Wprowadzenie do sieci konwolucyjnych

Marcin Wierzbiński

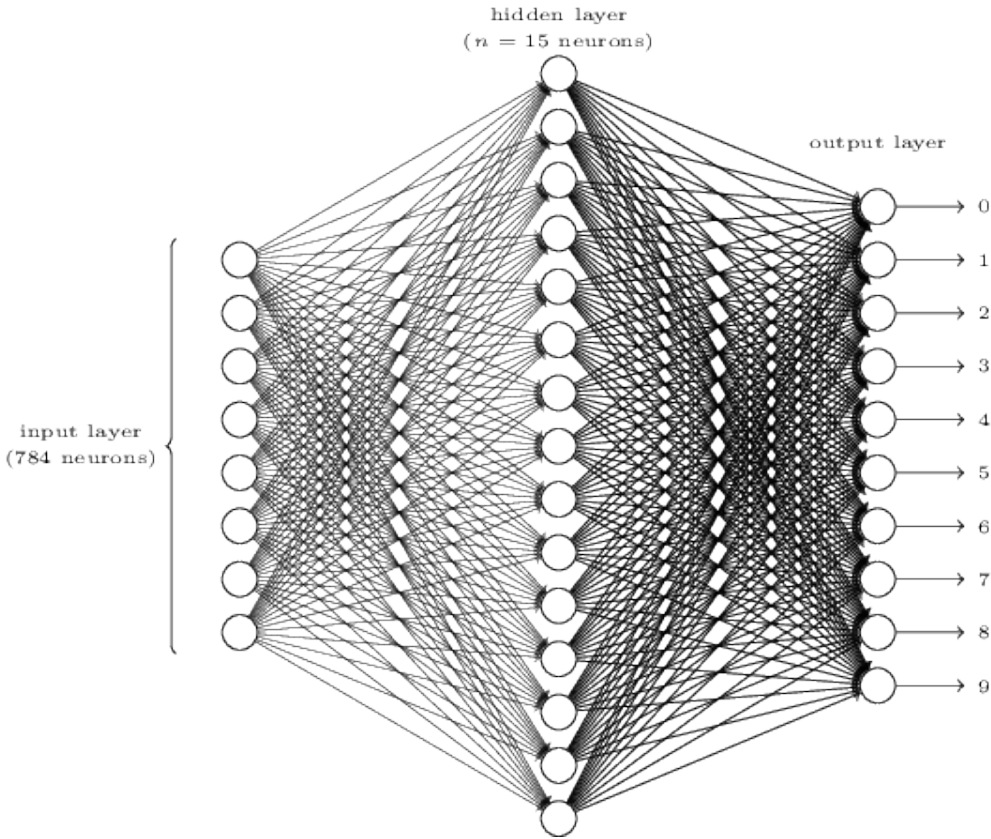
Zadanie klasyfikacji

1	1	0
4	2	1
0	2	1

.reshape(28*28)

