Распознание капчи (captcha) с помощью свёрточной нейронной сети (CNN)

Volkoshkursk ©

2017

Постановка задачи:

Распознать буквенно-цифровой чёрно - белый код на картинке, получаемый с помощью генератора от автора Piotr Kuszaj (https://github.com/kuszaj/claptcha) защищённого лицензией MIT License с помощью свёрточной нейросети, написанной на python 3 с использованием Keras, Tensorflow, PIL (pillow) и некоторых вспомогательных библиотек

Свёрточная нейросеть (CNN)

Свёрточная нейронная сеть (convolutional neural network, CNN, LeNet) была представлена в 1998 году французским исследователем Яном Лекуном (Yann LeCun) [1], как развитие модели неокогнитрон (neocognitron).

Свёрточный слой

Имея двумерное изображение **I** и небольшую матрицу **K** размерности (так называемое ядро свертки), построенную таким образом, что она графически кодирует какой-либо признак, можно вычислить свернутое изображение **I * K**, накладывая ядро на изображение всеми возможными способами и записывая сумму произведений элементов исходного изображения и ядра:

$$(I * K)_{xy} = \sum_{i=1}^{h} \sum_{j=1}^{w} K_{ij} \times I_{x+i-1,y+j-1}$$

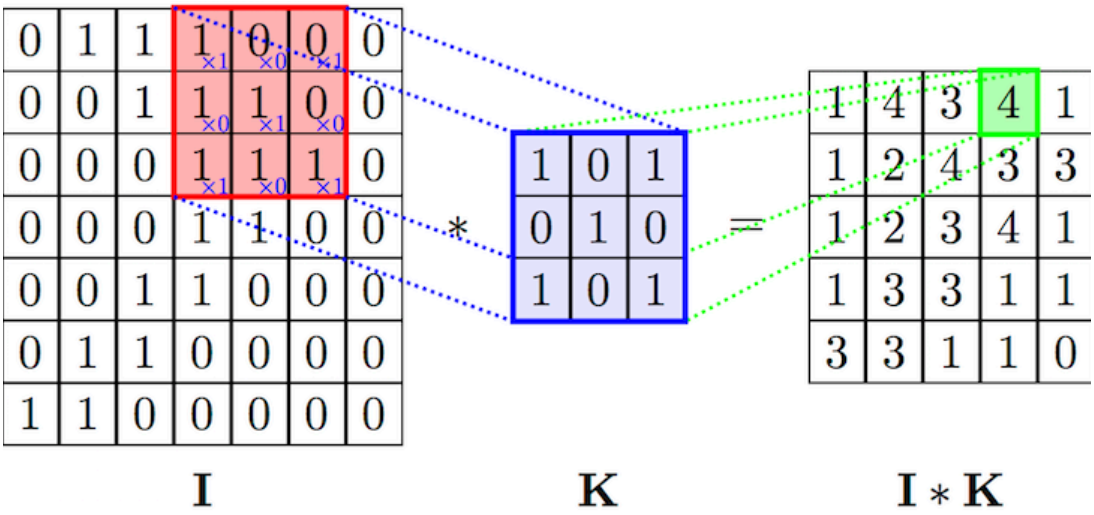


схема работы свёрточного слоя

(https://habrahabr.ru/company/wunderfund/blog/314872/)

Оператор свертки составляет основу сверточного слоя (convolutional layer) в CNN. Слой состоит из определенного количества ядер (с аддитивными составляющими смещения для каждого ядра) и вычисляет свертку выходного изображения предыдущего слоя с помощью каждого из ядер, каждый раз прибавляя составляющую смещения. В конце концов ко всему выходному изображению может быть применена функция активации .

