

Глубокое погружение в глубокое обучение



Шпилевский Яромир

Ведущий разработчик First Line Software

Agenda

- Рассмотрим некоторые типы слоёв (на примере Keras).
 - Какие у них есть гиперпараметры.
 - Для чего они нужны.
- Xception Network в качестве примера, как можно комбинировать слои.



SKILLFACTORY

Типы слоёв



Keras. Layers. Conv2D

- `tensorflow.keras.layers.Conv2D(`
 `filters, kernel_size, strides=(1, 1), padding='valid',`
 `data_format=None, activation=None`
 `)`






- Двумерная свёртка.

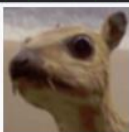

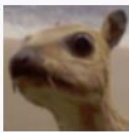
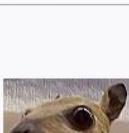
$$\begin{bmatrix} 4 & 5 & 4 & 4 \\ 4 & 2 & 3 & 5 \\ 5 & 5 & 4 & 4 \\ 7 & 7 & 5 & 5 \end{bmatrix} \overset{\text{kernel / ядро свертки / filter / фильтр}}{\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}} \text{ conv } \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \left[\begin{array}{cc} 6 & 8 \end{array} \right]$$

- kernel / ядро свертки / filter / фильтр подбирается в результате обучения.
- Смысл – получить «агрегат» пикселя и того, что вокруг него.

Свёртки изображений известны довольно давно

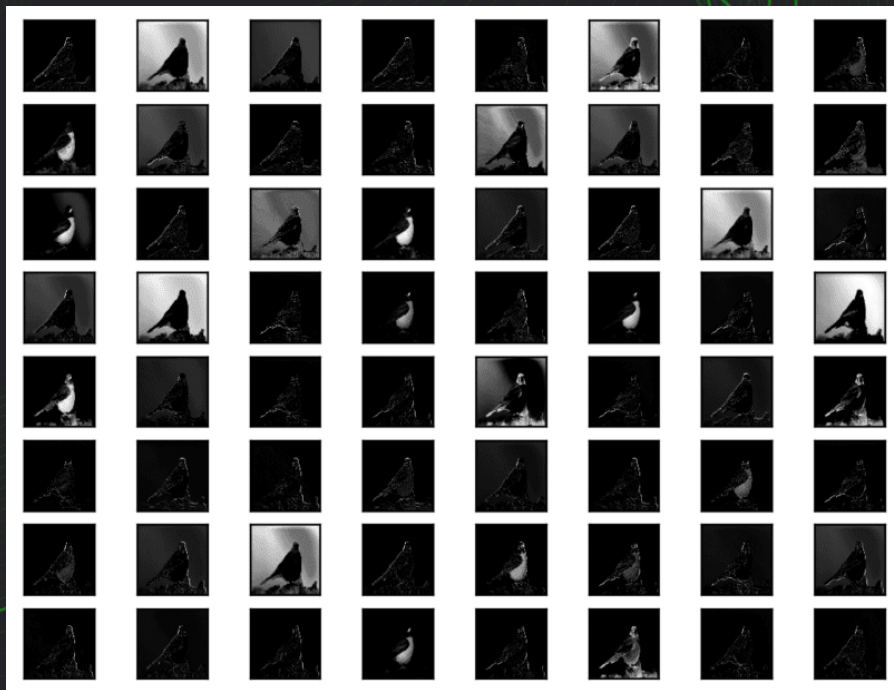
- В том числе известны фильтры для определённых преобразований: [\[1\]](#)

Operation	Kernel w	Image result $g(x,y)$
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	

Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur 3×3 (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
Gaussian blur 5×5 (approximation)	$\frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	
Unsharp masking 5×5 Based on Gaussian blur with amount as 1 and threshold as 0 (with no image mask)	$\frac{-1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & -476 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	

