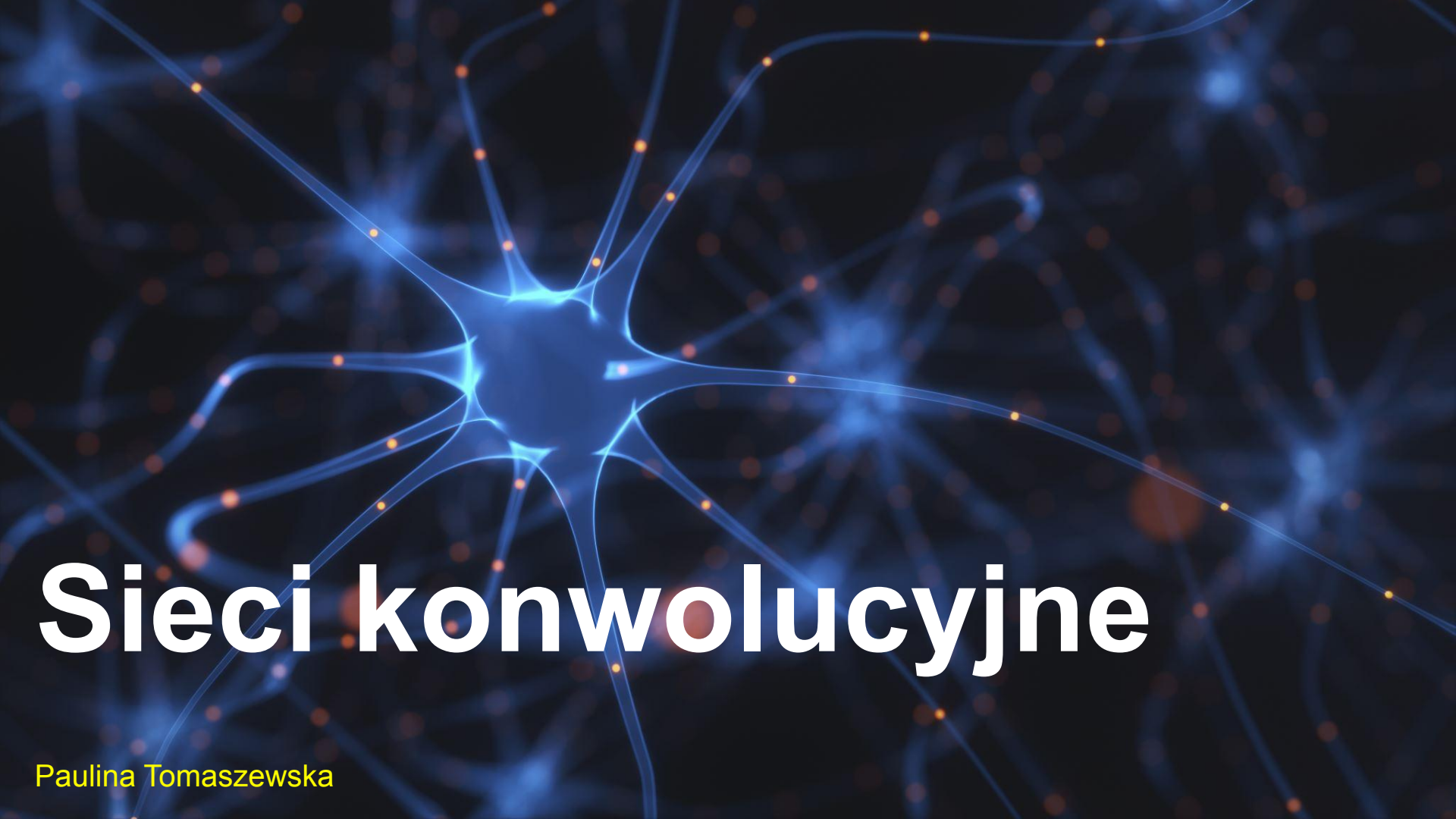
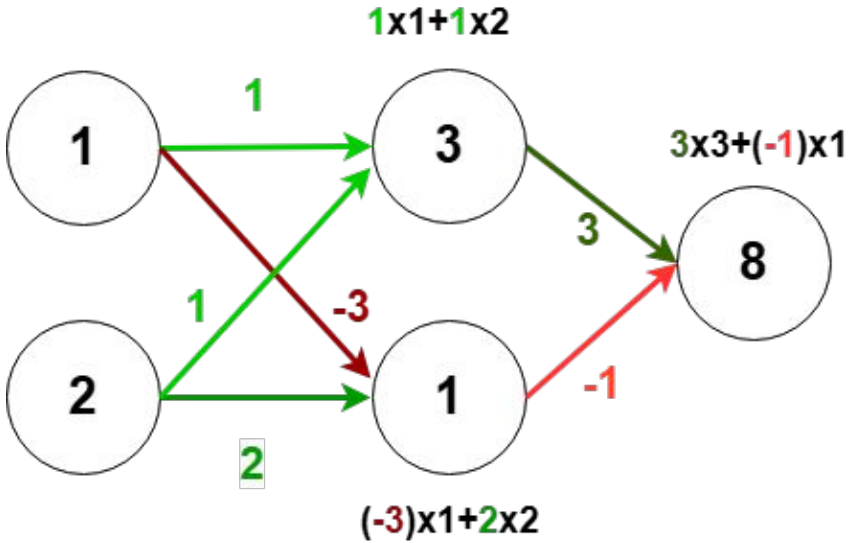