1
1
0
4
2
1
0
2
1

Spłaszczenie macierzy obrazu 3x3 na wektor 9x1



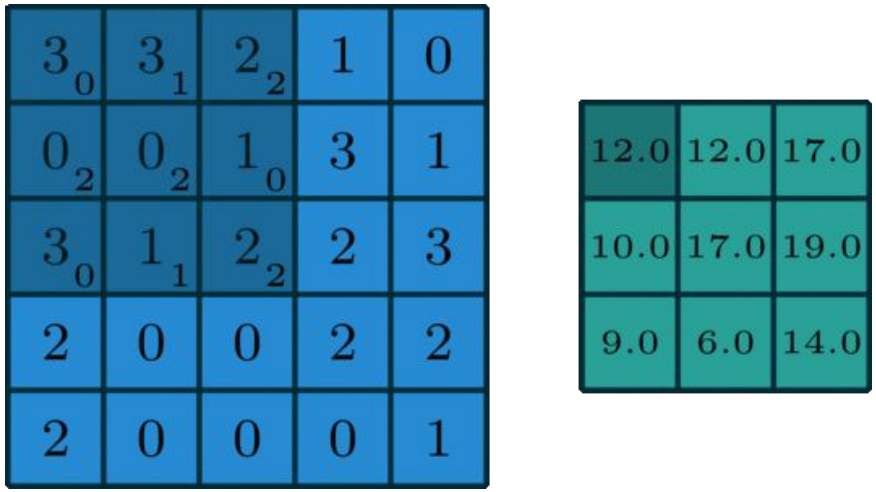
Sieci konwolucyjne

- Sieć konwolucyjna (CNN) to rodzaj sieci neuronowej, która jest często stosowana do analizy i przetwarzania obrazów.
- Sieci konwolucyjne składają się z kilku warstw przetwarzających, w tym warstw konwolucyjnych, warstw łączących (pooling), warstw normalizacyjnych i warstw w pełni połączonych. Każda z tych warstw wykonuje różne operacje, które przekształcają dane wejściowe w reprezentacje bardziej złożonych cech.