Субдискретизирующий слой

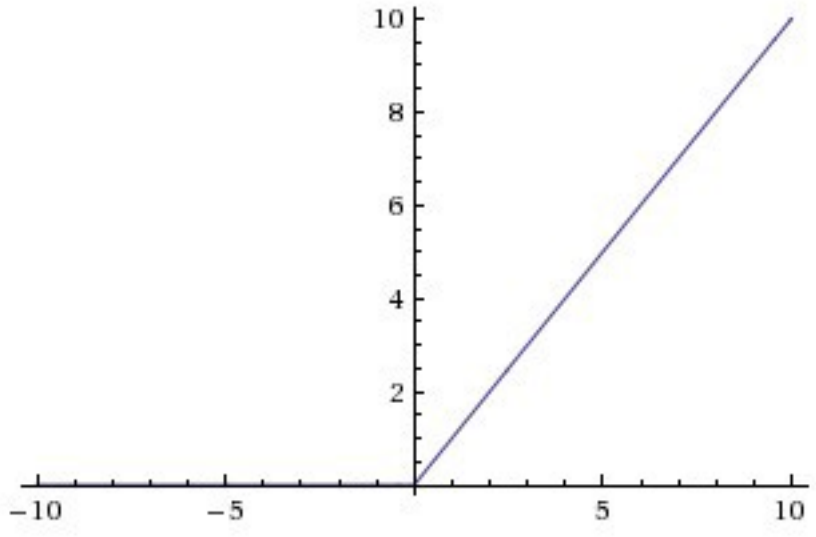
В этом разделе мы поговорим про субдискретизирующий (subsampling) слой. Слои этого типа выполняют уменьшение размера входной карты признаков (обычно в 2 раза). Это можно делать разными способами, в данном случае мы рассмотрим метод выбора максимального элемента (max-pooling) - вся карта признаков разделяется на ячейки 2x2 элемента, из которых выбираются максимальные по значению.

Использование этого слоя позволяет улучшить распознавание образцов с изменённым масштабом (уменьшенных или увеличенных).

Здесь применяются слои свёртки и подборки ещё раз (но меньшего размера и с большим количеством карт признаков)

Слой MLP

Последний из типов слоёв это слой ”обычного” многослойного перцептрона (MLP) (500/562) [1*] нейрона с функцией активации A(x) = max(0,x)(relu))[2]

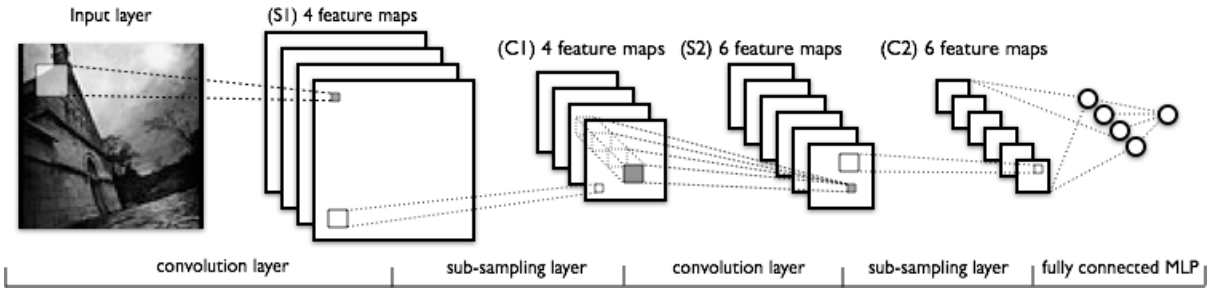


1) здесь и далее первое значение для числовой, а второе для букв и цифр

И, наконец, последний слой состоит из 10/36 нейронов с функцией активации softmax [3](MLP)

$$p_i = \frac{\exp(q_i)}{\sum_{j=1}^{n} \exp(q_j)},$$

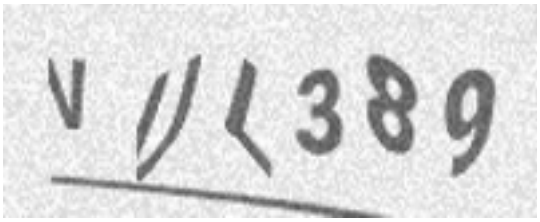
Общую схему сети можно представить так: (фото взято с http://deeplearning.net)