Keras. Layers. Conv2D. Для чего используется

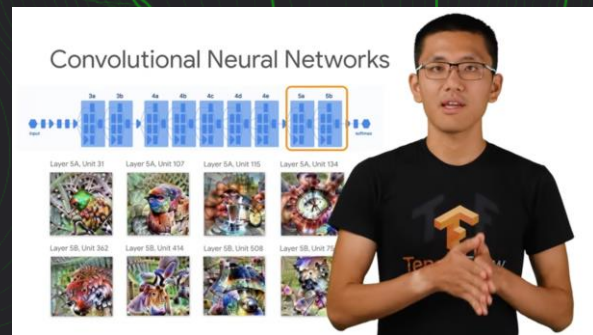
- Смысл фильтра – получить «агрегат» пикселя и того, что вокруг него.
- Что, если обучаться на изображениях интересных объектов и в процессе обучения находить значения фильтров, которые выделяют признаки, специфичные для определённого объекта?



Глубокое погружение в глубокое обучение

Визуализация промежуточных результатов

- Большая проблема в машинном обучении — интерпретируемость модели.
- Некоторые модели – неинтерпретируемы.
 - Или нет? Просто «не доходили руки» разложить их на составляющие?
- TensorFlow Lucid [\[2\]](#) [\[3\]](#)

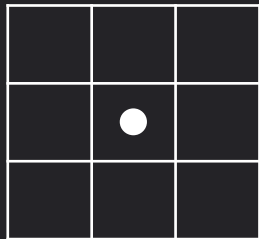


Keras. Layers. Conv2D. filters

- Количество kernels, которое нужно найти при обучении.
- Часто чем глубже слой в сети, тем большее количество фильтров обучают.
 - Фильтры ближе к входным сигналам находят в массивах пикселей более простые паттерны: границы объектов, углы, точки и т.п.
 - Фильтры ближе к выходным сигналам находят более сложные паттерны из комбинаций более простых: треугольник из углов и границ и т.п.
- Но бывают сети, где количество фильтров одинаковое на всех слоях.

Keras. Layers. Conv2D. kernel_size

- Размер ядра свёртки.
- Либо кортеж из двух элементов с размерами сторон, либо одно число, если шаг одинаковый по горизонтали и по вертикали.
- В подавляющем большинстве случаев используются нечетные числа.
- Цель - взять центральный пиксель и учесть всё его окружение из n пикселей.
- Поэтому:
 - 1 – центральный пиксель.
 - n соседей в каждую сторону по оси.
 - $2*n + 1$



Keras. Layers. Conv2D. strides

- Шаг наложения kernel'а.
- Либо кортеж из двух элементов с размерами сторон, либо одно число, если шаг одинаковый по горизонтали и по вертикали.
- В большинстве случаев (1, 1).
- Иногда применяют значение больше для того, чтобы уменьшить размер выхода.
 - Но редко, т.к. теряется часть информации.

Keras. Layers. Conv2D. padding

- Заполнять или нет изображение «полями» с нулями, чтобы kernel помещался на изображении целое число раз.
- valid – не заполнять, ведёт к небольшому уменьшению размерности.
- same – дозаполнить нулями.
- По умолчанию – valid, хотя чаще нужен same.
 - Не хотим терять информацию. Свёртка вокруг граничных пикселей тоже может дать полезный признак.

0	0	0	0	0	0
0	4	5	4	4	0
0	4	2	3	5	0
0	5	5	4	4	0
0	7	7	5	5	0
0	0	0	0	0	0

Keras. Layers. Conv2D. activation

- По умолчанию - None.
- Часто используют с None, потому что активация ставится далее отдельным слоем.
- Но можно также задать функцию активации прямо в слое свёртки:
 - ReLU
 - Leaky ReLU
 - ELU
 - Sigmoid
 - Tanh
 - Step

Keras. Layers. Conv2D. data_format

- Чисто технический параметр. Задаёт очередность компонентов во входном сигнале. Нужен, т.к. Keras может использовать разные бекенды. Компоненты:
 - Height (кол-во пикселей)
 - Width (кол-во пикселей)
 - Depth (кол-во байт на пиксель)
- `channels_last` – HWD
- `channels_first` – DHW
- TensorFlow использует `channels_last`.
- Theano использует `channels_first`.
- Начиная с версии Keras 2.4 TensorFlow – основной бекенд.

