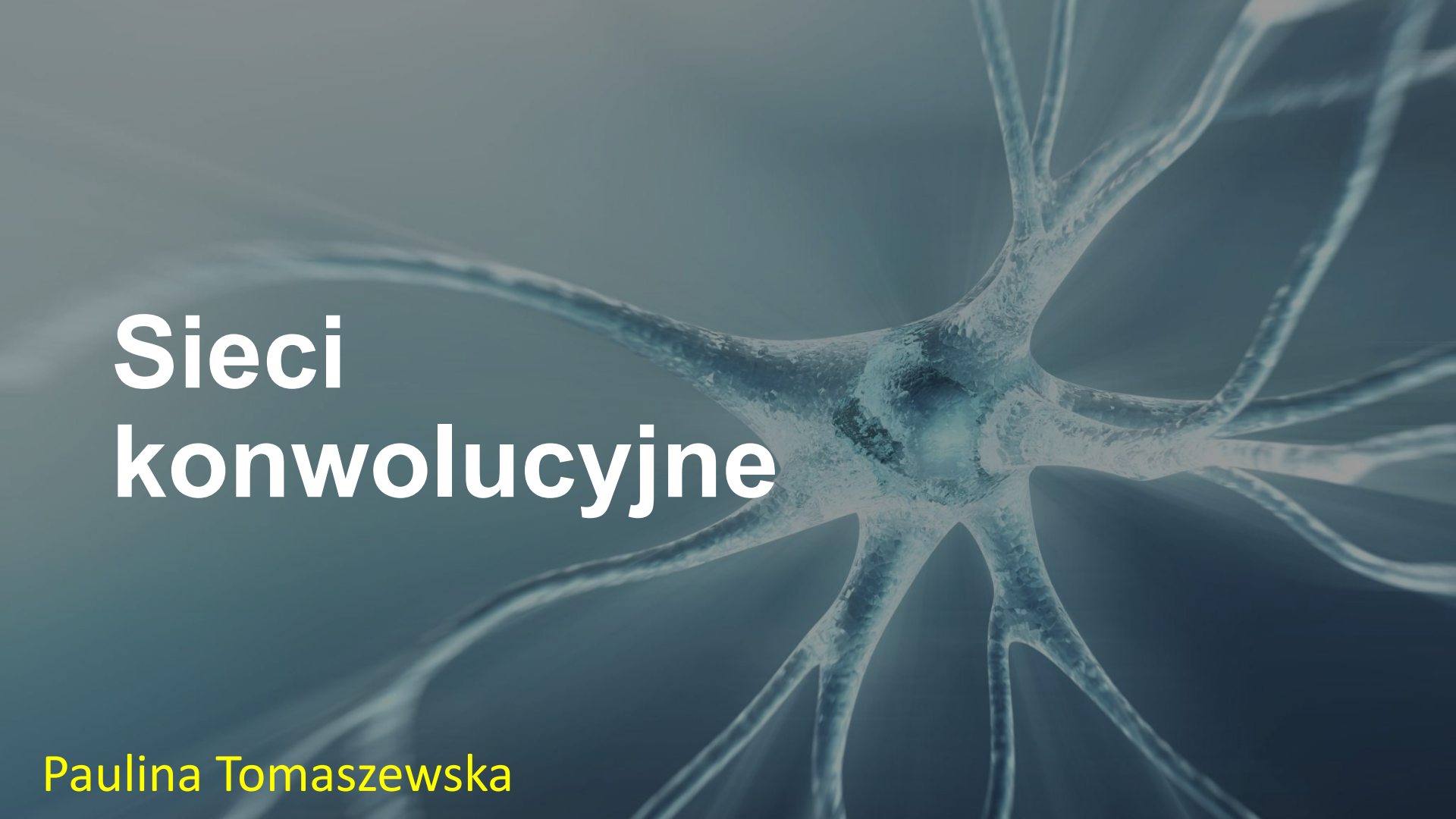


Sieci konwolucyjne

Paulina Tomaszewska

Uwaga:

W prezentacji są stosowane pewne uproszczenia i skróty myślowe, aby ułatwić intuicyjne zrozumienie treści.

Rozbudowane opisy zostały przygotowane z myślą o osobach, które nie uczestniczyły w zajęciach, aby umożliwić im samodzielne zapoznanie się z treściami.

