което ще разгледаме, е класификация на снимки по тяхното съдържание. Използва се от Facebook, Google и др. Можем да го видим приложено в Google Photos например – всичките ни снимки са разпределени по категории – например Плаж, Планини, Огън, Залез, Стадиони и т.н. Ако за всички снимки, разполагаме със съответните им описания, търсачките по изображения могат да станат по-добри и надеждни.
Второто приложение, на което ще се спрем, е използването на такава задача в алгоритмите за самоуправляващи се коли, тоест шофиращи се сами, без физически да е нужен шофьорът. Ако можем достатъчно добре да опишем и предадем на алгоритъма светът около тях, то той ще се справя значително по-добре с позиционирането и насочването на колата. Компании като Google и Tesla се занимават с разработването на такива автомобили.

И третото приложение, на което ще се спрем, е създаването на продукт за слепите хора, който да може да ги напътства, без да им е нужна асистенцията на друг човек. За целта трябва първо светът около тях да бъде описан и после този текст да бъде преобразуван в говор. В момента Nvidia се опитват да създадат такъв продукт.

Нашият модел приема картинка и текст, които можем да си представим като два отделни входа. Картинката е това, което всъщност ще описваме. Тя се подава на ResNet-50 неврнонна мрежа, на която предварително сме премахнали последния слой, който извършва класификацията и получаваме от нея векторното представяне на снимката, което ще подадем заедно с текста.

Текстът е поредицата от думи, които сме генерирали на предходни стъпки. Тоест на първа стъпка няма да има нищо значещо в този параметър, на втора – ще е думата генерирана от първата и т.н. докато не генерираме цялото описание. Тези два параметъра подаваме на многослойна невронна мрежа, от която като изход връщаме именно целия получен текст, стъпка по стъпка, който всъщтност представлява и генерираното описание на снимката.

Преглед на областта

Един подход, който е бил разработван за задачата за image captioning е за най-блиските съседи. Този подход намира списък от най-блиските съседни изображения в обучаващите данни, от които да заеме описание. Избираме описанието измежду кандидатите, което показва най-добри резултати.[4]

Разработка върху друг подход, който превъзхожда с 10% резултатите от по-ранните, се базира върху графи(GCap) за избиране на описание. Този подход е бърз и добре мащабируем, с линейно време за обуение и тестване спрамо размера на dataset-а.[5]

Третата статия, върху която ще се спрем описва модел, който научава зависимости и метрики между генерирано от претренирана конволюционна невронна мрежа представяне на изображение и фрази, които го описват. След това е способен да извлече фрази само по дадено му изображение. [6]

Последната разработка, на която ще обърнем внимание, е в основите и на нашата работа. Тя описва генеративен модел, базиран на дълбока рекурентна архитектура, която комбинива предимствата на computer vision и machine translation, за да генерира изречения, описващи снимка. Моделът успява да подобри досега постигнатите резултати върху много dataset-ове. Един от тях е Flickr30k, където успява да вдигне BLEU-1 score от 56 на 66. [7]

Данни и характеристики

Използвахме Frickr8k dataset-а за този проект.[1] Той съдържа 8 хиляди примера, които представляват картинка и по пет описания за всяка от тях, означени с името на снимката, диез и индекс от 0 до 4.

Пример:

```
1000268201_693b08cb0e.jpg#0	A child in a pink dress is climbing up a set of stairs in an entry way .
1000268201_693b08cb0e.jpg#1	A girl going into a wooden building .
1000268201_693b08cb0e.jpg#2	A little girl climbing into a wooden playhouse .
1000268201_693b08cb0e.jpg#3	A little girl climbing the stairs to her playhouse .
1000268201_693b08cb0e.jpg#4	A little girl in a pink dress going into a wooden cabin .
```

Данните са разделени както следва: 6000 обучаващи примера, 1000 – валидиращи и 1000 – тестови. Предварителната обработка, която сме приложили за картинките се състои в подаването им на ResNet-50 конволюционна невронна мрежа, без последния ѝ слой, която като резултат връща (от предпоследния) векторното представяне на подаденото ѝ изображение с размерност 2048. Всички вектори се записват като стойности в речник с ключ името на снимката.