Keras. Layers. Conv1D

- Одномерная свёртка.
- Эквивалентна Conv2D, если в `kernel_size` один из параметров равен 1.
- Может применяться не только для анализа изображений.
 - Например, для анализа временных рядов.
 - Одномерная свёртка «агрегирует» информацию о текущей точке и её ближайших соседях в прошлом и будущем.

Keras. Layers. Conv3D

- Трёхмерная свёртка.
- Аналогична Conv2D, только свёртывает значения по 3-м измерениям.

Keras. Layers. SeparableConv2D

- Раздельная свёртка: последовательность spatial convolution и pointwise convolution
- spatial convolution
 - Обработывает только пространственную корреляцию в рамках отдельного канала.
- pointwise convolution
 - Обработывает только межканальную корреляцию.
- Гипотеза – можно разделить обработку этих аспектов информации без потери выведенных свойств.
- Если это так, то две более простые свёртки проще вычислять.
- Параметры – аналогичны Conv2D.
- SeparableConv1D и SeparableConv3D – для одномерной раздельной свёртки и трёхмерной раздельной свёртки соответственно.

Keras. Layers. MaxPooling2D

- Отбирает максимальное значение из пула.
- Смысл – наиболее значимый признак.

$$\begin{bmatrix} \overline{4} & \overline{7} & 4 & 4 \\ \overline{4} & \overline{2} & 3 & 5 \\ \overline{5} & \overline{5} & 4 & 4 \\ 7 & 7 & 5 & 5 \end{bmatrix} \quad 7$$

- Также – уменьшает размерность задачи для последующих слоёв.

Keras. Layers. MaxPooling2D. Параметры

- `pool_size` – размер пула, из которого выбирать максимум.
 - Либо кортеж из двух элементов с размерами сторон, либо одно число, если шаг одинаковый по горизонтали и по вертикали.
 - Похож на `kernel_size` из `Conv2D`.
- `strides` – шаг пулинга.
 - По умолчанию равен `pool_size`.

4	7	4	4
4	2	3	5
5	5	4	4
7	7	5	5

- `padding` – заполнение полями с нулями для того, чтобы `pool_size` поместился целое число раз.

Keras. Layers. MaxPooling1D

- Одномерный max pooling.
- Эквивалентна MaxPooling2D, если в pool_size один из параметров равен 1.
- Может применяться не только для анализа изображений.
 - Например, для анализа временных рядов.
 - Одномерный max pooling отбирает максимум из текущей точки и её ближайших соседей в прошлом и будущем.

Keras. Layers. MaxPooling3D

- Трёхмерный max pooling.
- Аналогичен MaxPooling2D, только отбирает максимум из значений по 3-м измерениям.

Keras. Layers. BatchNormalization

- Проблема: выход каждого предыдущего слоя сильно влияет на вход последующего.
- Изменились коэффициенты слоя – изменилось распределение выходного сигнала.
- Этот эффект называется internal covariate shift.
- Нормализует входы – в рамках батча данных нормализует их, чтобы привести к распределению Гаусса.
- Обучение нормализует данные различными способами, чтобы найти оптимальное значение среднего и среднеквадратического отклонения.
- Это сглаживает оптимизируемую функцию.
- Что позволяет двигаться с большим шагом обучения.
- А значит уменьшает количество эпох, которое необходимо для сходимости
- И уменьшает ошибку обобщения (generalization error) – насколько точно модель может работать с данными, которые до этого не видела.
- Часто применяется после свёрточных слоёв перед функцией активации.
- Ко входу слоя – реже, т.к. входы – результат нелинейной функции и их распределение может быть другим, не распределением Гаусса.

Keras. Layers. Dropout

- Случайным образом обнуляет входной сигнал.
- Смысл – борьба с переобучением (overfitting).
- Также масштабирует необнулённые сигналы, чтобы сумма сигналов осталась неизменной.
- `rate` – коэффициент от 0 до 1. Какую долю входных сигналов обнулить.
 - Необнулённые входы умножаются на $1/(1 - \text{rate})$.
- `noise_shape` – маска, которую применить к входному сигналу.
 - Dropout слой может не только обнулять, но и применять шум.
 - Смысл тот же – борьба с переобучением.
 - По умолчанию значение `None` – обнулить.
- `seed` – `seed` для инициализации генератора случайных чисел, на основе которого выбирается, какие сигналы обнулять.