Капча (captcha)

Данный генератор создаёт капчу со следующими защитами:

- фоновый шум
- буквы на разной высоте
- буквы наклонены
- толщина линий букв меняется в большом диапазоне
- поперечные линии имеют тот же диапазон изменения толщины, что и буквы



пример капчи

Подготовительные операции

- Картинка приводится к чёрно-белому виду (что удаляет фоновый шум)
- Картинка делится на 6 частей
- Символы выравниваются по высоте

Зависимость параметров сети и точности работы на буквенно-числовых множествах

Зависимость количества нейронов в слое MLP и точности можно представить в виде таблицы (обучающее и тестовое множества для всех примеров одинаковые, количество эпох и другие параметры также не менялись):

количество нейронов в слое MLP	**точность на тестовых данных**
500	79,25 %
524	78,95 %
525	79,91 %
526	80,19 %
527	79,38 %
528	79,67 %
550	79,70 %
561	79,42 %
562	**81,32 %**
563	77,91 %
575	79,66 %
600	78,66 %

Зависимость количества нейронов во 2 свёрточном слое и точности можно представить в виде таблицы (обучающее и тестовое множества для всех примеров одинаковые, количество эпох и другие параметры также не менялись, в слое MLP 562 нейрона):

количество нейронов во 2 свёрточном слое	**точность на тестовых данных**
100	**81,32 %**
110	78,08 %
120	79,84 %
130	74,37 %

Зависимость размера ядра в 1 субдискретизирующем слое и точности можно представить в виде таблицы (обучающее и тестовое множества для всех примеров одинаковые, количество эпох и другие параметры также не менялись, в слое MLP 562 нейрона, во 2 свёрточном - 100):

размер ядра в 1 субдискретизирующем слое	**точность на тестовых данных**
2x2	**81,32 %**
2x5	72,77 %
1x1	76,30 %
3x3	68,54 %

Если у свёрточных слоёв и основного персептрона поменять функцию активации на softmax (остальные параметры прежние) результат 2.74%, а если у персептрона, выдающего результат - то 5.33%

Если у варианта с максимальным результатом (81,32 %) увеличить количество эпох до 36, то результат увеличится до 83.56%

Однако если изменять размер ядра во 2 субдискретизирующем слое, то зависимость его и точности будет представлять более интересный результат. Их также можно представить в виде таблицы (обучающее и тестовое множества для всех примеров одинаковые, количество эпох и другие параметры также не менялись, в слое MLP 562 нейрона, во 2 свёрточном - 100):

размер ядра в 2 субдискретизирующем слое	**точность на тестовых данных**
2x2	81,32 %
2x5	81,28 %
1x1	74,41 %
3x3	78,40 %
2x6	79,94 %
3x5	79,34 %

А если при этом ещё менять размер ядра свёртки во 2 слое то результат будет следующим:

размер ядра в 2 субдискретизирующем слое	**размер ядра свёртки во 2 слое**	**точность на тестовых данных**
2x5	**3x3**	**81,91 %**
2x6	**3x3**	81,45 %

размер ядра в 2 субдискретизирующем слое	размер ядра свёртки во 2 слое	точность на тестовых данных
2x5	**2x3**	79,45 %
2x5	**2x5**	79,75 %

Опять можно изменить количество нейронов во 2 сверхточном слое:

количество нейронов во 2 свёрточном слое	точность на тестовых данных
100	81,91 %
105	79,11 %
109	81,39 %
110	81,85 %
111	***82,18 %***
112	81,60 %

Если посмотреть на отчёт при обучении (графа точность на множестве подтверждения), можно построить таблицу зависимости этой точности и номера эпохи обучения:

номер эпохи	точность на данных подтверждения (validate acc)
1	54,68 %
2	68,18 %
3	73,70 %
4	76,11 %
5	78,62 %
6	79,27 %
7	79,58 %
8	79,89 %
9	81,60 %
10	81,85 %
11	81,87 %
12	82,03 %

номер эпохи	точность на данных подтверждения (validate acc)
13	82,25 %
14	83,33 %
15	83,22 %
16	83,07 %
17	83,17 %
18	82,98 %
19	83,28 %
20	83,67 %
21	83,87 %
22	83,50 %
23	83,17 %
24	83,71 %
25	***83,88 %***
26	83,55 %
27	83,81 %
28	83,67 %
29	83,83 %
30	83,82 %
31	83,83 %
32	83,84 %
33	83,87 %
34	83,67 %
35	***83,92 %***
36	83,78 %

Итог при такой конфигурации (за 36 эпох) - ***84,36%***

Если же количество эпох уменьшить до 25, то ***83,80%***

А если до 35, то ***84,45%***

Итоговые параметры для буквенно-цифровой капчи

1. Слой свертки, 75 карт признаков, размер ядра свертки: 5x5, функция активации - relu.
2. Слой подвыборки, размер пула 2x2.
3. Слой свертки, 111 карт признаков, размер ядра свертки 3x3, функция активации - relu.

4. Слой подвыборки, размер пула 2x5.
5. Полносвязный слой, 562 нейронов, функция активации - relu.
6. Полносвязный выходной слой, 36 нейронов, которые соответствуют классам букв и цифр 0 - 9 A-Z, функция активации - softmax.

Оптимизатор Адама, обучение с помощью обратного распространения ошибки, которая вычисляется с помощью перекрестной энтропии

Обучение длится 35 эпох, результат работы на тестовом множестве составил ***84,45%***

Файл с программой - z1/z1.py

Параметры сети для цифровой капчи

Для неё подошли параметры из примера распознавания цифр mnist [4]

1. Слой свертки, 75 карт признаков, размер ядра свертки: 5x5.
2. Слой подвыборки, размер пула 2x2.
3. Слой свертки, 100 карт признаков, размер ядра свертки 5x5.
4. Слой подвыборки, размер пула 2x2.
5. Полносвязный слой, 500 нейронов.
6. Полносвязный выходной слой, 10 нейронов, которые соответствуют классам цифр от 0 до 9.

Оптимизатор Адама, обучение с помощью обратного распространения ошибки, которая вычисляется с помощью перекрестной энтропии

Обучение длится 10 эпох

Результат работы на тестовом множестве составил ***98.04%***

Файл с программой - z0/z0.py

Содержимое архива

В папке «z0» помимо «z0.py» лежит ещё бинарный файл с весовыми коэффициентами «weight.npy», скрипт, генерирующий капчу и ответ на неё «test.py», бинарные файлы с подготовленным представлением обучающего/тестового (после обучения создавались новые) множества картинок «save.npy» и ответов к ним «ans_save.npy»

В папке «z1» помимо «z1.py» лежит ещё бинарный файл с весовыми коэффициентами «weight.npy», скрипт, генерирующий капчу и ответ на неё «test.py», бинарные файлы с подготовленным представлением обучающего множества картинок «save.npy» и ответов к ним «ans_save.npy», а также бинарные файлы с подготовленным представлением тестового множества картинок «save_test.npy» и ответов к ним «ans_save_test.npy»

Список литературы

- LeCun, Yann. «LeNet-5, convolutional neural networks» – http://yann.lecun.com/exdb/lenet/
- Е.С.Борисов «Классификатор изображений на основе свёрточной сети». http://mechanoid.kiev.ua/ml-lenet.html
- А. Созыкин «Сверточная нейронная сеть для распознавания рукописных цифр MNIST» https://www.asozykin.ru/deep_learning/2017/05/08/CNN-for-MNIST.html
- Keras Documentation https://keras.io/
- Claptcha documentation https://github.com/kuszaj/claptcha
- python 3 documentation https://docs.python.org/3/