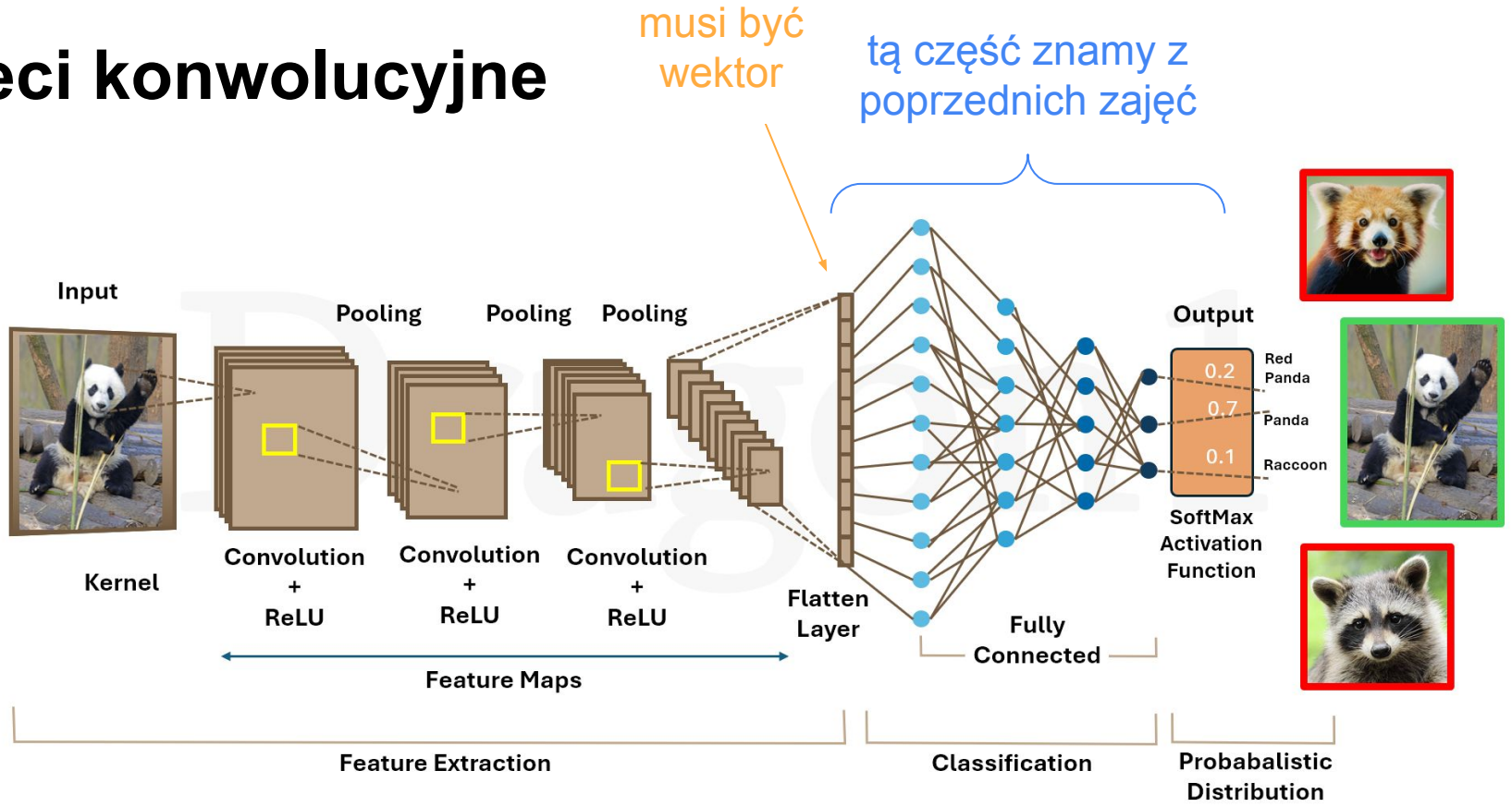
Już klasyfikowaliśmy obrazki używając sieci neuronowych. Czy był to optymalny sposób?

Nie!

- Przeanalizowaliśmy małe obrazy (28x28) w skali szarości, co daje 784 wartości.

