Сверточные нейросети

С.И.Хашин
http://math.ivanovo.ac.ru/dalgebra/Khashin/index.html

Ивановский государственный университет

.

Иваново-2023

План

Определение

Python

Свёрточный нейрон

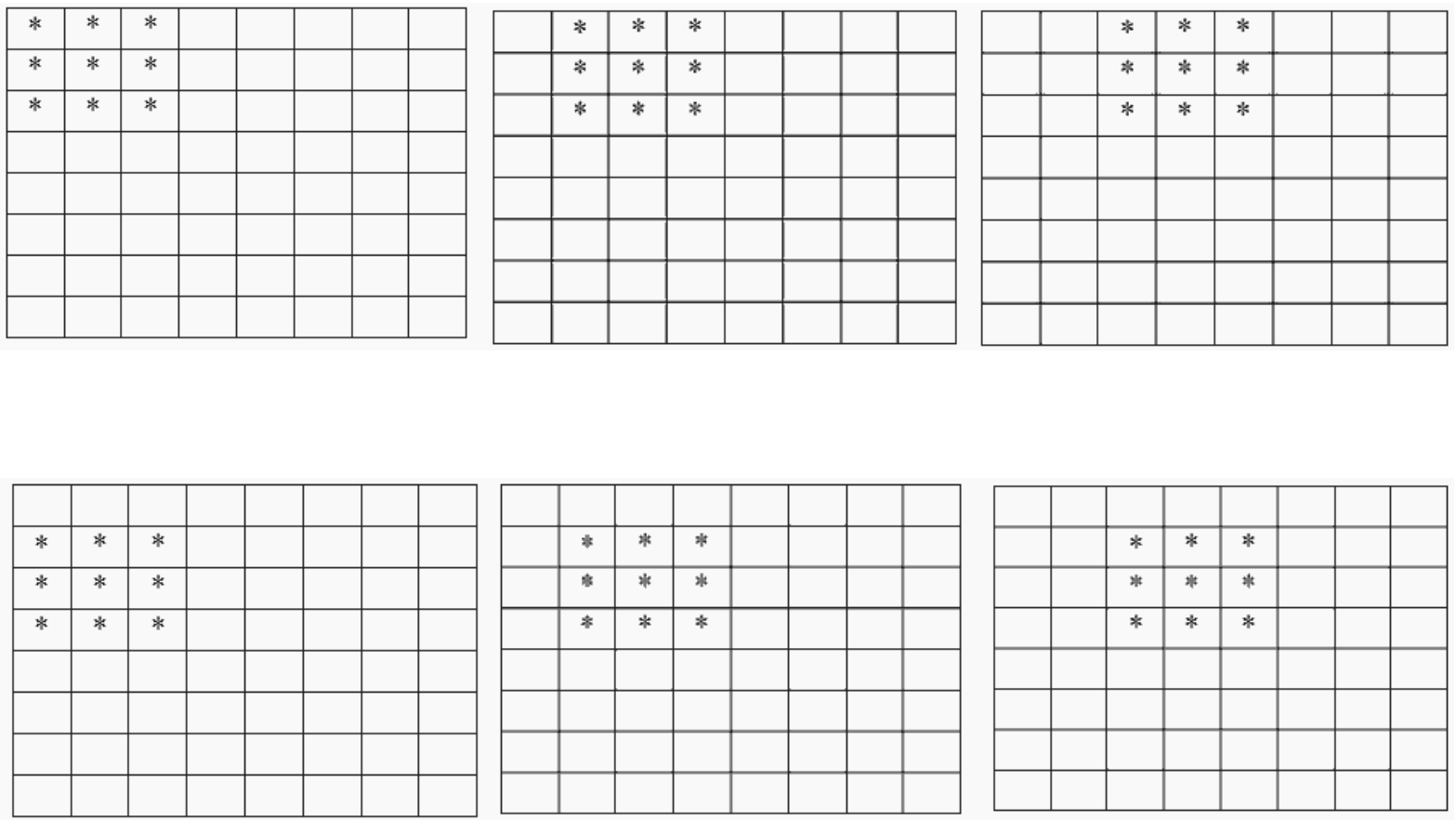
В свёрточной нейронной сети (convolutional neural network, CNN) обязательно есть слой свёрточных нейронов.
Рассмотрим в начале ОДИН свёрточный нейрон размера 3*3 для задачи MNIST, его коэффициенты

$$W = \begin{array}{|c|c|c|} \hline w_{00} & w_{01} & w_{02} \\ \hline w_{10} & w_{11} & w_{12} \\ \hline w_{20} & w_{21} & w_{22} \\ \hline \end{array}$$

Как вы помните, в задаче MNIST каждый обучающий вектор является U целочисленной матрицей размера $28 * 28$.
Это и есть коэффициенты свёрточного нейрона.
Мы строим свёртку матрицы U и маски W: $W * U$.
Обычно к результату этого преобразования применяется некоторая функция активации, например ReLU, Sigmoid, tanh или какая-то иная.

