Свёрточные сети



Полносвязная сеть



Свёрточная сеть

Свёрточные слои:

model.add(Conv2D(filters = 8, kernel\_size = (5,5), padding = 'same', activation ='relu', input\_shape = (28,28,1)))

- этот слой создает ядро свертки, которое свертывается с входными данными слоя для получения тензора выходных данных.

filters = 8 - размерность выходного пространства (т.е. количество выходных фильтров в свертке) ???

kernel\_size = (5,5) - кортеж/список из 2 целых чисел, задающий высоту и ширину окна двумерной свертки. Может быть одним целым числом для указания одинакового значения для всех пространственных измерений.

padding = 'same' - один из "valid" или "same". "valid" означает отсутствие заполнения, "same" приводит к заполнению нулями равномерно слева/ справа или вверх / вниз от входных данных. Когда padding="same" и strides=1, выходные данные имеют тот же размер, что и входные.

input\_shape = (28,28,1) - 4+D тензор с формой: match\_shape + (channels, rows, cols), если data\_format='channels\_first' или 4+D тензор с формой: match\_shape + (rows, cols, channels), если data\_format='channels\_last'

model.add(MaxPool2D(pool\_size=(2,2)))

- уменьшает выборку входных данных по их пространственным размерам (высоте и ширине), беря максимальное значение для окна ввода (размера, определенного параметром pool\_size) для каждого канала ввода. Окно сдвигается на шаг вдоль каждого измерения.

pool\_size=(2,2) - кортеж из 2 целых чисел, размер окна, для которого следует принимать максимальное значение.

model.add(Dropout(0.25))

- Dropout layer случайным образом устанавливает входные единицы в 0 с частотой rate на каждом шаге во время обучения, что помогает предотвратить переобучение. Входные данные, для которых не установлено значение 0, масштабируются на 1/ (1 - rate) таким образом, что сумма по всем входным данным остается неизменной.

model.add(Flatten())

- реорганизует входные данные в одномерный массив.

***Свёрточная сеть, число нейронов:***

model.add(Conv2D(filters = 8, kernel\_size = (5,5),padding = 'valid', activation ='relu', input\_shape = (28,28,1)))

- 24\*24\*8 = 4608

model.add(MaxPool2D(pool\_size=(2,2)))

- 24\*24\*8 -> 12\*12\*8 = 1152

model.add(Conv2D(filters = 16, kernel\_size = (3,3),padding = 'valid', activation ='relu'))

- 10\*10\*16 = 1600

model.add(MaxPool2D(pool\_size=(2,2), strides=(2,2)))

- 10\*10\*16 -> 5\*5\*16 = 400

model.add(Dense(10, activation = "softmax"))