  Ale wyobraźmy sobie, że mamy większe obrazy, powiedzmy 224x224 w RGB (3 kanały), co daje 150 528 po operacji spłaszczenia... w tym przypadku potrzebowalibyśmy 150 528 neuronów w pierwszej warstwie feed-forward, więc mielibyśmy dużą macierz wag W o rozmiarze (150 528x ...).
- Operacja spłaszczenia powoduje utratę aspektu przestrzennego obrazów.

Sieci konwolucyjne

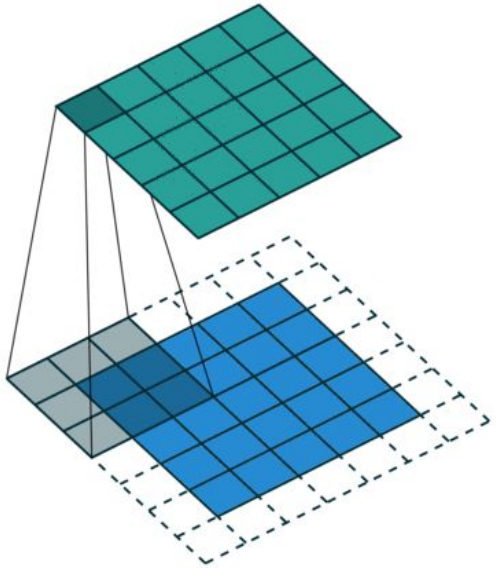


Podstawowym elementem sieci konwolucyjnych są **warstwy konwolucyjne**, w których wykorzystywane są **filtry (ang. *kernels*)**.

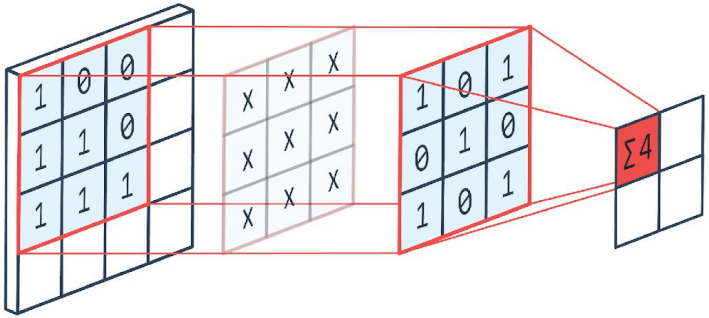
Filtry to małe macierze z trenowalnymi wagami.

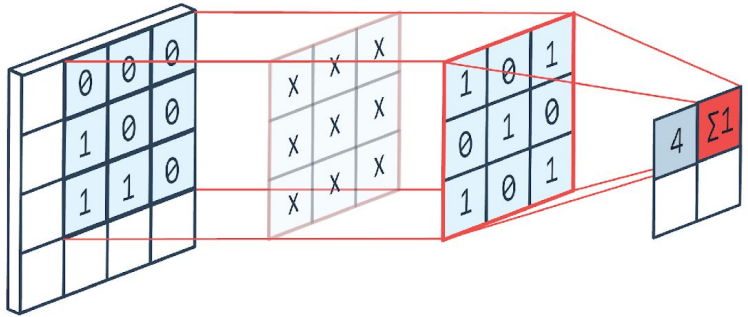
Jak działa warstwa konwolucyjna?

- Aplikujemy filtry do fragmentu danych wejściowych, dokonujemy operacji matematycznej i następnie przesuwamy (ang. *sliding window*).
- Operacja polega na tym, mnożymy każdy element danych wejściowych razy odpowiadający mu element w filtrze. Następnie wszystkie wyliczone wartości sumujemy i wpisujemy do wyjścia warstwy.