Sieci konwolucyjne

Paulina Tomaszewska

Jakie sieci neuronowe już znamy?

- sieci w pełni połączone (*fully-connected/ dense layers, multilayer perceptron, MLP*)
- <u>na ich wejściu są wektory</u>



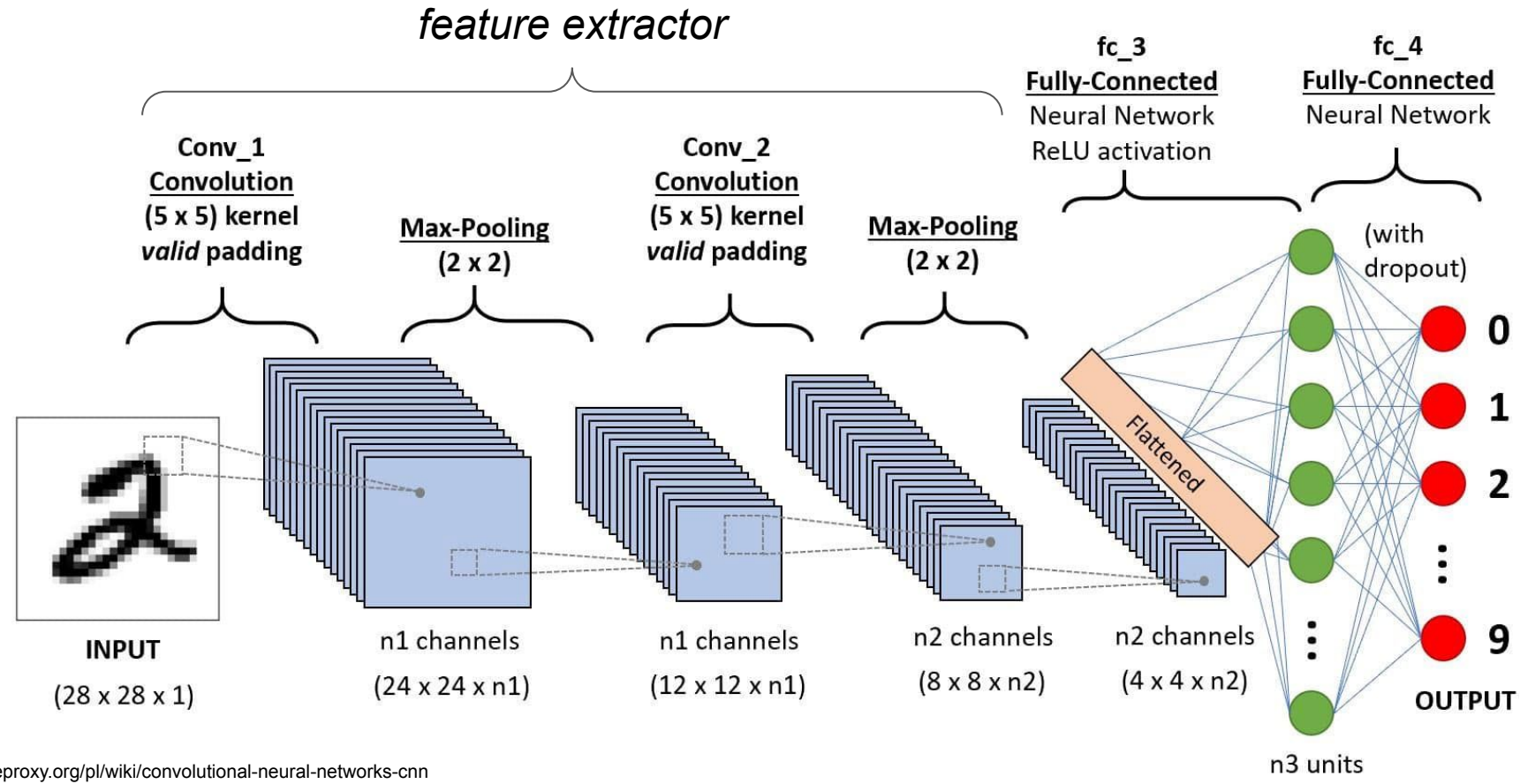
W prezentacji są stosowane pewne uproszczenia i skróty myślowe, aby pozwolić na intuicyjne zrozumienie treści.