Свёрточный нейрон

Накладываем матрицу (маску) W на исходную матрицу размера $28 * 28$:



Свёрточный слой нейронов

Матрица 3*3 укладывается в изображение размера 28*28 по горизонтали 26 раз и сколько же по вертикали.
Результатом свёртки одного свёрточного нейрона с одним обучающим изображением будем матрица размера 26*26.
Если мы возьмем K свёрточных нейронов размера 3*3, то результатом их применения к исходной картинке 28*28 будет K матриц размера 26*26, или один массив (тензор) размера K*26*26.
У каждого свёрточного нейрона имеется 9 параметров (коэффициентов), всего 9K параметров у всего свёрточного слоя.

$$(70\,000 * 28 * 28) \rightarrow (70\,000 * K * 26 * 26)$$

Классическая архитектура CNN:

$$Input \rightarrow Conv \rightarrow ReLU \rightarrow Conv \rightarrow RuLU \rightarrow Pool \rightarrow FullyConneted$$

Свёрточный нейрон CIFAR10

Отличие от MNIST: входные данные имеют размерность (32*32*3). Поэтому свёрточный нейрон имеет 3*3*3=27 коэффициентов:

R			G			B		
w_{000}	w_{001}	w_{002}	w_{100}	w_{101}	w_{102}	w_{200}	w_{201}	w_{202}
w_{010}	w_{011}	w_{012}	w_{110}	w_{111}	w_{112}	w_{210}	w_{211}	w_{212}
w_{020}	w_{021}	w_{022}	w_{120}	w_{121}	w_{122}	w_{220}	w_{221}	w_{222}

Свёрточная сеть из K нейронов размера 3*3*3, будет иметь результатом K матриц размера 30*30, или тензор размера K*30*30.

$$(60\,000 * 3 * 32 * 32) \rightarrow (60\,000 * K * 30 * 30)$$

Python, MNIST, Подготовка данных

Подготовка данных

```
data = np.load("mnist_std.npz")
Y,X = data['Y'], data['X'].astype(float)
X /= 255
X = X.reshape((-1,28,28,1))

dolya = 60000  # разбивка на train/test
Y_train, X_train = Y[:dolya],  X[:dolya]
Y_test, X_test = Y[dolya:],  X[dolya:]

num_classes = 10
input_shape = (28, 28,1)
```

model

Построение модели

```
N_conv = 32  # нейронов в свёрточном слое
model = tf.keras.Sequential(
[
  tf.keras.Input(shape=input_shape),
  tf.keras.layers.Conv2D(N_conv, kernel_size=(3, 3),
                         activation="relu"),
  tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
  tf.keras.layers.Conv2D(64, kernel_size=(3, 3),
                         activation="relu"),
  tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dropout(0.5),
  tf.keras.layers.Dense(num_classes, activation="softmax")
] )
print('model:', model.summary())
```

model.summary()

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320
max_pooling2d (MaxPooling2D) (None, 13, 13, 32)        0
conv2d_1 (Conv2D)            (None, 11, 11, 64)        18496
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 64)          0
flatten (Flatten)            (None, 1600)              0
dropout (Dropout)            (None, 1600)              0
dense (Dense)                (None, 10)                16010
=================================================================
Total params: 34,826
```

Обучение

```
nEpochs = 10  # количество эпох
batch_size = 1000
tm = time.time()
model.fit(x=X_train, y=Y_train, epochs=nEpochs,
  batch_size=batch_size,
  validation_data=(X_test, Y_test))
tm = (time.time() - tm) / nEpochs
test_loss, test_acc = model.evaluate(X_test, Y_test)
print(
f'\nТочность:{test_acc:6.4f}, время:{tm:5.2f} сек./эпоху')
```

Результаты

```
Epoch 9/10
60/60 acc: 0.9783 val_acc: 0.9860
Epoch 10/10
60/60 acc: 0.9799 val_acc: 0.9859
```