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

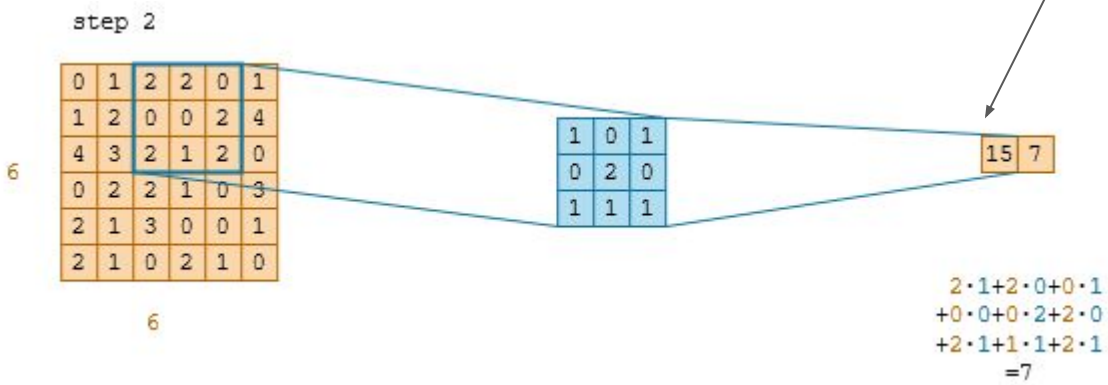




https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/blocks/2d-convolution-block

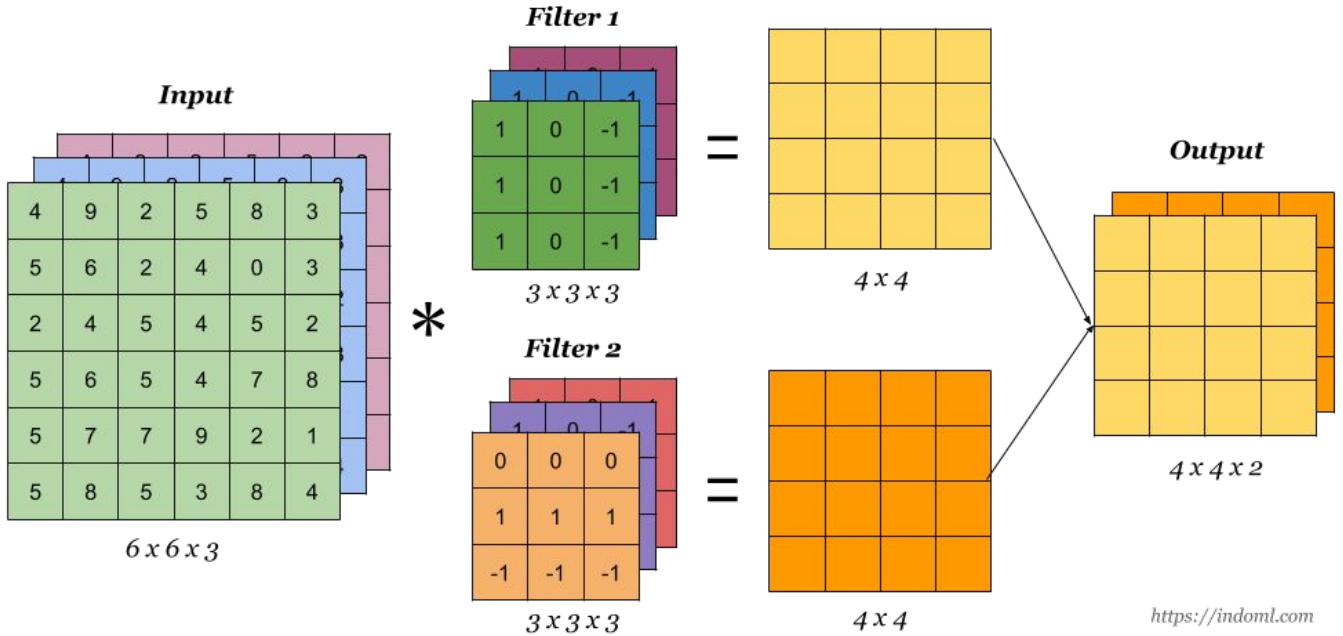
Spróbuj policzyć krok nr 1 algorytmu. Czy wyjdzie Ci wartość 15?