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

Jak w takiej sytuacji analizować inne dane?

Dzisiaj skupimy się na danych obrazowych

Sieć konwolucyjna



https://fineproxy.org/pl/wiki/convolutional-neural-networks-cnn

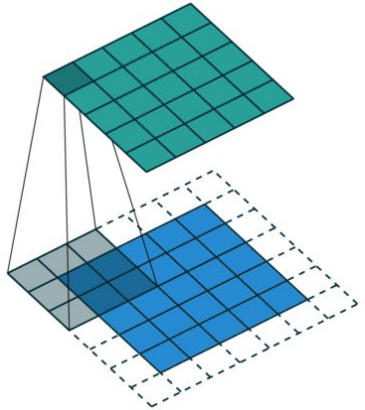
sieć konwolucyjna = ekstraktor cech + “głowa” (*fully-connected layers*)

składa się głównie z:

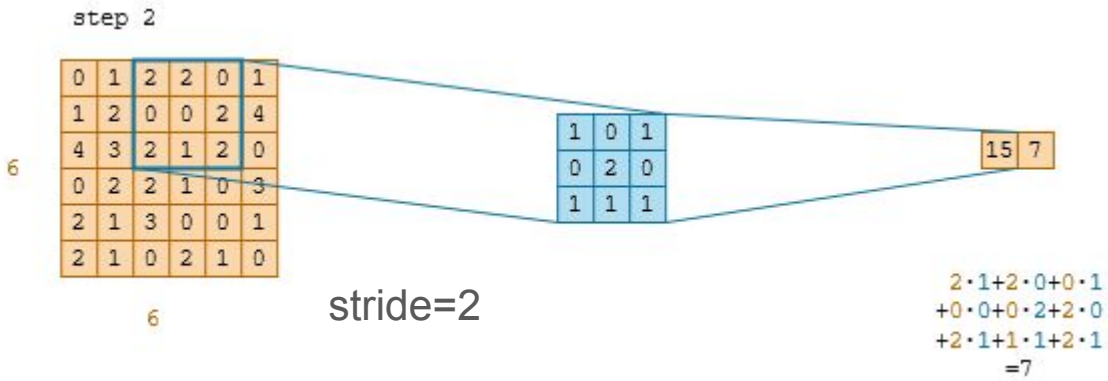
- warstw konwolucyjnych
- warstw pooling

Warstwa konwolucyjna

Przesuwamy *kernel* (małą macierz z wagami) wzdłuż obrazka, a gdy dojdziemy do granicy obrazka, przechodzimy do niższych wierszy. Gdy *kernel "leży"* na danym fragmencie obrazka dokonujemy mnożenia element po elemencie między wartościami z *kernel* i fragmentem obrazka. Na sam koniec wszystkie otrzymane wartości sumujemy i zapisujemy w macierzy wyjściowej.