Всяко описание е във вид на изречение. Създадохме аналогичен речник и за тях, в който имаме ключ – име на снимката, и стойности – масив от петте описания за всяка, в почистен вид. Почистеният вид на едно изречение наричаме текста му, с премахнати препинателни знаци и думи с дължина от един символ или такива, които съдържат числа, преобразувани само в малки букви.

Пример на първото от горепосочените описания в почистен вид:

```
A child in a pink dress is climbing up a set of stairs in an entry way . ->
child in pink dress is climbing up set of stairs in an entry way
```

На всяко такова описание се добавя най-отпред, като първа, думата startseq, а в края, като последна – endseq. Тези думи служат за означаване на началото и края на описанието съответно.

Преди, обаче, да ги подадем на невронната мрежа, тези описания трябва да бъдат дообработени до вид, представляващ вектор от числа, в който да могат да бъдат разбрани от нея. Тоест ни трябва числова репрезентация на всяка от думите. За целта използваме вградения в Keras Tokenizer, който създава word_index речник от множество от изречения. Речникът е на принципа дума -> индекс и затова всяка дума има уникален индекс, който представлява цяло положително число. Индекс се дава на база на честотата на срещане на дадената дума, като по-често срещаните думи имат по-малък индекс(първите няколко думи често са стоп думи, тъй като те се срещат най-често в текст). Например ако имате изречението „The cat sat on the mat“, ващият речник ще бъде създаден като word_index[“the”] = 1, word_index[“cat”] = 2 и т.н. 0-та е запазена за допълване на изреченията до дадена дължина, тоест нулев индекс не се дава. След като създадохме такъв речник, установихме, че имаме 7579 различни думи в нашите обучаващи примери. Когато вече имаме създаден tokenizer можем да му подадем описание и той да ни го превърне във вектор от числа, който можем да подадем на невронната мрежа. Всеки вектор на първата си позиция ще има съответстващия индекс на startseq, а на последна – този на endseq, но векторите ще бъдат с различна дължина. За да се справим с този проблем, изчислихме, че най-дългото описание е с дължина 34 думи и допълваме всички вектори до дължина 34 с нули в края. Това не пречи на комуникацията между нашите модел и речник, защото както обяснихме по-горе нулевия индекс е запазен и на него не съответства дума от tokenizer-а.

Методи

1. ResNet-50 (съкратено от Residual Networks), която сме използвали за преобразуването на изображения във вектори, е многослойна конволюционна невронна мрежа, служеща за разпознаване и класификация на изображения. Те се подават като масив с размерност 224x224x3. Претренирана е с повече от милион изображения от базата на ImageNet. Мрежата съдържа 50 слоя и може да класифицира изображенията в 1000 категории, като например молив, мишка, клавиатура и много видове животни. Както, обаче, вече обяснихме, ние премахнахме последния 50-ти слой, който извършва самата класификация и просто взимаме като изход от нея векторът, генериран за съответното изображение, точно преди да бъде подаден за класификация. Размерността му е 2048.
2. Модел. Моделът ни наподобява работата на seq2seq моделите. Или иначе казано архитектурата му е от тип енкодер – декодер. Както се разбира от името им, този тип архитектури са съставени от два основни компонента – енкодер и декодер.Енкодерът представлява стек от няколко рекурентни елемента, където всеки от тях приема елемент от входната последователност от данни, събира нужната информация за него и го предава нататък по слоевете, от които се генерира закодирания вектор. Той е последното, което се произвежда от енкодера. Този вектор цели да капсулира информацията за всички входни елементи, за да помогне на декодера да направи аксимално точни предположения. Този вектор е всъщност и входът, който се подава на декодера. Декодерът представлява стек от няколко рекурентни елемента, които на всяка времева стъпка предсказват някакъв изход.Всеки рекурентен слой получава информация от предходния и генерира изход, както и собствена информация. Предсказаното от последния слой на декодера е всъщност това, което получаваме като изход от невронната ни мрежа.
3. Енкодерът ни е съставен от следните слоеве:
    a. Input слой с размерност 2048, на който се подава векторът, съответстващ на изображение.