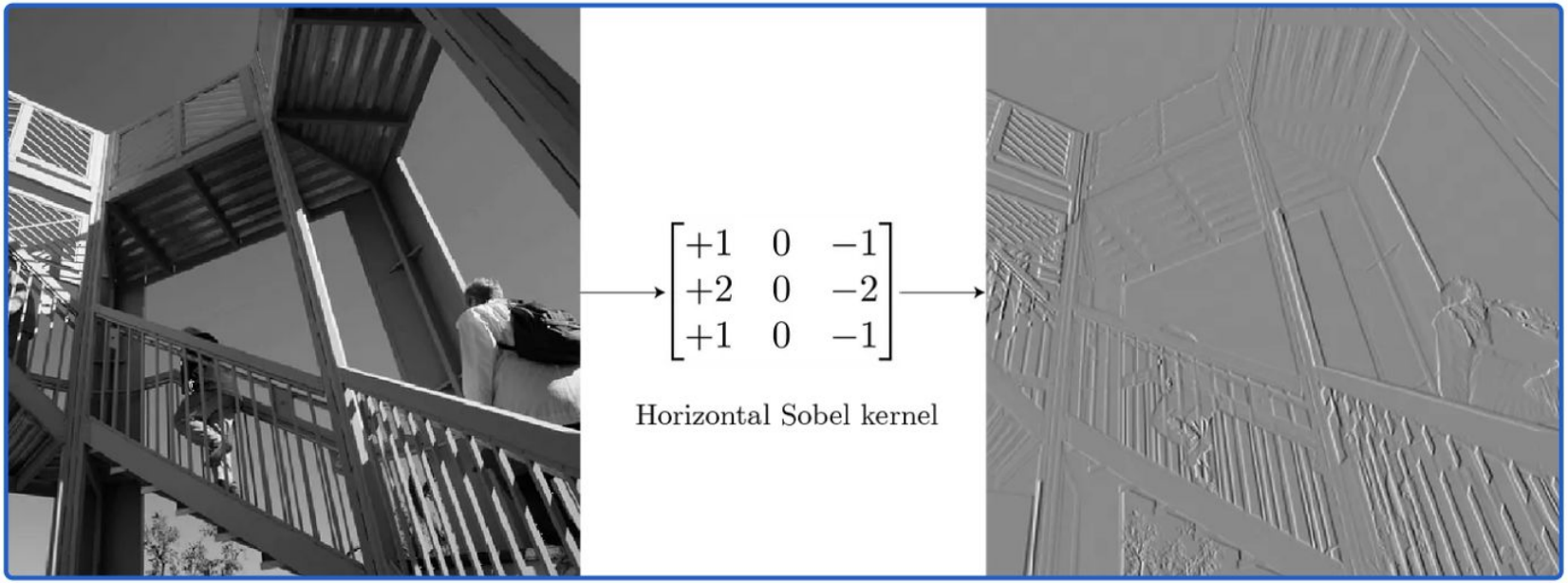
Warstwa konwolucyjna - filtry

- Warstwa składa się z zestawu **filtrów**, które są stosowane na wejściowej macierzy obrazu lub cech, aby wyodrębnić z niej cechy.
- Każdy **filtr** składa się z małego kwadratowego okna (zwykle 3x3 lub 5x5 pikseli), które poruszane jest po całym wejściowym obrazie lub mapie cech, a wynikowe wartości (tj. suma iloczynów pikseli w oknie i wag filtru) są zapisywane w odpowiednim miejscu w mapie wynikowej.
- Operacja konwolucji pozwala na wykrycie cech, takich jak krawędzie, narożniki, tekstury itp. w obrazie lub mapie cech. Dzięki wielu **filtrom**, każdy z których jest w stanie wykryć różne cechy, warstwa konwolucyjna może automatycznie uczyć się reprezentacji złożonych wzorców, co jest trudne do osiągnięcia w przypadku standardowych sieci neuronowych.

Filtr dla sieci konwolucyjnej



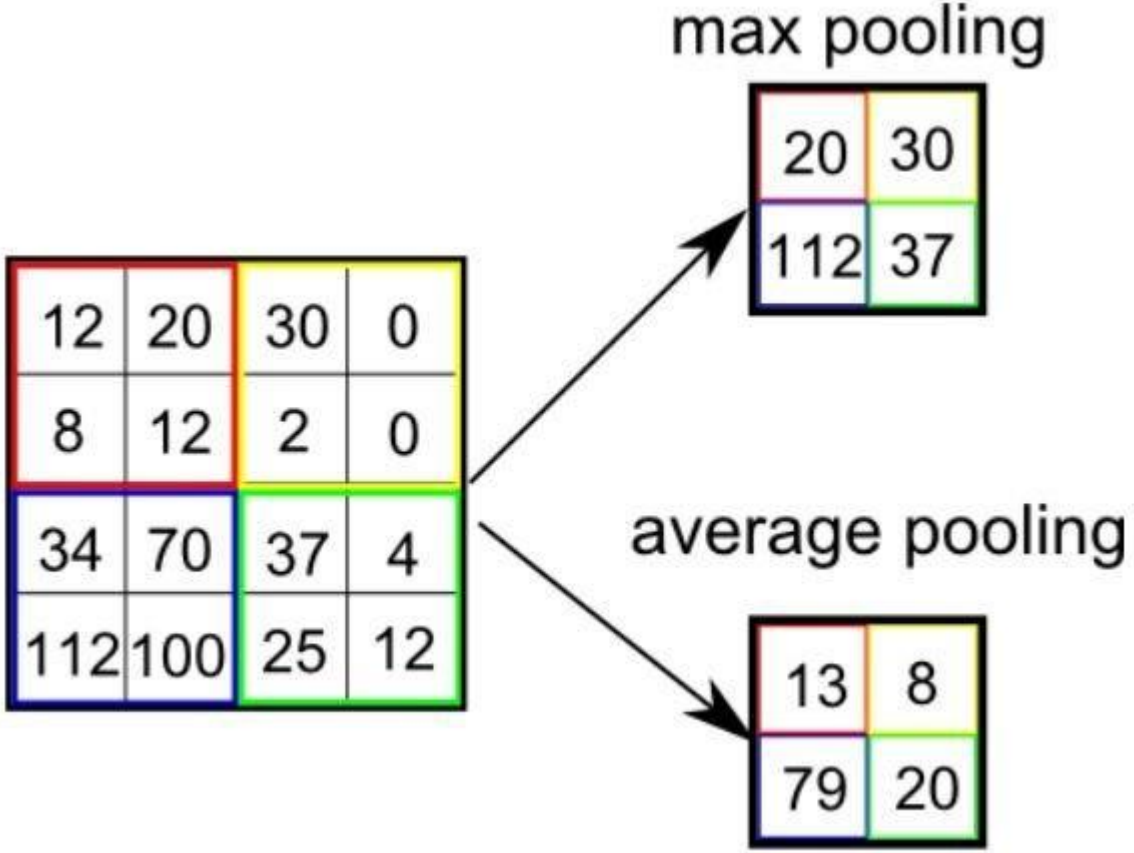
Kernel


$\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}$
Horizontal Sobel kernel