Keras. Layers. Dense

- Полносвязный слой.
- Смысл – определяет, какие признаки больше характерны для определённого класса.
- `units` – количество выходов.
- `activation` – функция активации.



SKILLFACTORY

Xception Network



Xception Network. Пример реализации на Keras

```
# Метод для создания модели Xception network.  
def build_xception_network(input_shape, num_classes):
```

```
    inputs = tf.keras.Input(shape=input_shape)
```

```
    # Нарастивание можно сделать частью архитектуры нейронной сети.  
    #x = augmentation_network(inputs)
```

```
    x = tf.keras.layers.experimental.preprocessing.Rescaling(1.0 /  
255)(inputs)
```

```
    x = tf.keras.layers.Conv2D(32, 3, strides=2, padding="same")(x)
```

```
    x = tf.keras.layers.BatchNormalization()(x)
```

```
    x = tf.keras.layers.Activation("relu")(x)
```

```
    x = tf.keras.layers.Conv2D(64, (3), padding="same")(x)
```

```
    x = tf.keras.layers.BatchNormalization()(x)
```

```
    x = tf.keras.layers.Activation("relu")(x)
```

```
    previous_block_activation = x
```

```
    for size in [128, 256, 512, 728]:
```

```
        x = tf.keras.layers.Activation("relu")(x)
```

```
        x = tf.keras.layers.SeparableConv2D(size, 3, padding="same")(x)
```

```
        x = tf.keras.layers.BatchNormalization()(x)
```

```
        x = tf.keras.layers.Activation("relu")(x)
```

```
        x = tf.keras.layers.SeparableConv2D(size, 3, padding="same")(x)
```

```
        x = tf.keras.layers.BatchNormalization()(x)
```

```
        x = tf.keras.layers.MaxPooling2D(3, strides=2, padding="same")(x)
```

```
        residual = tf.keras.layers.Conv2D(size, 1, strides=2, padding="same")(  
            previous_block_activation
```

```
        )
```

```
        x = tf.keras.layers.add([x, residual])
```

```
        previous_block_activation = x
```

```
    x = tf.keras.layers.SeparableConv2D(1024, 3, padding="same")(x)
```

```
    x = tf.keras.layers.BatchNormalization()(x)
```

```
    x = tf.keras.layers.Activation("relu")(x)
```

```
    x = tf.keras.layers.GlobalAveragePooling2D()(x)
```

```
    if num_classes == 2:
```

```
        activation = "sigmoid"
```

```
        units = 1
```

```
    else:
```

```
        activation = "softmax"
```

```
        units = num_classes
```

```
    x = tf.keras.layers.Dropout(0.5)(x)
```

```
    outputs = tf.keras.layers.Dense(units, activation=activation)(x)
```

```
    return tf.keras.Model(inputs, outputs)
```

Xception Network. Некоторые принципы

- Rescale – нормирование входов.
 - Яркость пикселя каждого канала 0-255 преобразуем к 0-1.0 .
- Нераздельная свёртка при маленьком количестве фильтров – учитываем и пространственную корреляцию и межканальную корреляцию.
- При среднем количестве фильтров используем параллельно нераздельную и раздельную свёртку.
- При большом количестве фильтров – раздельную свёртку.
- GlobalAveragePooling – усреднение результатов.
- Dropout 50% - 50%, чтобы предотвратить overfitting.
- Полносвязный слой, чтобы сопоставить признаки с классом.

Ссылки

1. Kernel ([https://en.wikipedia.org/wiki/Kernel_\(image_processing\)](https://en.wikipedia.org/wiki/Kernel_(image_processing)))
2. TensorFlow Lucid (<https://github.com/tensorflow/lucid>)
3. Visualizing Convolutional Neural Networks using Lucid (<https://www.youtube.com/watch?v=b27hzEs8YWW>)

Глубокое погружение в глубокое обучение

Спасибо!