b. Dropout слой с 0,2 rate, който цели да не допусне overfitting, като на всяка итерация от тренирането на модела, задава на случайна част, големината на която е посочена в rate, нулеви тегла. Тоест връзките в модела, съответстващи на тези тегла, се пропускат в дадената итерация. Този слой е свързан с Input слоя от предната точка.
c. Dense слой с активационна функция relu. Dense слоевете представляват обикновени пълно свързани слоеве. Активационната функция, която се прилага – relu представлява:
$$f(x) = max(0, x)$$
Тоест за всеки вход x, връща него, ако е положителен, и 0 в противен случай. Този слой е свързан с Dropout слоя от предната точка.
d. RepeatVector слой – той повтаря подадения му вектор n на брой пъти. В нашия модел n е 34. Използваме този слой, за да преобразуваме вектора към поредица от 34 вектори, които да можем да подадем на LSTM по-нататък в мрежата. Този слой е свързан с Dense слоя от предната точка.
e. Input слой с размерност 34, на който се подава векторът представляващ изречението. На този слой в началото за всеки трениращ екземпляр ще се подава вектор съсавен само от индекса на startseq на 0-ва позиция и индекс 0 на позиции от 1 до 33. След това, ако вече например сме генерирали три думи от описанието, то техните съответни индекси ще са се появили съответно на позиции 1, 2 и 3, а останалите ще са нули. Това продължава докато не се генерира думата endseq, която означава край на описанието.
f. Embedding слой на който подаваме input и output dimentions съответно 7579 и 256. Този слой създава от всяко число, което му се подава на входа – векторно представяне със съответната размерност, посочена в output dimention. В нашия случай това е 256. Тоест на всяка стъпка ние ще подаваме на Embedding слоя вектор с дължина 34, състоящ се от цели положителни числа, а той ще ни връща 34 вектора с дължина 256, които представляват представянето на всяко от тези числа. Освен това сме подали и параметъра mask_zero=True, който означава, че нулата е използвана като специален символ за допълване до даден размер. Този слой е свързан с Input слоя от предната точка.
g. Dropout слой с 0,2 rate. – отново цели да предотврати overfitting. Този слой е свързан с Embedding слоя от предната точка.
h. LSTM слой с размерност на изхода 256x34, тъй като сме означили output dimention 256 и return sequences=True, което означава да се връщат изходите от вкяка кутийка(тоест 34 такива изхода с размерност 256). LSTM слоевете са рекурентни слоеве, които имат памет. Подходящи са за идентични като модел данни, които се повтарят през някакъв период от време. LSTM слоевете имат гейтове, които определят каква част от информацията от предната стъпка да бъде запомнена в паметта им. Така, пазейки информация от предходни изпълнения, те се опитват да открият зависимости и модели в данните, които пристигат периодично. В нашия случай – всеки път подаваме вектор с дължина 34. Не сме подали активационна функция, което означава, че се използва тази по подразбиране – tanh. Тя представлява:
$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$
Този слой е свързан с Dropout слоя от предната точка.
i. Concatenate слой – този слой конкатенира по посочена от нас ос двата вектора, получени от RepeatVector слоя и от LSTM слоя. Всеки от тези слоеве връща изход с размер 34x256, тоест след конкатенацията имаме изход с размерност 34x512. Този слой е свързан с RepeatVector и LSTM слоевете.

4. Декодерът ни се състои от следните слоеве:
   a. LSTM слой с размерност на изхода 256. Той е свързан с Concatenate слоя, който представлява и последен слой на енкодера.
   b. Dense слой с размерност големината на речника ни - 7975 и активационна функция softmax. Softmax е функция, която приема като вход вектор на K реални числа и го нормализира във вероятностно разпределение, състоящо се от K вероятности. Dense слоя връща вероятностите за всяка дума от речника да бъде новата предсказана дума. Вземаме като резултат предсказанието с най-голяма вероятност (argmax).