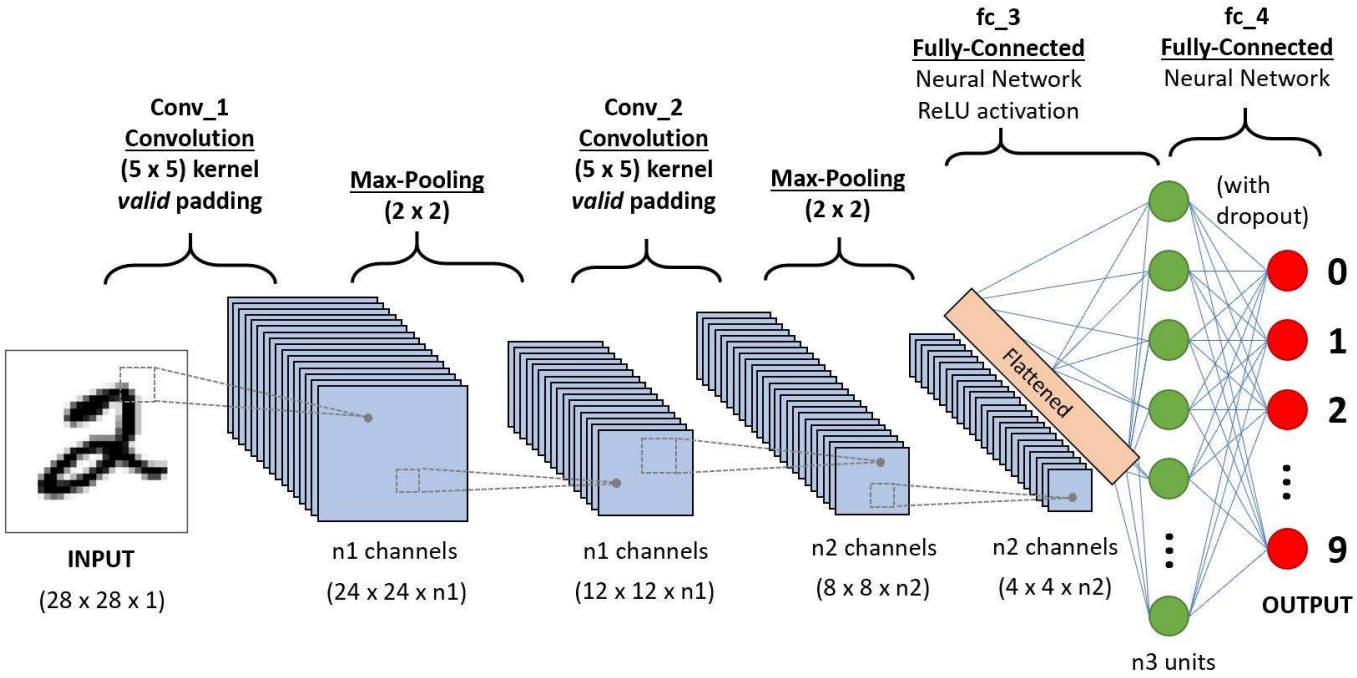
Max pooling

- Warstwa Pooling jest odpowiedzialna za redukcję rozmiaru przestrzennego Convolved Feature.
- Ma to na celu zmniejszenie mocy obliczeniowej potrzebnej do przetworzenia danych poprzez redukcję wymiarowości.
- Jest ona przydatna do wyodrębnienia dominujących cech, które są niezmienne pod względem rotacyjnym i pozycyjnym, co pozwala zachować proces efektywnego trenowania modelu.