stride = 2



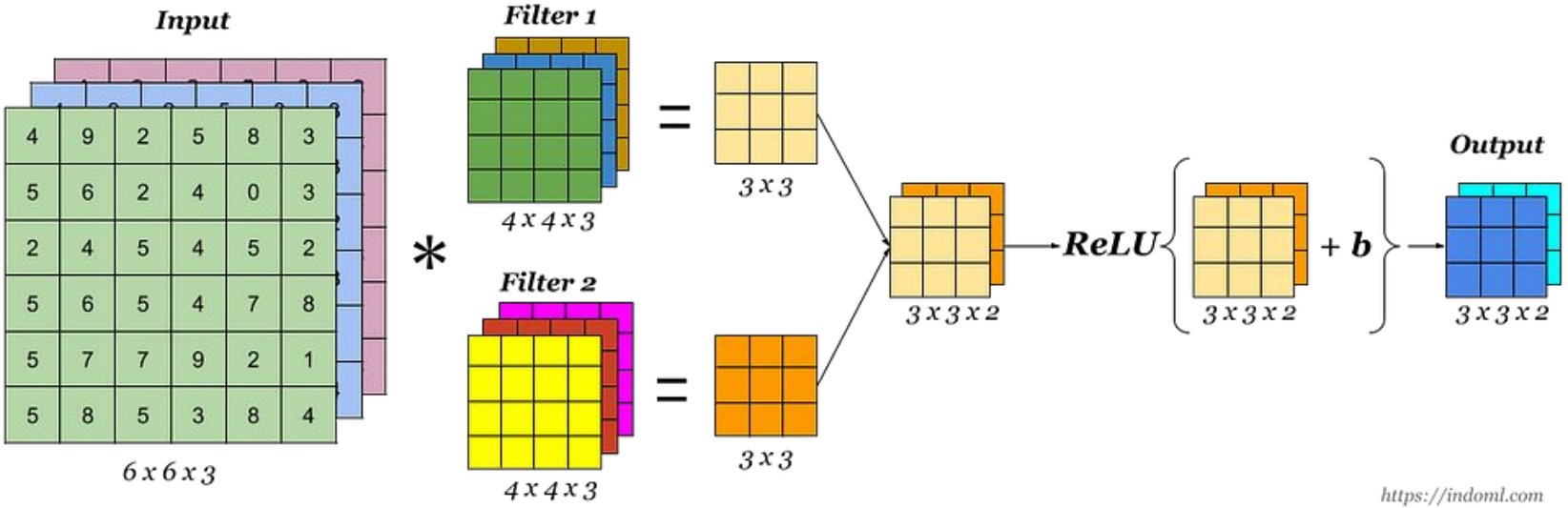
https://towardsdatascience.com/a-comprehensible-explanation-of-the-dimensions-in-cnns-841dba49df5e?gi=d70e6c1a4c9c

- Filtry mają tyle kanałów ile kanałów mają dane wejściowe.
- W jednej warstwie konwolucyjnej stosujemy wiele filtrów.
- Na wyjściu z warstwy jest tyle kanałów ile filtrów zostało zaaplikowanych.



https://towardsdatascience.com/beginners-guide-to-understanding-convolutional-neural-networks-ae9ed58bb17d

W warstwie konwolucyjnej uwzględniany bywa wyraz wolny. Wynik tej warstwy trafia do funkcji aktywacji (zazwyczaj ReLU).



https://towardsdatascience.com/beginners-guide-to-understanding-convolutional-neural-networks-ae9ed58bb17d

Padding

- dodaje obwódkę wokół danych wejściowych (najczęściej zera)

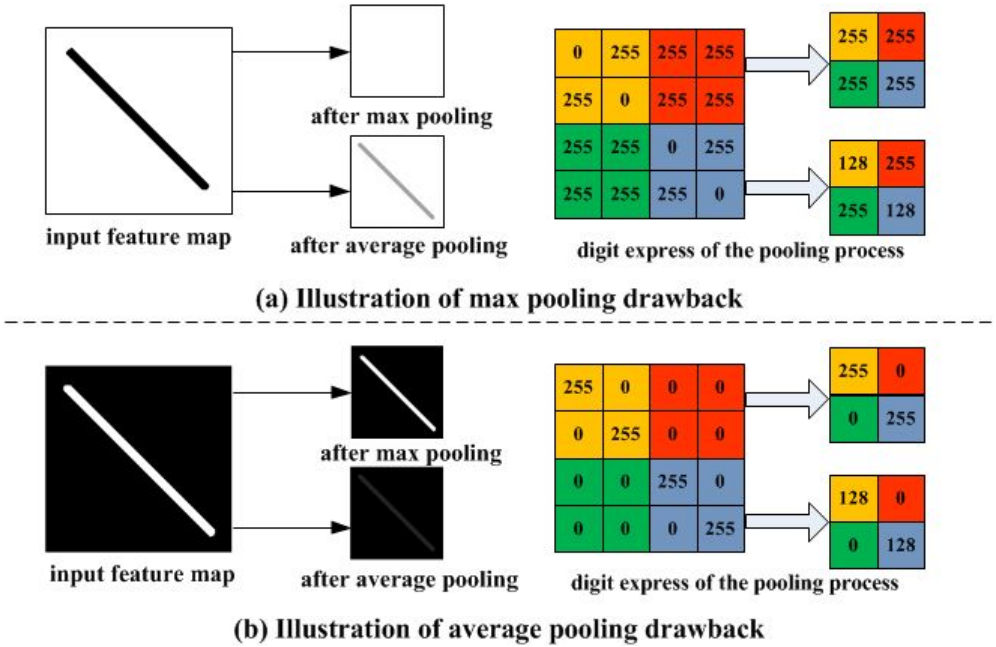
Rozmiar wyjścia z warstwy konwolucyjna:

$$\text{Output Size} = \frac{\text{Input Size} + 2 \times \text{Padding} - \text{Kernel Size}}{\text{Stride}} + 1$$

Parametry w warstwie konwolucyjnej

- **Rozmiar filtru (często 3x3, 5x5)**
- **Liczba filtrów**
- **Stride:** liczba pikseli o które przesuwamy filtr w kolejnych krokach
- **Padding:** szerokość dodatkowej obwódki, którą aplikujemy do danych

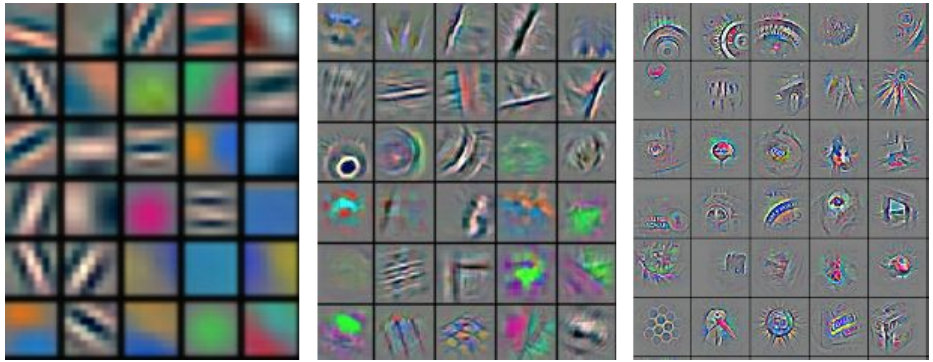
Pooling



- używany aby zmniejszyć rozmiar przestrzenny danych
- nie ma trenowanych parametrów
- parametr określa z obszaru o jakim rozmiarze wyliczamy wartości (średnia albo max.)
- często używany po warstwie konwolucyjnej

https://www.researchgate.net/figure/Toy-example-illustrating-the-drawbacks-of-max-pooling-and-average-pooling_fig2_30002

Uważa się, że sieci konwolucyjne działają w hierarchiczny sposób - czyli początkowo warstwy wyłapują proste kształty, a następnie coraz bardziej skomplikowane.



https://stackoverflow.com/questions/41598858/how-can-filters-later-in-a-cnn-recognize-color