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



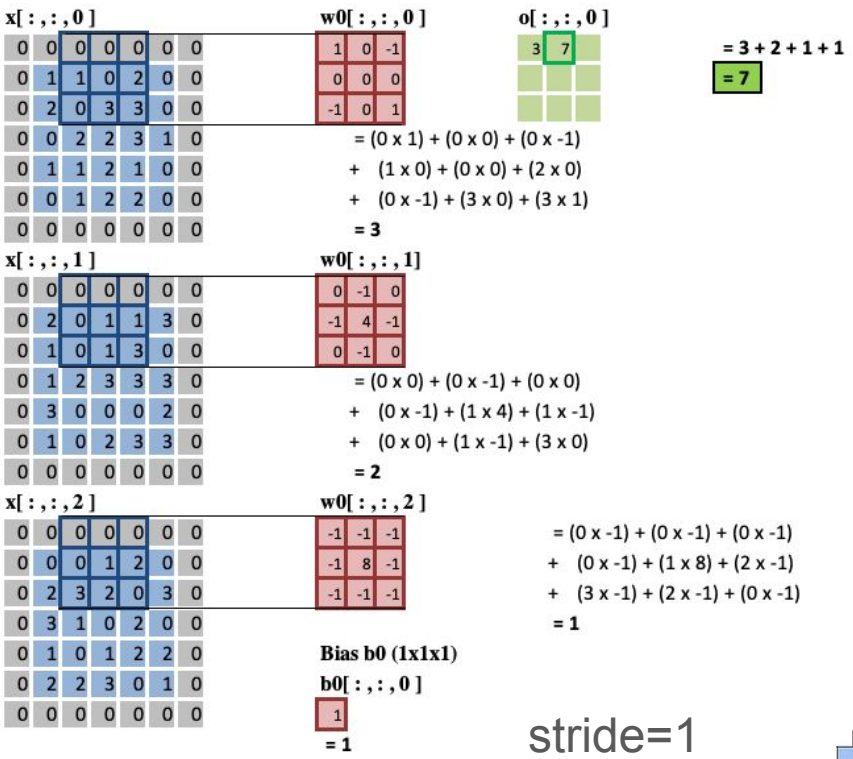
potem dodajemy jeszcze wyraz wolny do tej sumy i aplikujemy funkcję aktywacji → w efekcie otrzymujemy *feature map*

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

Najważniejsze parametry warstwy konwolucyjnej

- *kernel size* - rozmiar macierzy kernel (zazwyczaj *3x3*, *5x5*)
- *stride* - określa o ile “pikseli” przesuwamy *kernel*
- *padding size* - określa ile szerokości ma mieć “obwódka” wokół danych wejściowych
- *channel in* - liczba kanałów w danych wejściowych
- *channel out* - liczba kanałów wyjściowych

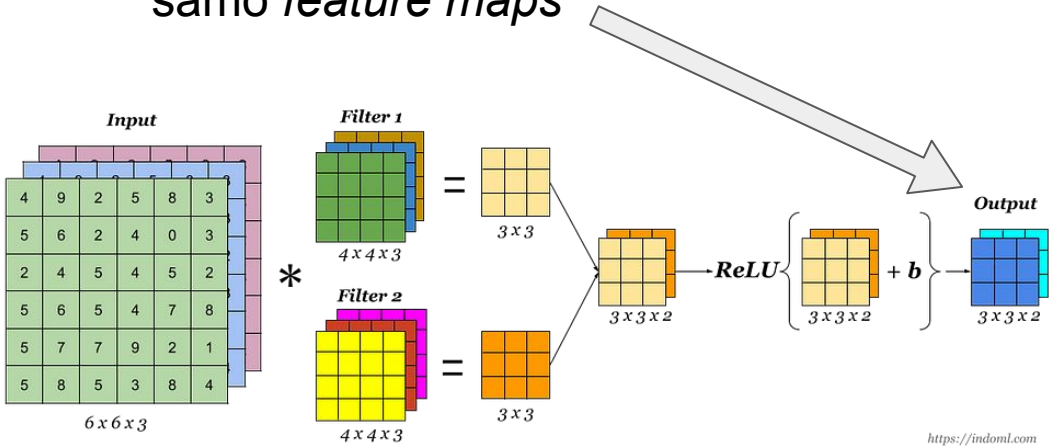
W ramach treningu będziemy aktualizować wagi w *kernel’ach*, które początkowo są losowe.



stride=1

Kernel

- warstwy konwolucyjne stosujemy do tensorów (czyli nie tylko wprost obrazów)
- gdy dane wejściowe do warstwy konwolucyjnej mają *n* kanałów to pojedynczy kernel też musi mieć *n* kanałów
- w jednej warstwie konwolucyjnej możemy zastosować więcej niż jeden kernel, wtedy na wyjściu będzie tyle samo *feature maps*