Max pooling


max pooling
20 30
112 37
12 20 30 0
8 12 2 0
34 70 37 4
112 100 25 12
average pooling
13 8
79 20

Przykładowa architektura dla MNISTa



Powody skuteczności:

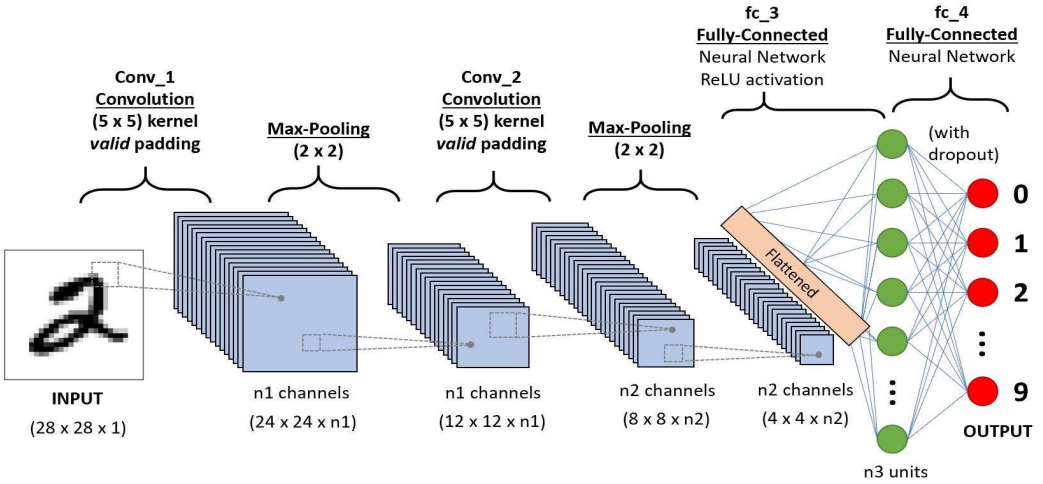
- Sieć konwolucyjna wykorzystuje specjalne warstwy konwolucyjne, które umożliwiają wykrywanie lokalnych wzorców w obrazach, takich jak krawędzie, narożniki, tekstury itp.
- Sieć konwolucyjna wykorzystuje także warstwy łączące, które zmniejszają wymiary map cech i zapewniają inwariantność na poziomie przestrzennym.
- Architektura sieci konwolucyjnej jest często bardziej efektywna niż standardowej sieci neuronowej, ponieważ wykorzystuje mniejszą liczbę parametrów.
- Sieć konwolucyjna ma kilka warstw konwolucyjnych i warstw łączących, które pozwalają na współdzielenie parametrów między nimi.

Sieć konwolucyjna w Kerasie

```
import numpy as np
from keras.datasets import mnist
from keras.utils import np_utils
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
from keras.optimizers import SGD

# wczytanie danych
(X_train, y_train), (X_test, y_test) = mnist.load_data()

# skalowanie danych
X_train = X_train.astype('float32') / 255
X_test = X_test.astype('float32') / 255

# konwersja etykiet na wektory one-hot
y_train = np_utils.to_categorical(y_train,10)
y_test = np_utils.to_categorical(y_test,10)

# dodanie kanału kolorów
X_train = np.expand_dims(X_train, axis=3)
X_test = np.expand_dims(X_test, axis=3)
```

Stworzenie modelu sieci konwolucyjnej

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28,28,1)))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))
```



Kompilacja modelu

```
# kompilacja modelu
model.compile(loss='categorical_crossentropy', optimizer=SGD(learning_rate=0.01), metrics=['accuracy'])

# trening modelu
model.fit(X_train, y_train, batch_size=32, epochs=1, verbose=1)

# ocena modelu na zbiorze testowym
score = model.evaluate(X_test, y_test, verbose=0)

# wypisanie wyniku
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

Resources

- https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1