Nie zawsze musimy trenować modele od zera

Nie zawsze musimy trenować modele od zera

Możemy zaadaptować modele pretrenowane

Lista pretrenowanych modeli do klasyfikacji obrazów w module torchvision

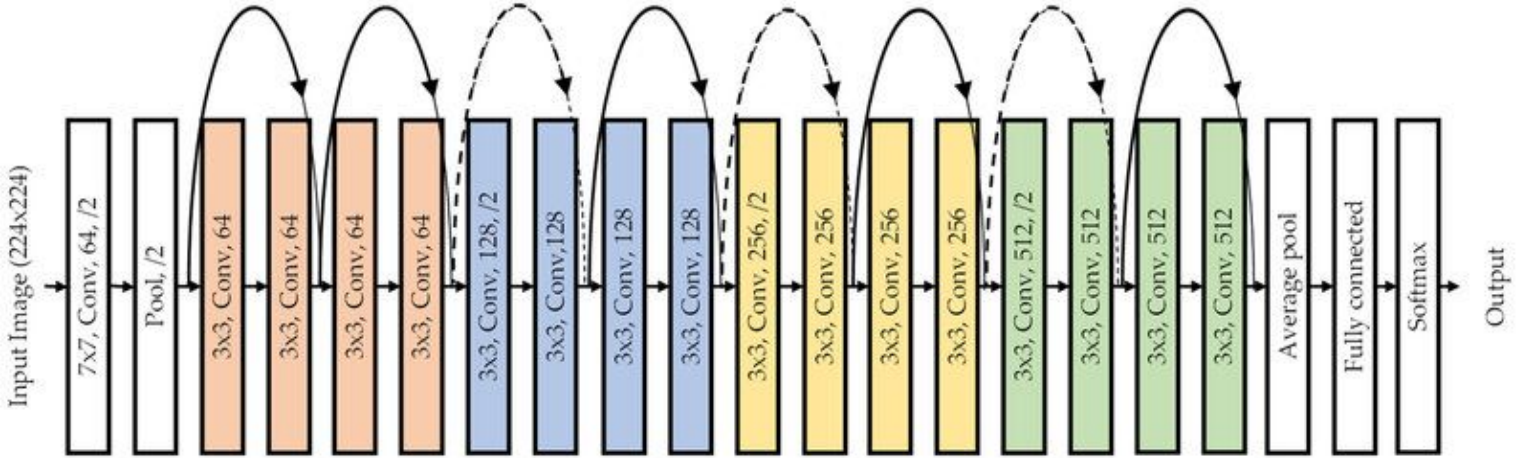
- AlexNet
- VGG
- ResNet
- SqueezeNet
- DenseNet
- Inception v3
- GoogLeNet
- ShuffleNet v2
- MobileNet v2
- ResNeXt
- Wide ResNet
- MNASNet

Modele są pretrenowane na bardzo duży zbiorze zdjęć zwanym Imagenet.

https://pytorch.org/vision/0.8/models.html

Resnet

- poprzez zastosowanie *residual connnections (*zwanych także *skip connections)* unika się problemu zanikającego gradientu, który występowałby w przypadku bardzo głębokich modeli



https://www.researchgate.net/figure/Structure-of-the-Resnet-18-Model_fig1_366608244

Różne wersje modeli

Liczba wskazuje na liczbę warstw konwolucyjnych i w pełni połączych. Warstwy pooling nie są trenowalne i dlatego nie są uwzględniane.

```
resnet18(*[, weights, progress])
```

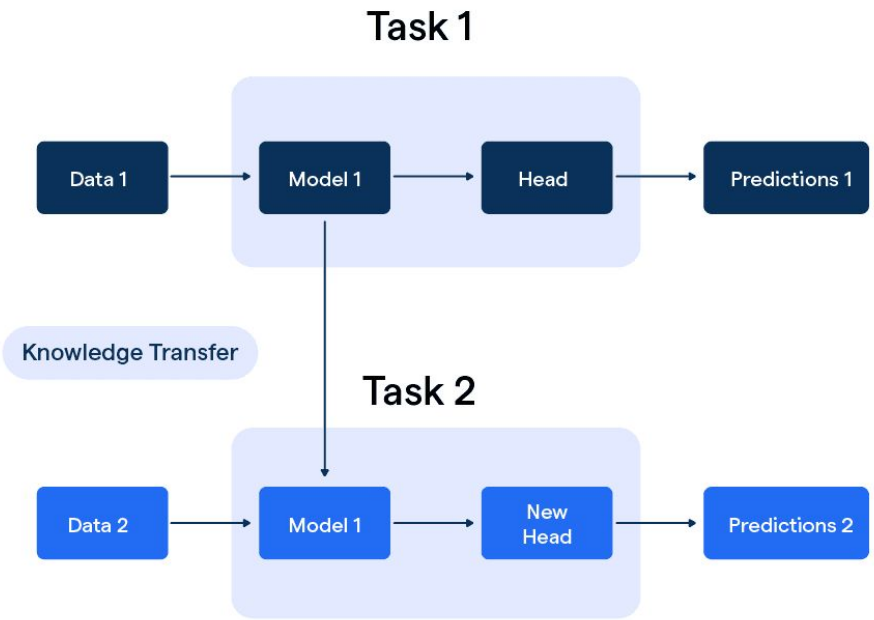
```
resnet34(*[, weights, progress])
```

```
resnet50(*[, weights, progress])
```

```
resnet101(*[, weights, progress])
```

```
resnet152(*[, weights, progress])
```