Експерименти

Експериментирахме с различни архитектури на модели. Първият, който опитахме беше съставен от CNN и Dense слоеве и от Embedding и LSTM слоеве, чиито изходи се подават на Concatenate слоя, а след това има два Dense слоя. От този модел получихме следните резултати:

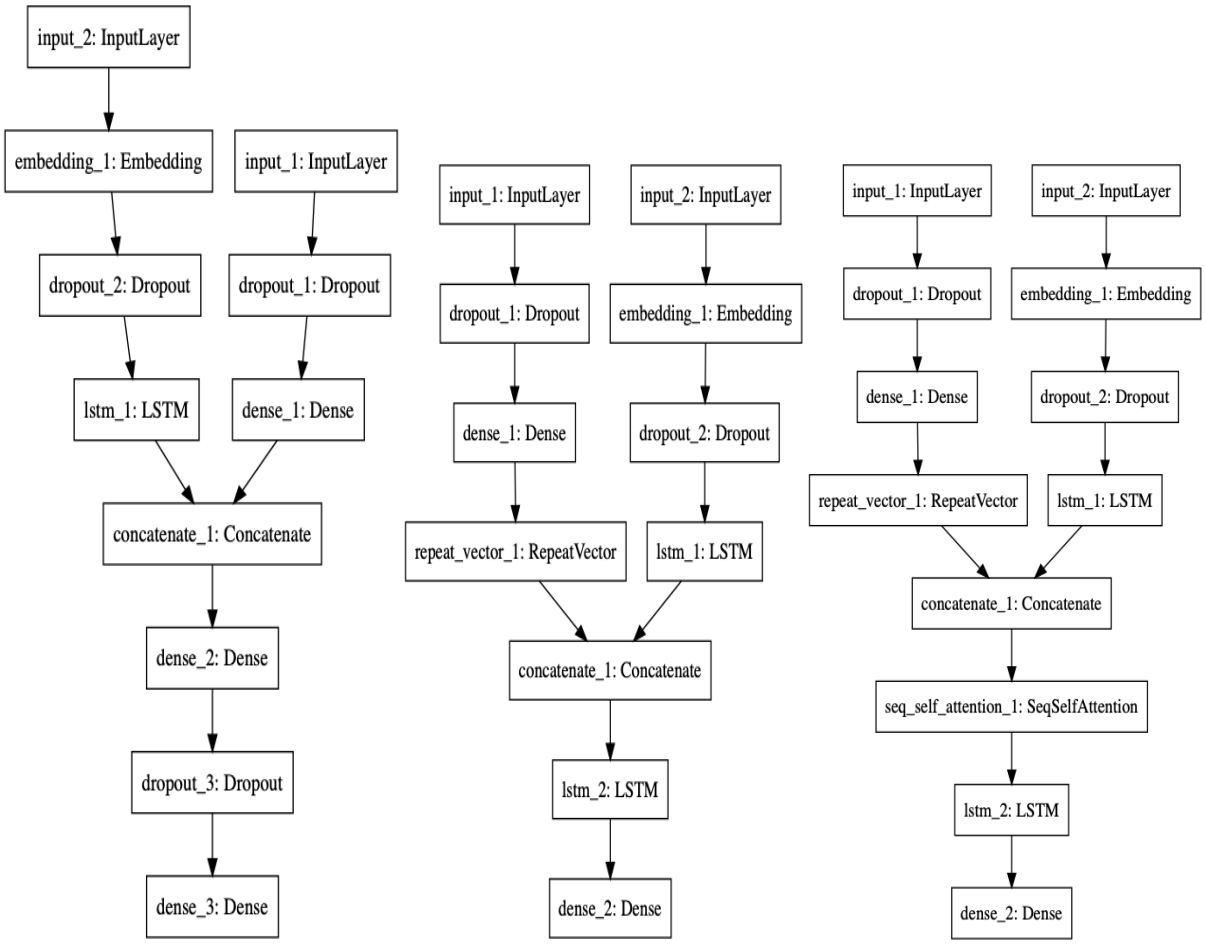
```
BLEU-1: 0.555860
BLEU-2: 0.279278
BLEU-3: 0.180292
BLEU-4: 0.077575
```