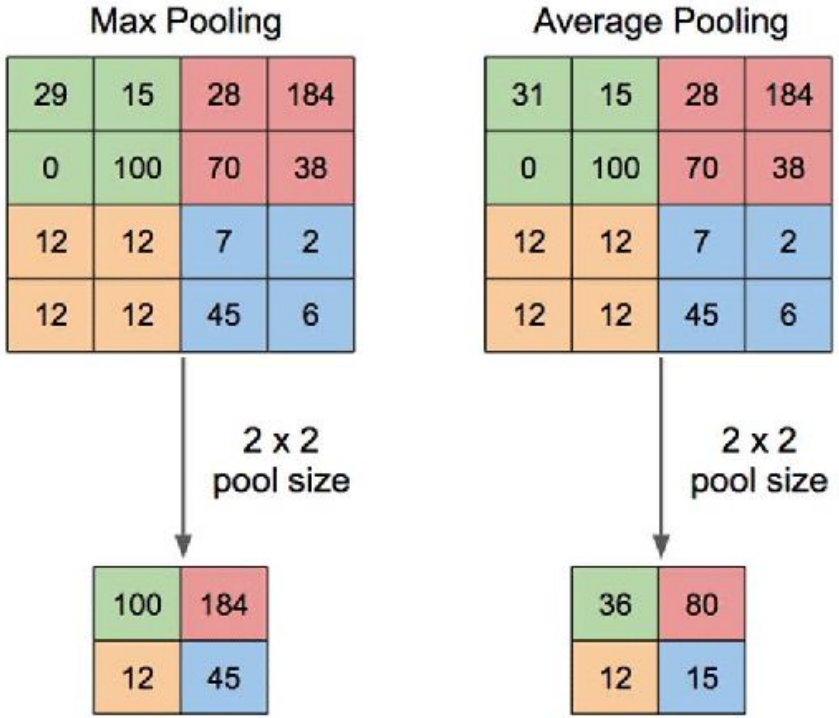


https://towardsdatascience.com/understanding-convolutions-by-hand-vs-tensorflow-8e64053f673e

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

Pooling

- warstwa podobna do konwolucyjnej, ale nie ma trenowalnych parametrów
- w ramach obszarów o zdefiniowanym rozmiarze wyliczana jest wartość średnia i maksymalna



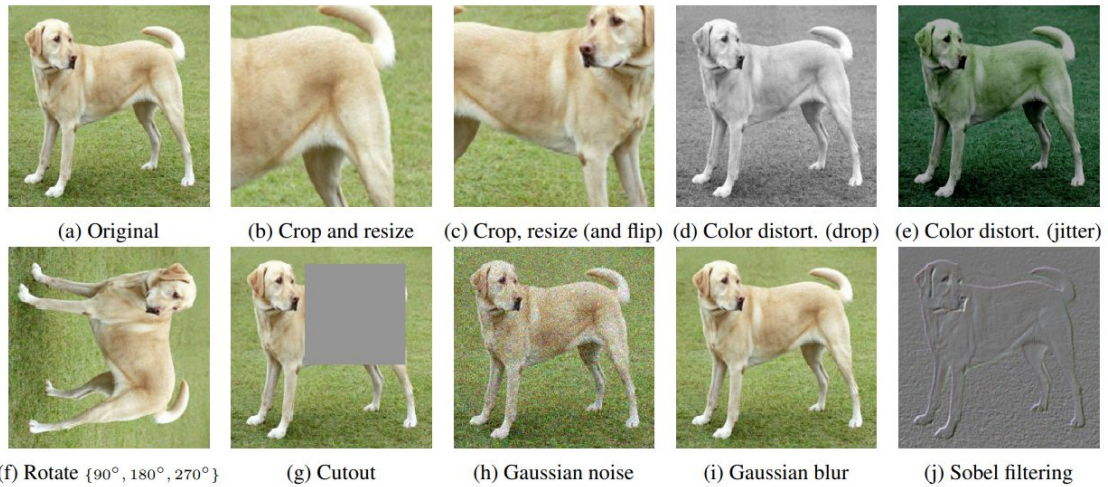
Yani, Muhamad & Irawan, S, & Setianingsih, Casi. (2019). Application of Transfer Learning Using Convolutional Neural Network Method for Early Detection of Terry's Nail. Journal of Physics: Conference Series.

Idea w sieciach konwolucyjnych

- w ekstraktorze cech zazwyczaj zmniejszamy rozmiar przestrzenny → *feature maps* są coraz mniejsze ale jest ich coraz więcej z warstwy na warstwę
- na wyjściu ekstraktora cech chcemy otrzymać wektor - robimy to albo poprzez operację *flatten* albo *Global Average Pooling* (wylicza średnią z każdej *feature map'y* tworząc wektor)
- mając wyekstrahowane cechy (wektor) stosujemy warstwy w pełni połączone, aby dokonać np. klasyfikacji

Data augmentation

- jeżeli mamy za mało danych treningowych, możemy zastosować modyfikacje danych tak, aby nie zmieniły się ich etykiety



Chen, T., Kornblith, S., Norouzi, M., & Hinton, G.E. (2020). A Simple Framework for Contrastive Learning of Visual Representations. *ArXiv, abs/2002.05709*.