Transfer learning



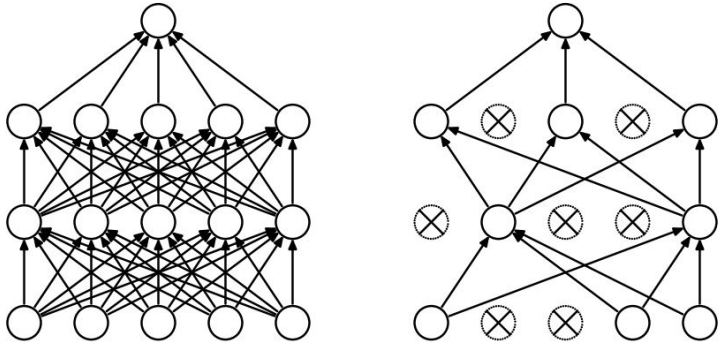
Najbardziej popularny scenariusz:

1. zamrażamy ekstraktor cech (nie aktualizujmey wag, bo nie rejestrujemy dla tych warstw gradientu)
2. inicjalizujemy warstwy w pełni połączone (zwane głową)
3. trenujemy model na danych docelowych

https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a?gi=f7cc6a12ff3a

Przydatne techniki, aby uniknąć przeuczenia modeli

- **Dropout**
  - wyłączamy podczas treningu losowe neurony z parametrem *p*
  - podczas ewaluacji wszystkie neurony są włączone
  - z uwagi na dwa różne tryby działania tej warstwy ważne jest, aby korzystać z komend model.train(), model.eval()



(a) Standard Neural Net (b) After applying dropout.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15, 1 (January 2014), 1929–1958.

Przydatne techniki, aby uniknąć przeuczenia modeli

- **Data augmentation**
  - zmniejszenie problemu małego etykietowanego zbioru treningowego poprzez dodanie próbek po transformacjach niezmieniających etykiet

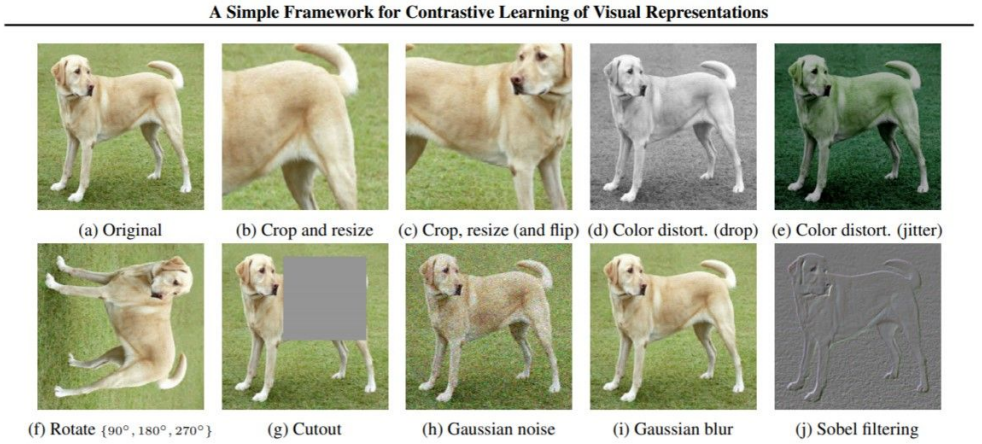


Figure 4. Illustrations of the studied data augmentation operators. Each augmentation can transform data stochastically with some internal parameters (e.g. rotation degree, noise level). Note that we *only* test these operators in ablation, the *augmentation policy used to train our models* only includes *random crop (with flip and resize)*, *color distortion*, and *Gaussian blur*. (Original image cc-by: Von.grzanka)

Chen, Ting, et al. "A simple framework for contrastive learning of visual representations." *International conference on machine learning*. 2020.