След това променихме модела с идея да се доближим повече до seq2seq архитектирата. Добавихме RepeatVector слой след CNN и Dense слоевете и преди Concatenate слоя. След него вместо два Dense слоя променихме модела така, че да имаме един LSTM и един Dense слой. Това всъщност се оказа и най-успешния модел за този проект. Резултатите му са посочени в секция Резултати.

В следващия си експеримент решихме да добавим и Attention слой към модела, който да разположим между енкодера и декодера. Към този експеримент нямахме големи очаквания, тъй като Attention се използва, когато имаме голям обем информация в дълги изречения. В Keras няма вграден Attention слой, използвахме SeqSelfAttention layer от библиотеката keras-self-attention.[2] Моделът не можа да надскочи очакванията ни и резултатите, които получихме са:

```
BLEU-1: 0.576892
BLEU-2: 0.330481
BLEU-3: 0.235706
BLEU-4: 0.118113
```

Ето и графично представяне на трите модела:



Резултати

За да оценим как се справя моделът ни използвахме метриката BLEU score. BLEU score е създеден като алгоритъм за оценяване на машинен превод от един естествен език на друг. Резултатът на BLEU score е число между 0 и 1, което цели да определи съответствията между кандидатът за оценка и множеството от реални резултати. В нашия случай ние използваме BLEU score като оценка на подобието на генерираното от модела описание с петте съдържащи се в dataset-а за съответното изображение. Резултатите ни са:

```
BLEU-1: 0.577624
BLEU-2: 0.334999
BLEU-3: 0.232525
BLEU-4: 0.112670
```

С BLEU-1 оценяваме сходствата между униграмите в кандидата и примерите тоест той зависи от това колко думи от кандидата се срещат в примерите. BLEU-2 оценява съответствията между биграмите в кандидата и примерие тоест колко двойки от думи съдържащи се в кандидата се съдържат и в примерите. Аналогично BLEU-3 работи с триграми, а BLEU-4 с четириграми.

За learning rate използвахме стойността по подразбиране за Адам оптимизатора, която е 0,001.

За да избегнем overfitting използвахме два Dropout слоя, всеки от които разглежда случайни 20% от теглата в слоя като нули. Тоест на всяка итерация случайни 20% от всички тегла не се вземат предвид от модела.

Оценките на моделите ни не даваха много добри резултати, затова решихме да проучим какъв е проблемът с тях. Първият проблем, който открихме е, че в обучаващите ни данни има 7579 различни

думи, в тестовите – 3176, но от тези 3176 има 542, които не се срещат в обучаващите. Тоест имаме 542 думи, които моделът ни не е виждал по време на обучението си и съответно няма как изобщо да познае при работата си. Докато разглеждахме резултатите и данните си, откихме че някои генерирани от нашия модел описания са в сегашно продължително време (напр. Two dogs are playing on the grass), а в примерите, които имаме за тестване изречението е в сегашно просто време (Two dogs paly on the grass.). Въпреки, че семантичните значения на двете изречения съвпадат, тоест моделът ни се е справил перфектно със задачата си, BLEU score за този кандидат е едва 0,71, защото той разглежда play и are playing като различни думи. За да се справим с този проблем направихме следния експеримент. Приложихме лематизатор върху двете изречения и това подобри BLEU score на 0,82, защото след лематизация playing става play, тоест имаме още една съвпадаща дума. След като преценихме, че това е значително подобрение, решихме да приложим тази стратегия върху всички тестови примери. Така резултатите на модела ни се покачиха до следните:

```
BLEU-1: 0.607223
BLEU-2: 0.351466
BLEU-3: 0.251055
BLEU-4: 0.128617
```

BLEU-1 score се е покачил с 3 процента, BLEU-2 и BLEU-3 с по 2, а BLEU-1 с един процент. Това, обаче, не може да се разглежда като подобрение на модела, защото единственото което сме променили е начина, по който се оценяват резултатите.

Заключение

Целта на този проект беше да се разгледат различни модели за извършване на image captioning. В обобение можем да заключим, че Sequence to sequence(seq2seq) архитектурата даде най-добри резултати.

След експериментите ни със Seq2seq с Attention установихме, че този тип архитектура е излишна за нашите нужди, защото Attention слоя помага на модела да запомни повече информация и да се справя по-добре с откиването на зависимости в дълга последователност от данни. В нашите данни описанията са относително кратки и нямаме дълги последователности от информация.

Относно оценяването на резултатите от различните ни опити, установихме, че BLEU score (в частност BLEU-3 и BLEU-4) не е добра метрика, за измерване на това колко е добър даден image captioning модел. След наблюдения върху резултатите ни, забелязахме, че моделът ни е генерирал много точни описания на някои снимки, но заради несъвпадения на конкретните използвани думи или времена в изречението, BLEU оценката е много ниска.

Бъдещо развитие

За постигане на по-добри резултати в бъдеще, моделът може да се обучи с по-голям набор от данни(Например Flickr30k, който съдържа 30000 примера). При оценяването на модела, когато се търси предсказаната дума може да се използва Beam Search, вместо използвания от нас Greedy Search, който просто взима аргумента с най-висока вероятност(прилага се argmax върху вероятностите на всички думи). Може да се направи изследване за по-добра метрика за оценяване на такъв тип модели от BLEU score.

Принос

Работата по този проект може да се раздели на следните отделни точки: Предварителна обработка на снимките, предварителна обработка на описанията, създаване на прост първоначален модел, създаване на seq2seq модел, създаване на seq2seq модел с attention слой и анализ на получвните резултати.

Разделихме работата си по равно, като Калоян обработи текста, а Елена – изображенията. Калоян създаде и обучи първият модел и го разшири до seq2seq архитектурата, а Елена – моделът с Attention слой. В анализа на резултатите, които получихме се включихме и двамата и открихме проблеми свързани с неправилно разпределение на думите в train и test сетовете, както и причини, поради които BLEU score е толкова нисък.

Литература

[1] https://www.kaggle.com/srbhshinde/flickr8k-sau

[2] https://pypi.org/project/keras-self-attention/

[3] https://keras.io/

[4] https://arxiv.org/pdf/1505.04467.pdf?fbclid=IwAR2xGCqEcu10YvJ9yQ-WL0nue-vjShMHu6eTUo49rZzsG4MoA65CpSWzHRw

[5]https://ieeexplore.ieee.org/abstract/document/1384943?fbclid=IwAR39LBIVldiyYMpN1MAruWdAFRKpaa_S_pTQeZbxAPYYZ3RTlLyLLV7BGJM

[6] https://arxiv.org/abs/1502.03671?fbclid=IwAR2AmtDuALoCe-BWG3GJ2QEWhnkcSxBgNzKGE5C5oSxt7Mb9sOa6hfFxdbc

[7]https://arxiv.org/pdf/1411.4555.pdf?fbclid=IwAR39LBIVldiyYMpN1MAruWdAFRKpaa_S_pTQeZbxAPYYZ3RTlLyLLV7BGJM

[8] https://arxiv.org/abs/1512.03385

[9] https://arxiv.org/abs/1409.3215

[10] https://nlp.stanford.edu/pubs/emnlp15_attn.pdf

Декларация за липса на плагиатство
– Тази курсова работа е наша работа, като всички изречения, илюстрации и програми от други хора са изрично цитирани.
– Тази курсова работа или нейна версия не са представени в друг университет или друга учебна институция.
– Разбираме, че ако се установи плагиатство в работата ми ще получа оценка “Слаб”.
– Трите имена и подпис на студента:

Калоян Иванов Спиридонов: Елена Тодорова Захариева: