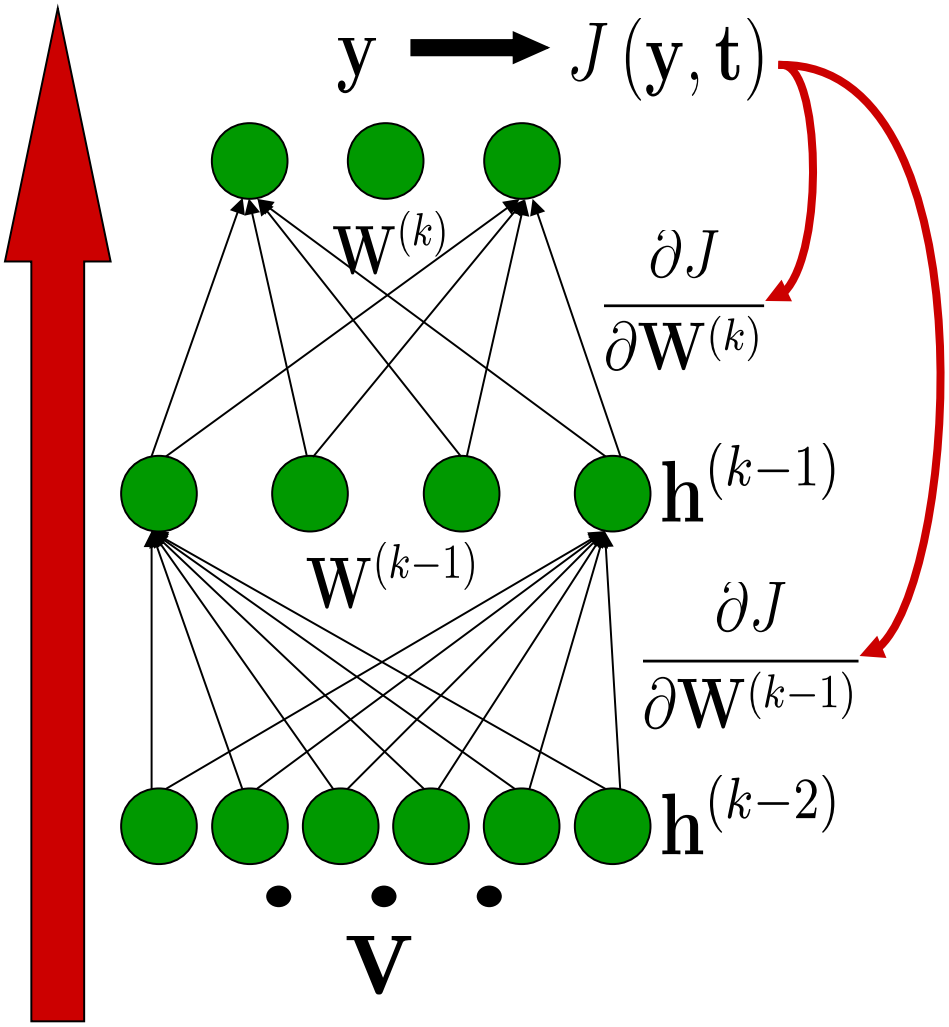
Sieci Neuronowe

Regresja logistyczna i regresja softmax

Regresja logistyczna



W zagadnieniu regresji logistycznej przyjmujemy, że obserwacje należą do dwóch klas:

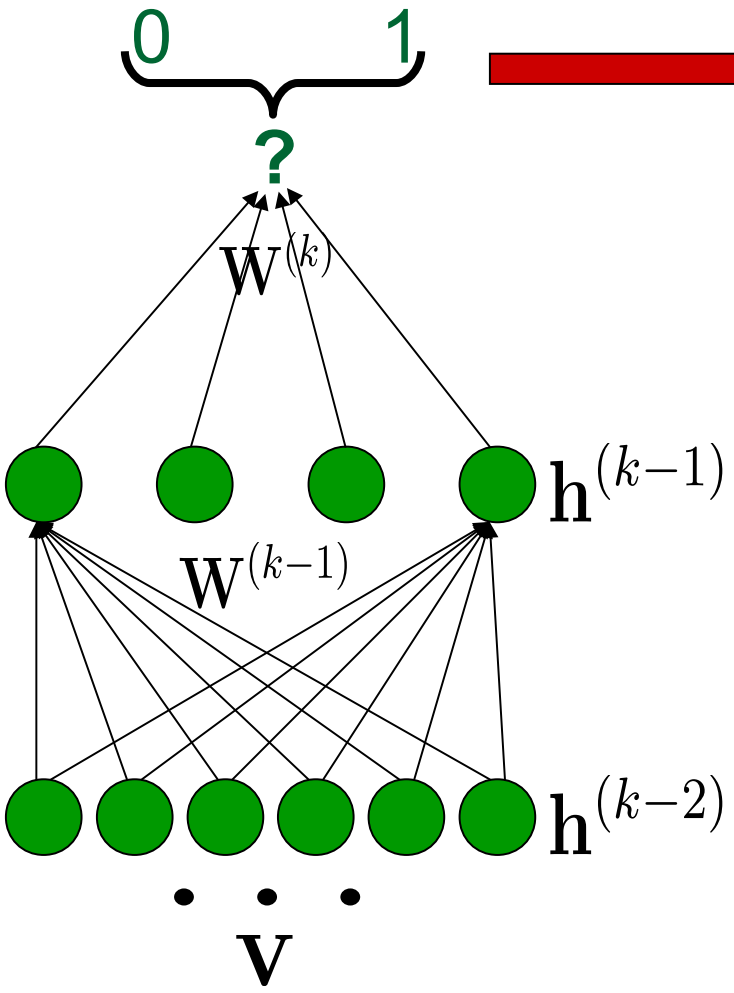
$$t_i \in \{0, 1\}$$

Np.:

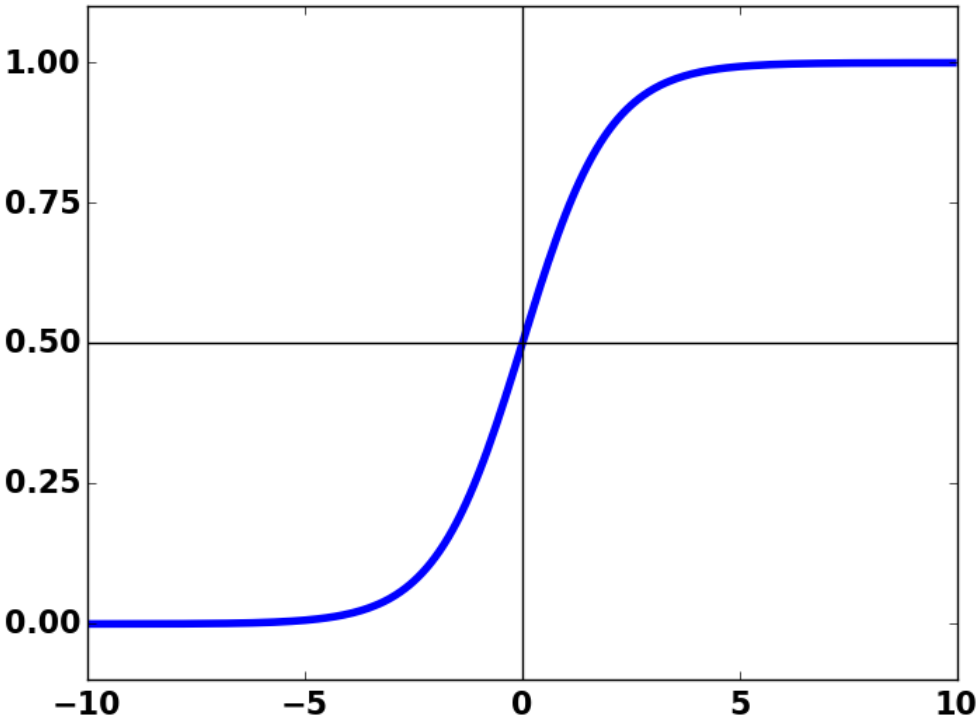
- spam (1) vs. nie spam (0).
- twarz (1) vs. nie twarz (0).

Celem uczenia jest znalezienie takich wag, by MLP potrafił przewidywać klasy dla zaprezentowanych mu nowych obserwacji.

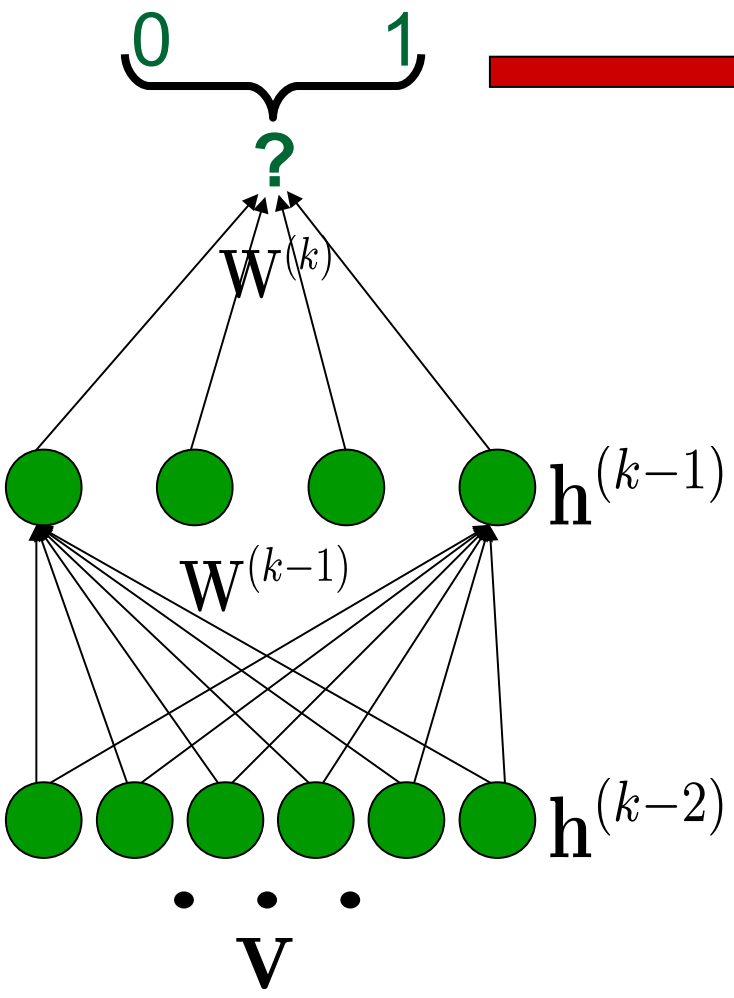
Regresja logistyczna



$$g(z) = \frac{1}{1 + \mathrm{e}^{-z}} \quad (= \sigma(z))$$



Regresja logistyczna

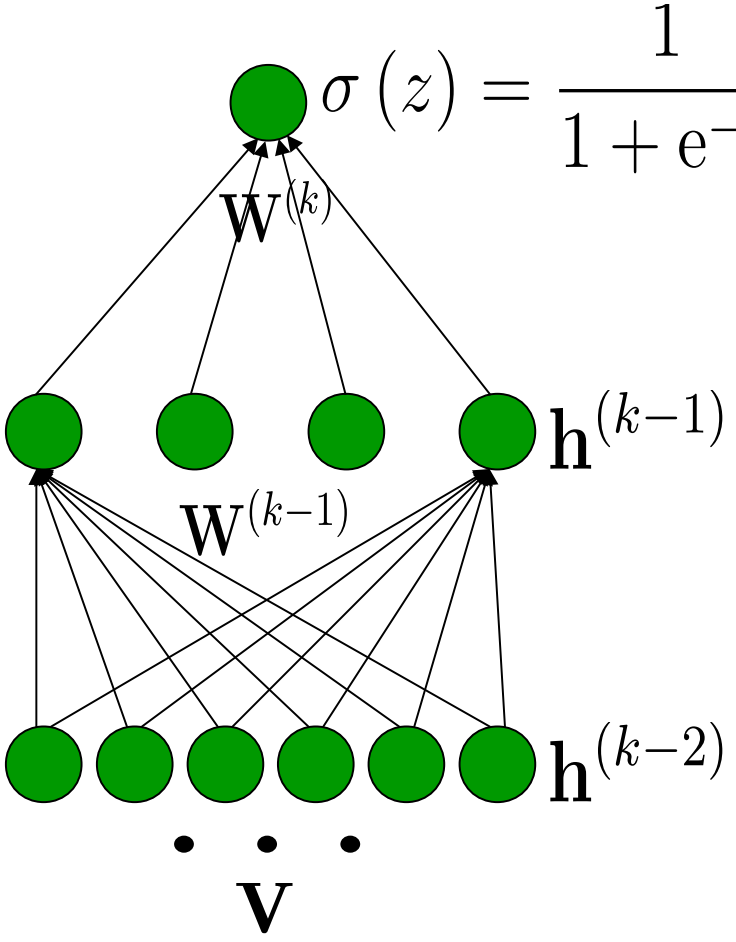


$$g(z) = \underbrace{\frac{1}{1+\mathrm{e}^{-z}}}_{} \quad (= \sigma(z))$$

Funkcja **logistyczna** (sigmoidalna)

Logit

Regresja logistyczna



Jaka funkcja kosztu będzie odpowiednia dla tego zadania?

$$J(\sigma(z), t) = ?$$

Może:

$$J(\sigma, t) = \frac{1}{2}(t - \sigma)^2 \quad (?)$$

Regresja logistyczna

Policzmy pochodną kosztu po logicie:

$$\begin{aligned}\frac{\mathrm{d}\sigma}{\mathrm{d}z} &= \frac{\mathrm{d}}{\mathrm{d}z}\left(1+\mathrm{e}^{-z}\right)^{-1} = -\left(1+\mathrm{e}^{-z}\right)^{-2}\frac{\mathrm{d}}{\mathrm{d}z}\left(1+\mathrm{e}^{-z}\right) \\ &= \left(1+\mathrm{e}^{-z}\right)^{-2}\mathrm{e}^{-z} = \frac{1}{1+\mathrm{e}^{-z}}\frac{\mathrm{e}^{-z}}{1+\mathrm{e}^{-z}} = \sigma\cdot\left(\frac{1+\mathrm{e}^{-z}}{1+\mathrm{e}^{-z}}-\frac{1}{1+\mathrm{e}^{-z}}\right) \\ &= \sigma\cdot(1-\sigma)\end{aligned}$$

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}z}J\left(\sigma\left(z\right),t\right) &= \frac{\mathrm{d}J}{\mathrm{d}\sigma}\frac{\mathrm{d}\sigma}{\mathrm{d}z} = \sigma\cdot(1-\sigma)\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}\sigma}\left(t-\sigma\right)^2 \\ &= \boxed{-\sigma\cdot(1-\sigma)\left(t-\sigma\right)}\end{aligned}$$

Regresja logistyczna

Policzmy pochodną kosztu po logicie:

$$\frac{\mathrm{d}}{\mathrm{d}z} J\left(\sigma\left(z\right), t\right) = -\sigma \cdot \left(1-\sigma\right)\left(t-\sigma\right)$$

Rozważmy przypadek, gdy obserwacja ma klasę 1 a odpowiedź MLP wynosi $\sigma = 10^{-6}$:

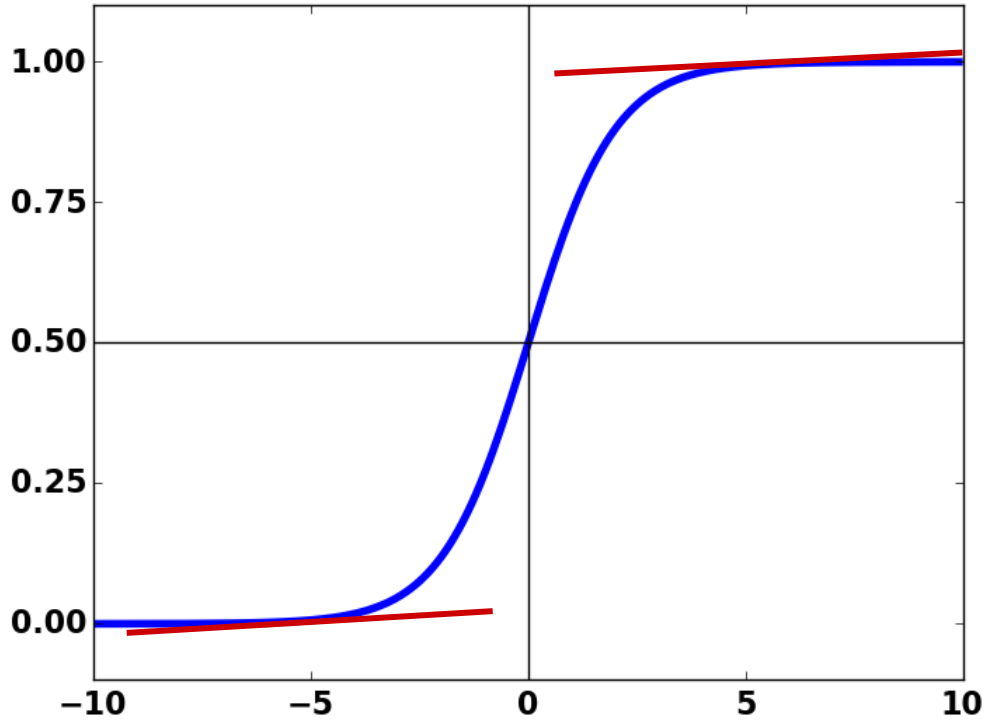
$$\frac{\mathrm{d}}{\mathrm{d}z} J\left(\sigma = 10^{-6}, t = 1\right) = -\sigma \cdot \left(1-\sigma\right)\left(t-\sigma\right)$$

$$= -10^{-6} \cdot \left(1-10^{-6}\right)\left(1-10^{-6}\right) \approx -10^{-6}$$ **!**

Podobnie będzie, gdy obserwacja ma klasę 0, a MLP odpowiada $\sigma = 0.999999$.

Regresja logistyczna

Przyczyną jest bardzo mały gradient funkcji logistycznej, gdy logit jest daleko od zera.



Musimy dobrać lepszą funkcję kosztu.

Koszt ten powinien skompensować „niefortunny” gradient funkcji logistycznej.

Regresja logistyczna

Okazuje się, że odpowiedni koszt można skonstruować biorąc pod uwagę kryterium największej wiarygodności (*maximum likelihood*).

Odpowiedź MLP możemy potraktować jako prawdopodobieństwo, które MLP przypisuje klasie 1. Wówczas:

- gdy klasa obserwacji wynosi $t = 1$, to prawdopodobieństwo jakie MLP przypisuje prawidłowej klasie wynosi σ.
- gdy klasa obserwacji wynosi $t = 0$, to prawdopodobieństwo jakie MLP przypisuje prawidłowej klasie wynosi $1 - \sigma$.

Regresja logistyczna

Obie te zależności możemy wyrazić zwięźle w postaci:

$$P\left(t|\phi_{\mathrm{MLP}}\right) = \sigma^{t} \cdot \left(1-\sigma\right)^{1-t}$$

gdzie:

$$\phi_{\mathrm{MLP}} \text{ - parametry MLP}$$

Prawdopodobieństwo, że MLP udzieli prawidłowych odpowiedzi dla całego zbioru uczącego wynosi więc:

$$P\left(\mathcal{X}|\phi_{\mathrm{MLP}}\right) = \prod_{i=1}^{n} \sigma_i^{t_i} \cdot \left(1-\sigma_i\right)^{1-t_i}$$

Regresja logistyczna

To prawdopodobieństwo przyjmujemy jako funkcję wiarygodności naszego modelu (t.j. MLP):

$$\mathcal{L}\left(\phi_{\mathrm{MLP}}, \mathcal{X}\right) = \prod_{i=1}^{n} \sigma_i^{t_i} \cdot \left(1 - \sigma_i\right)^{1-t_i}$$

Cel uczenia MLP stawiamy jako maksymalizację funkcji wiarygodności.

W tym celu będziemy szukali minimum zanegowanej logarytmicznej funkcji wiarygodności.

Regresja logistyczna

W tym celu będziemy szukali minimum zanegowanej logarytmicznej funkcji wiarygodności:

$$\arg\min_{\phi_{\mathrm{MLP}}} \left[-\log \mathcal{L}\left(\phi_{\mathrm{MLP}}, \mathcal{X}\right) \right]$$

Biorąc zanegowany logarytm z funkcji wiarygodności otrzymujemy funkcję kosztu postaci:

$$J\left(\phi_{\mathrm{MLP}}, \mathcal{X}\right) = -\log \prod_{i=1}^{n} \sigma_i^{t_i} \cdot \left(1-\sigma_i\right)^{1-t_i}$$

$$= -\sum_{i=1}^{n} t_i \log \sigma_i + \left(1-t_i\right) \log\left(1-\sigma_i\right)$$

Regresja logistyczna

Będziemy uczyć stochastycznym spadkiem gradientu.

W każdym kroku uczącym będziemy więc liczyć koszt po jednym (losowo wybranym) przykładzie uczącym.

Ostatecznie otrzymujemy więc funkcję kosztu postaci:

$$J\left(y, t; \boldsymbol{\phi}_{\mathrm{MLP}}\right) = -\left[t \log \sigma + (1-t) \log\left(1-\sigma\right)\right]$$

Entropia krzyżowa (*cross-entropy*)

Regresja logistyczna

Obecnie pochodna kosztu po logicie wynosi:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}z}J\left(\sigma\left(z\right),t\right) &= \frac{\mathrm{d}J}{\mathrm{d}\sigma}\frac{\mathrm{d}\sigma}{\mathrm{d}z} = -\sigma\cdot\left(1-\sigma\right)\frac{\mathrm{d}}{\mathrm{d}\sigma}\left[t\log\sigma+\left(1-t\right)\log\left(1-\sigma\right)\right] \\ &= -\sigma\cdot\left(1-\sigma\right)\left[\frac{t}{\sigma}-\frac{1-t}{1-\sigma}\right] \\ &= -t\left(1-\sigma\right)+\left(1-t\right)\sigma\end{aligned}$$

co daje:

$$\frac{\mathrm{d}}{\mathrm{d}z}J\left(\sigma,t\right) = \begin{cases}\sigma-1 & \text{gdy } t=1 \\ \sigma & \text{gdy } t=0\end{cases} \quad \Rightarrow \quad \boxed{\frac{\mathrm{d}J}{\mathrm{d}z} = \sigma - t}$$

Regresja softmax

Regresja logistyczna pozwala przypisać obserwację do jednej z dwóch kas.

Co w przypadku gdy mamy więcej klas?

Może neuron wyjściowy będzie przewidywał indeks klasy – a więc wartości 0, 1, 2, 3, …?

Modelowanie takiej schodkowej funkcji byłoby bardzo trudne.

- Takie naiwne rozwiązanie nie da dobrych rezultatów.

Regresja softmax

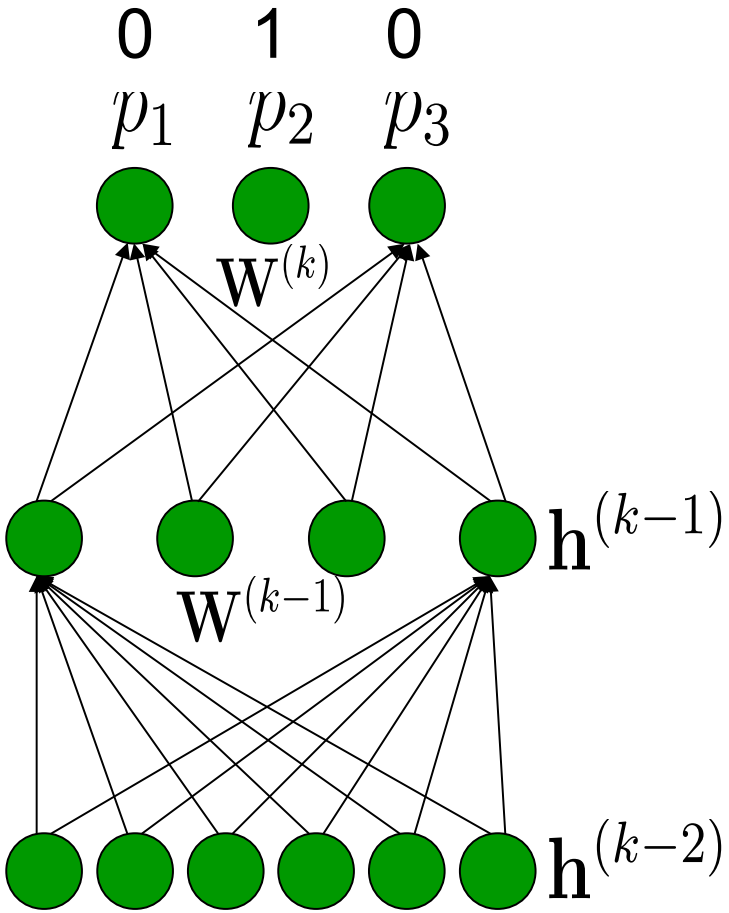
Lepszym rozwiązaniem będzie skonstruowanie warstwy wyjściowej w postaci *m* neuronów (*m* – liczba przewidywanych klas).

- Każdy z neuronów będzie przewidywał prawdopodobieństwo przynależności wejściowej obserwacji do klasy reprezentowanej przez ten neuron.
- Aktywacje neuronów w warstwie wyjściowej muszą więc sumować się do jedności.
- Wartości przewidywane będą miały postać:

$$\mathbf{t} = (0, 0, \ldots, 0, 1, 0, \ldots, 0)$$

pozycja odpowiadająca klasie do której należy obserwacja

Regresja softmax



Jaką funkcją będziemy modelować prawdo-podobieństwa w warstwie wyjściowej?

Użyjemy **funkcji softmax**

$$p_i = \frac{\mathrm{e}^{z_i}}{\sum_{l=1}^{k} \mathrm{e}^{z_l}}$$

Zwróć uwagę, że indeksy dolne w powyższym wzorze numerują klasy (nie przykładu uczące).

Regresja softmax

Podobnie jak w przypadku regresji logistycznej, funkcję kosztu odpowiednią dla regresji softmax można skonstruować wychodząc z kryterium największej wiarygodności.

Otrzymamy wówczas funkcję kosztu postaci (dla stochastycznego spadku gradientu):

$$J(\mathbf{p}, \mathbf{t}) = -\sum_{i=1}^{k} t_i \log p_i \qquad \nabla_{\mathbf{z}} J(\mathbf{p}, \mathbf{t}) = \mathbf{p} - \mathbf{t}$$

gdzie:

$$\mathbf{t} = (t_1, t_2, \ldots, t_k) \text{ - wartość przewidywana}$$

$$\mathbf{p} = (p_1, p_2, \ldots, p_k) \text{ - odpowiedź sieci}$$

Sieci Neuronowe

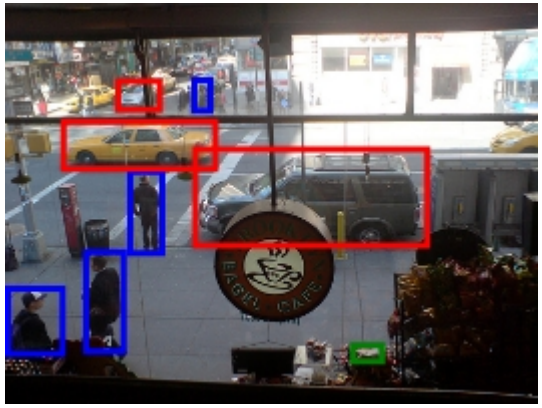
Sieci konwolucyjne

Rozpoznawanie obrazów








O. Russakovsky, J. Deng, et al. "ImageNet Large Scale Visual Recognition Challenge" arXiv:1409.0575, 2014

Sieci konwolucyjne

Klasyczna sieć konwolucyjna to wielowarstwowa sieć neuronowa, której architektura jest przystosowana do przetwarzania danych o charakterze przestrzennym.

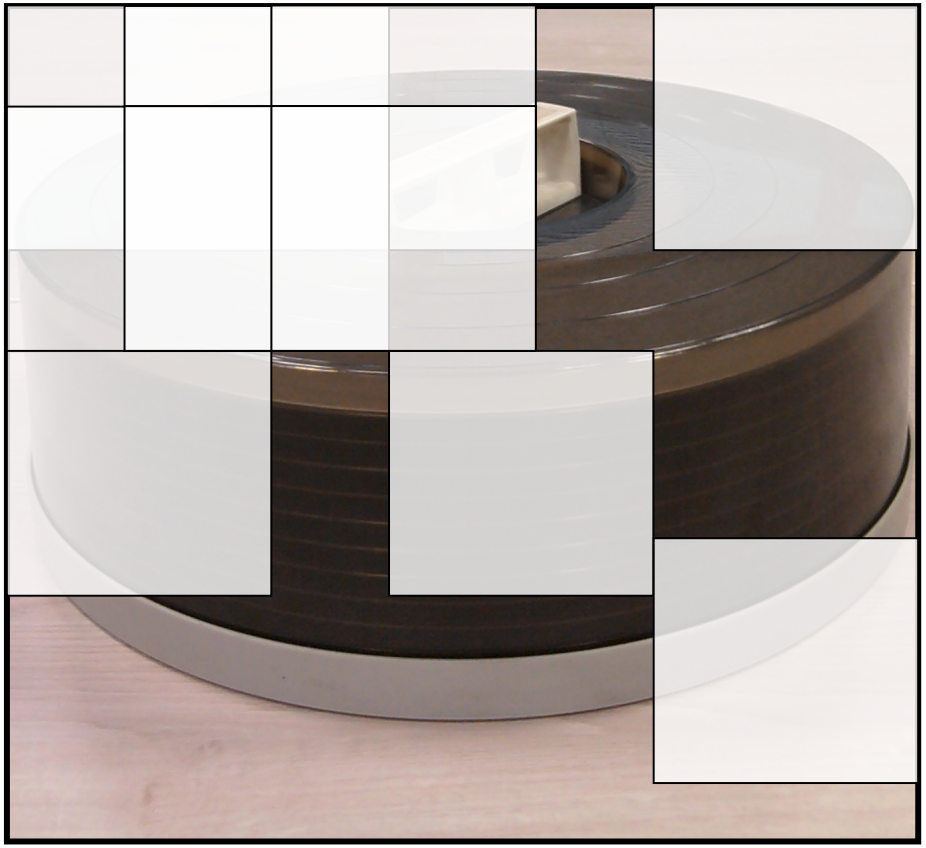
- Głównie obrazy / wolumeny 3D.
- Obecnie coraz częściej stosowane w analizie danych sekwencyjnych (np. tekstu).

Sieci konwolucyne zbudowane są z dwóch podstawowych rodzajów warstw: warstwy konwolucyjnej i warstwy zbierającej.

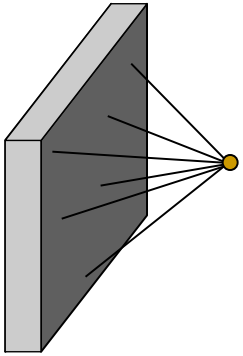
Sieci konwolucyjne uczone są algorytmem wstecznej propagacji błędu.

Warstwa konwolucyjna

Warstwa konwolucyjna dzieli więc obraz na niewielkie, z reguły zachodzące na siebie fragmenty: tzw. pola odbiorcze (ang. *receptive fields*).



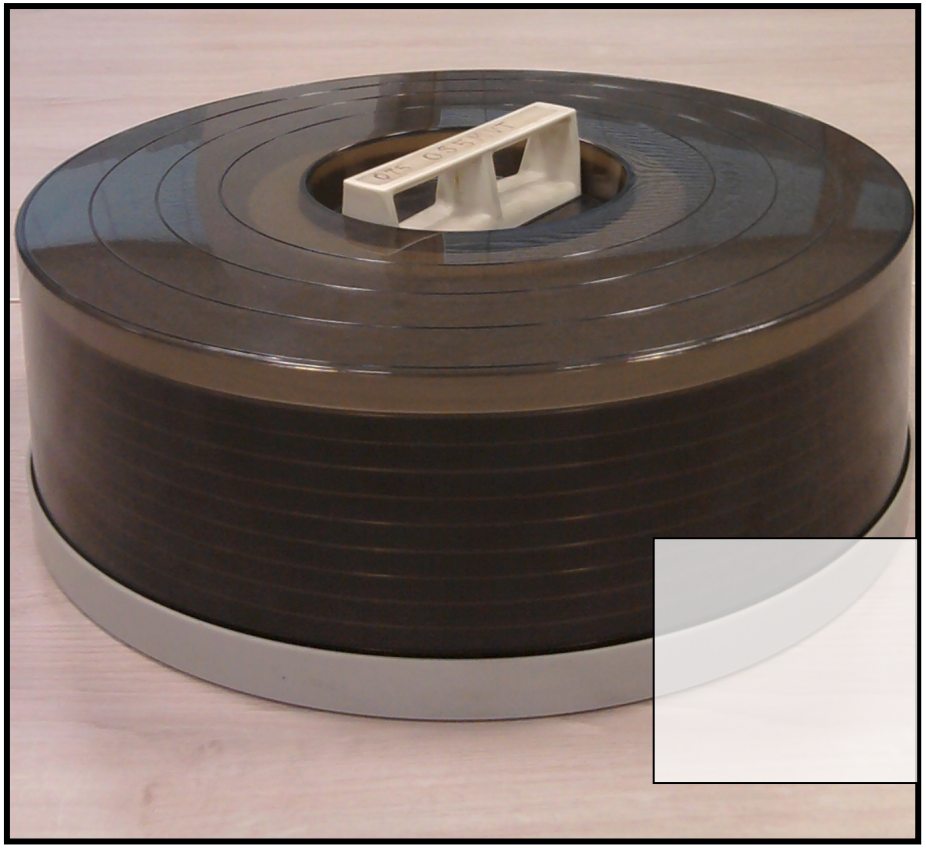
Piksele w polu odbiorczym stanowią wejście do neuronu warstwy konwolucyjnej.



Neuron ten ma taki sam mechanizm aktywacji, jak neuron w MLP.

Warstwa konwolucyjna

Warstwa konwolucyjna dzieli więc obraz na niewielkie, z reguły zachodzące na siebie fragmenty: tzw. pola odbiorcze (ang. *receptive fields*).



Aktywacje neuronu są wyliczane osobno dla każdego pola odbiorczego.

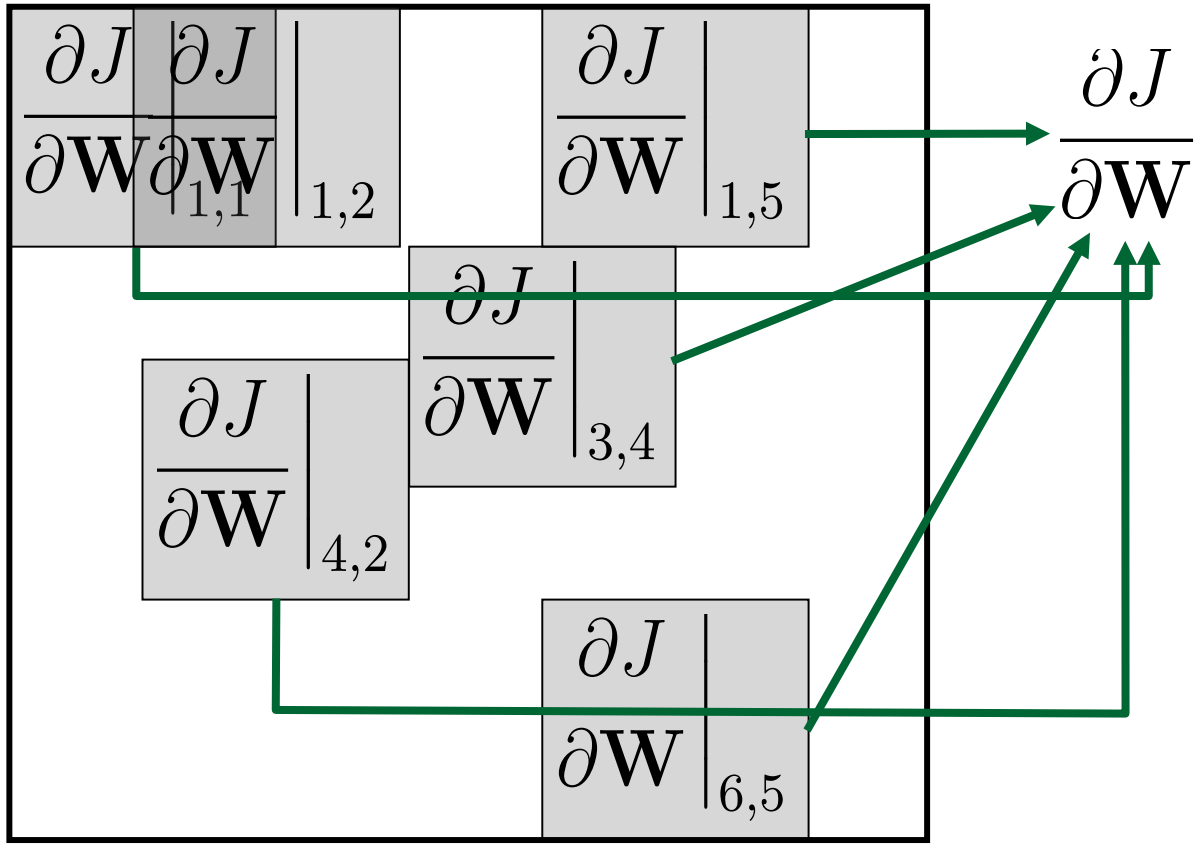
Jednak wszystkie pola odbiorcze współdzielą wagi!

Neuron realizuje więc w istocie operację przypominającą konwolucję. Stąd potoczne określenie: filtr.

W mechanizmie tym kryje się założenie, że cecha wykrywana przez neuron może być zlokalizowana w dowolnym obszarze zdjęcia.

Warstwa konwolucyjna

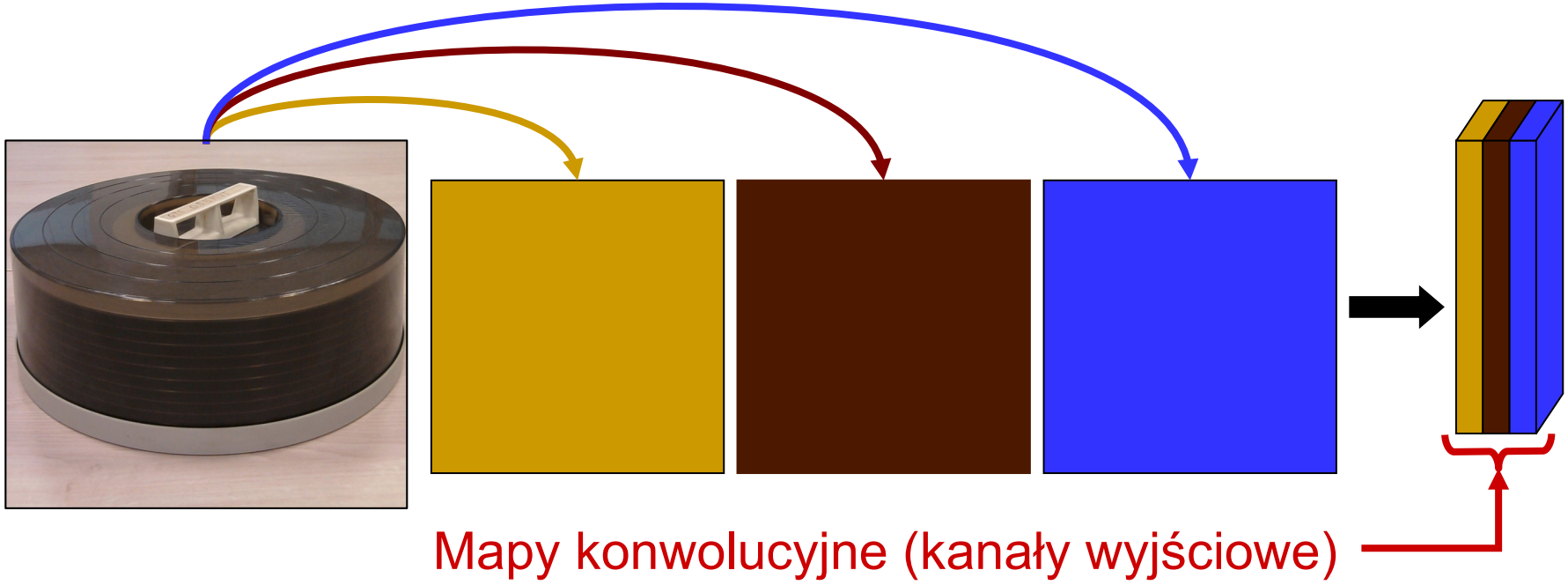
W trakcie wstecznej propagacji błędu, w każdym polu odbiorczym wyliczane są pochodne funkcji kosztu. Pochodne z poszczególnych pól sumują się, dając ostatecznie poprawkę do wektora wag neuronu.



Warstwa konwolucyjna

Pojedynczy neuron w warstwie konwolucyjnej wykrywa określoną cechę (np. krawędź) w całym obszarze zdjęcia.

- Aby rozpoznać obiekt musimy oczywiście wykryć wiele cech.
- Warstwa konwolucyjna będzie więc miała wiele neuronów.



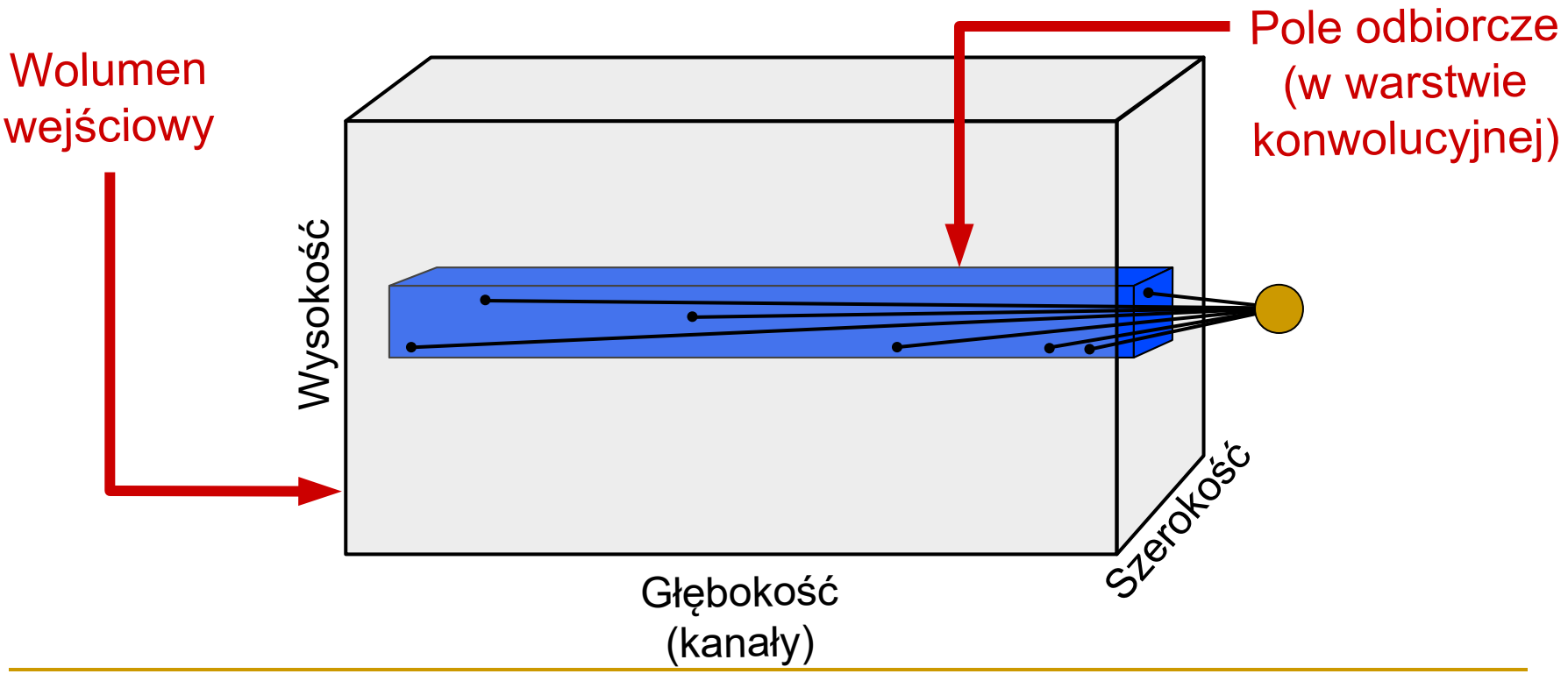
- Warstwa konwolucyjna przekształca więc wolumen wejściowy w wolumen wyjściowy.

Warstwa konwolucyjna

Ale jak w warstwie konwolucyjnej traktowane są poszczególne kanały obrazu?

- Na przykład kolory?

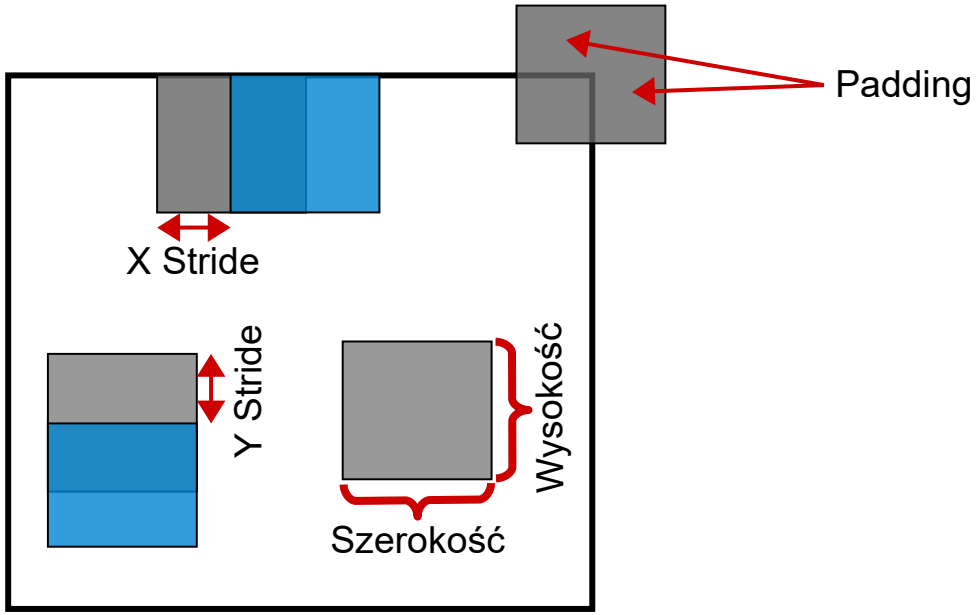
W warstwie konwolucyjnej pole odbiorcze obejmuje wszystkie wejściowe kanały.



Warstwa konwolucyjna

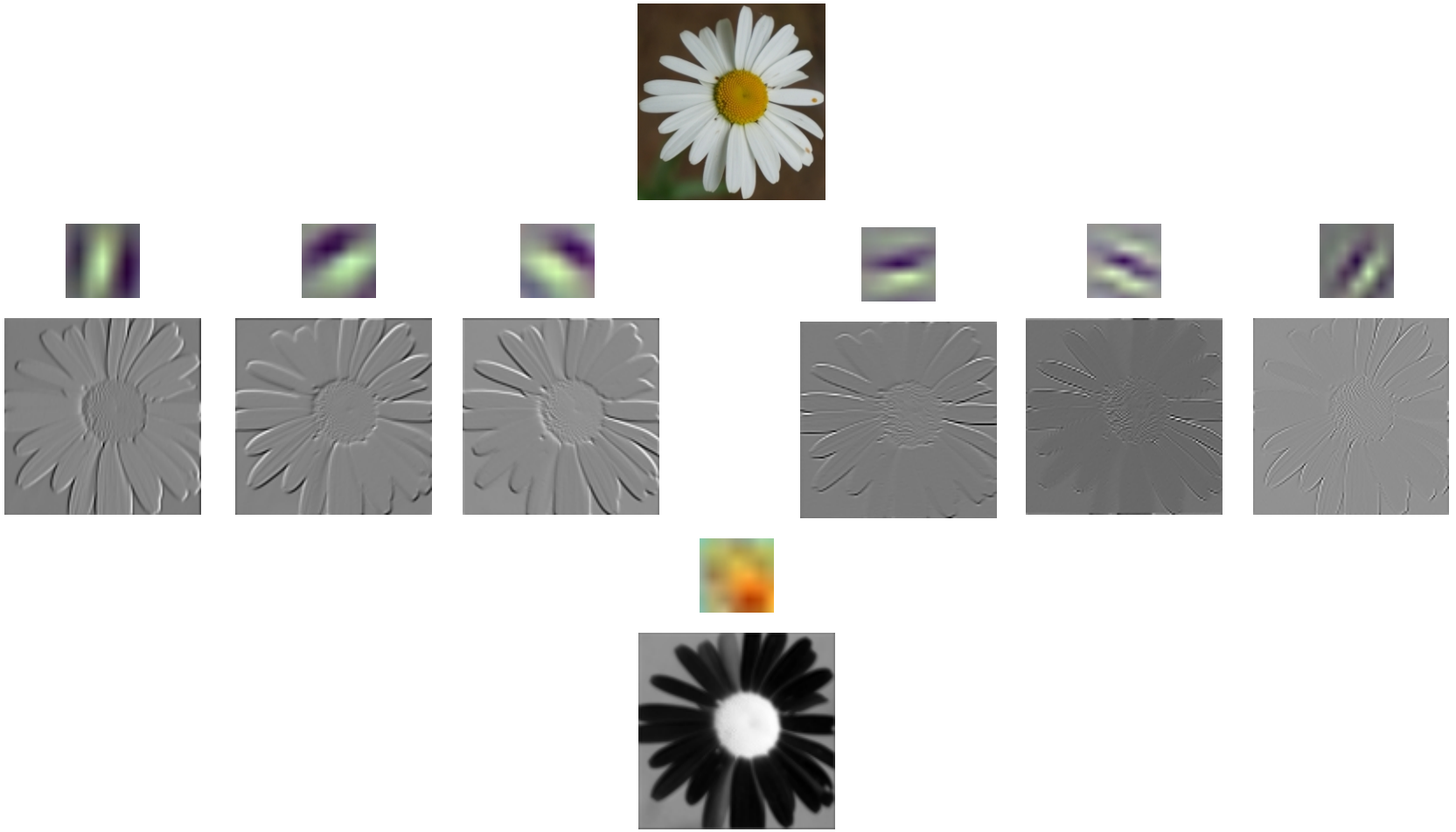
Specyfikując warstwę konwolucyjną musimy określić:

- liczbę kanałów wejściowych/wyjściowych,
- wysokość i szerokość pola odbiorczego (filtru),
- przesunięcie (w pionie i poziomie) pomiędzy sąsiadującymi polami odbiorczymi (ang. *stride*),
- ewentualny padding na brzegach wolumenu (np. zerami).



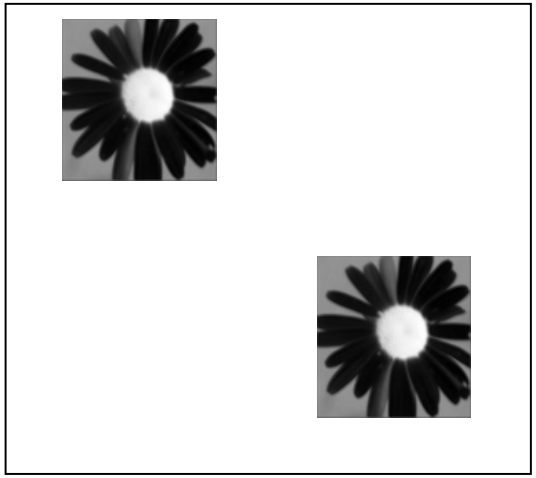
Warstwa konwolucyjna

Jakich cech uczą się warstwy konwolucyjne?



Efekty translacji

Zwróć uwagę, że aktywacje w warstwie konwolucyjnej zależą od położenia obiektów na zdjęciu.

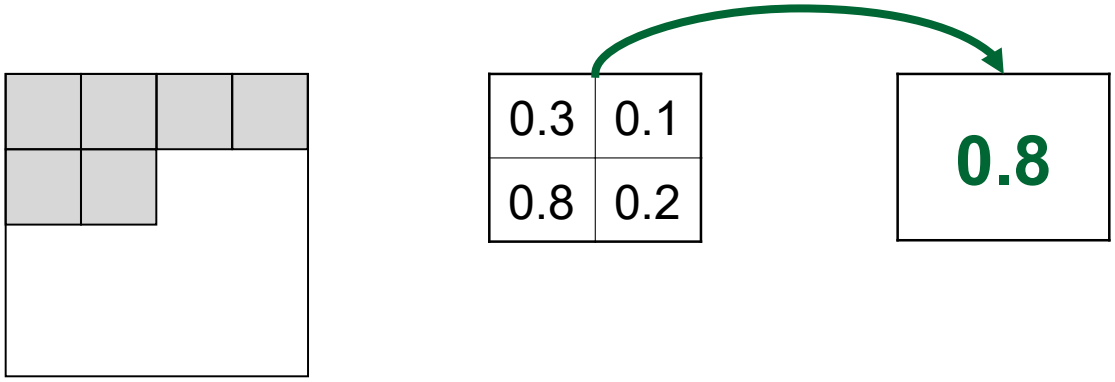


- Przesunięcie obiektu powoduje odpowiednie przesunięcie aktywacji w warstwie konwolucyjnej.
- Warstwa konwolucyjna jest więc ekwiwariantna ze względu na translację.
- Dążymy jednak do zbudowania modelu, który będzie poprawnie intepretował obrazy, niezależnie od lokalizacji uwidocznionych na nich obiektów.
- Model taki powinien być inwariantny ze względu na translację.

Warstwa zbierająca

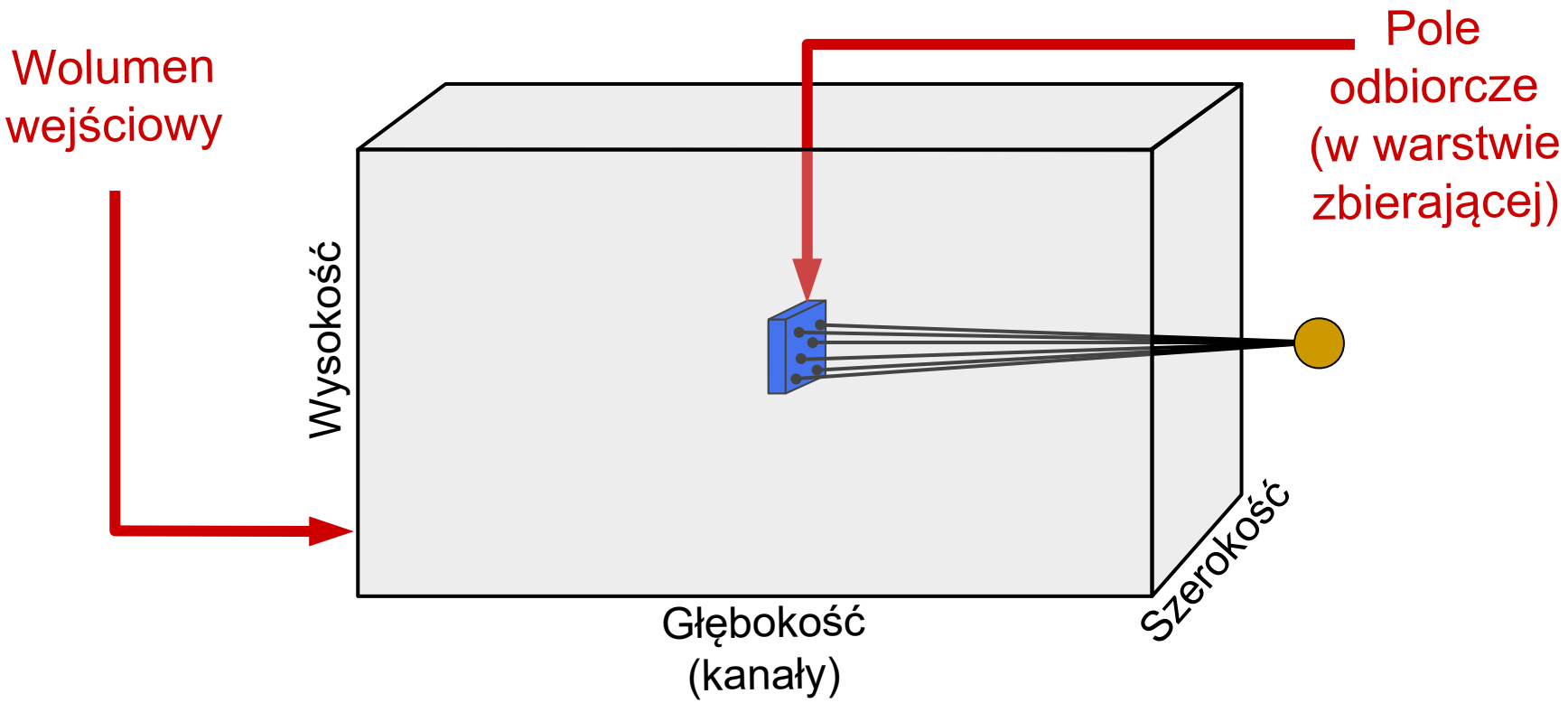
W sieciach konwolucyjnych inwariancję ze względu na translację zapewniają warstwy zbierające (ang. *pooling layers*).

- Podobnie jak warstwa konwolucyjna, warstwa zbierająca dzieli wejściowy wolumen na pola odbiorcze.
- Aktywacją warstwy zbierającej dla pojedynczego pola odbiorczego jest (z reguły) maksimum intensywności pikseli w tym polu.



Warstwa zbierająca

Ale uwaga: warstwa zbierająca operuje osobno na każdym kanale wejściowego wolumenu.



- Warstwa zbierająca zmniejsza więc szerokość i wysokość wejściowego wolumenu, lecz nie zmienia głębokości.

Warstwa zbierająca

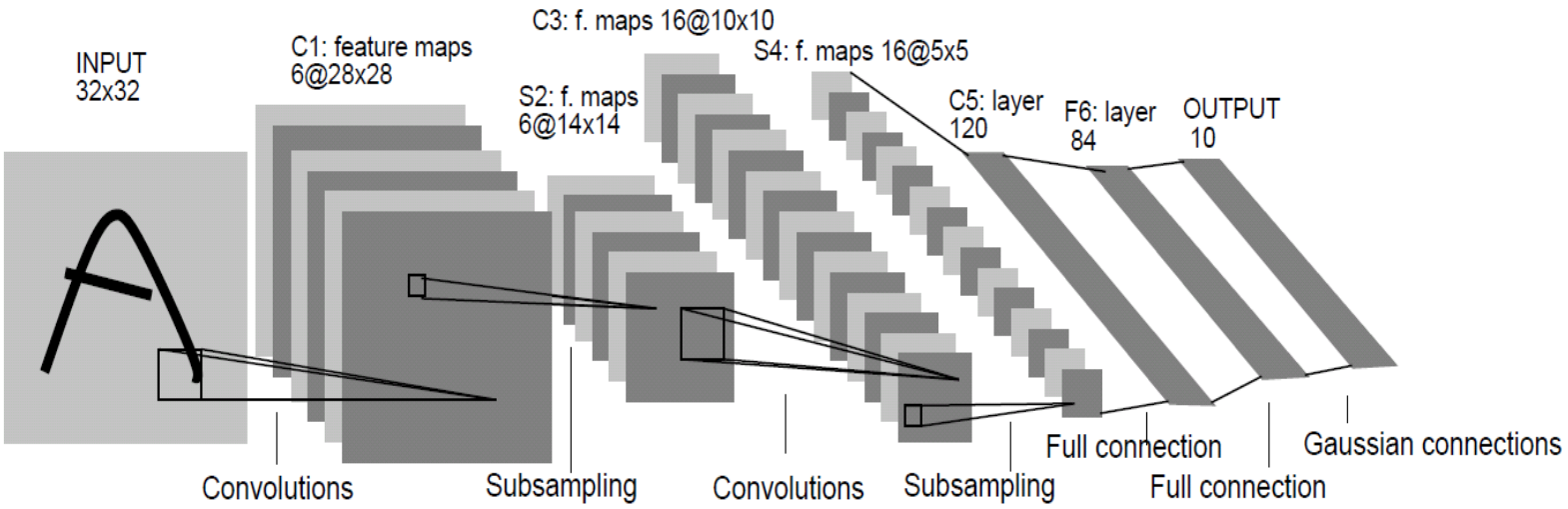
Dlaczego warstwy zbierające wprowadzają inwariancję ze względu na translację?

- Jeśli w polu zbierającym znajduje się pewna cecha rozpoznawana przez sieć (np. krawędź), to wrażliwy na tą cechę neuron będzie miał wysoką wartość aktywacji.
- Warstwa zbierająca zredukuje całe pole odbiorcze, w którym wykryto tą cechę, do jednego piksela.
- Wartością aktywacji dla pola odbiorczego będzie aktywacja neuronu, który wykrył cechę.
- W ten sposób zaniedbana zostanie część informacji o dokładnej lokalizacji wykrytej cechy.
- Umieszczenie w sieci kilku warstw zbierających może więc zniwelować efekty translacji na zdjęciu.

Sieci LeNet

Pierwsze poważne zastosowania sieci konwolucyjnych przedstawił Yann LeCun.

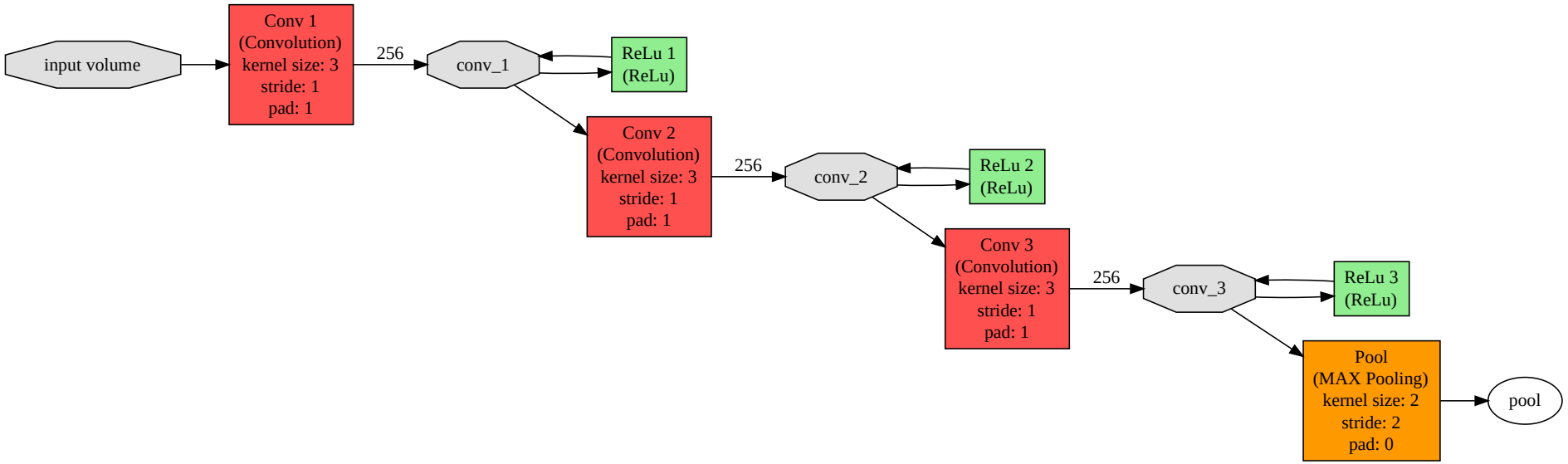
Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, "Gradient-based learning applied to document recognition", Proceedings of the IEEE, 86(11), pp. 2278 –2324, 1998



Współczesne Architektury CNN: sieci VGG

Sieci VGG charakteryzują się bardzo jednorodną architekturą.

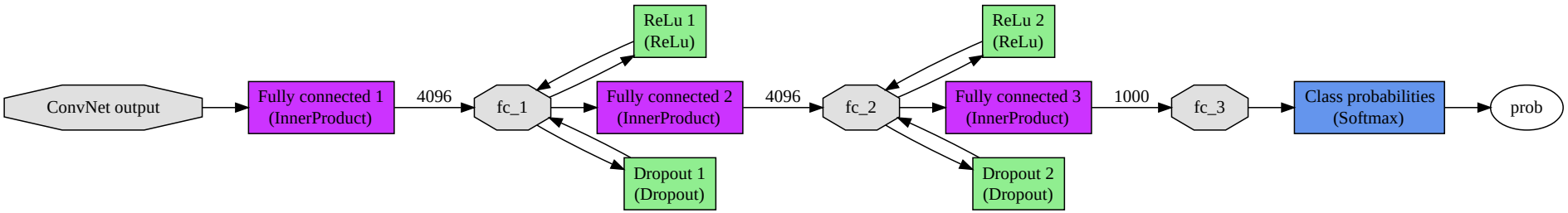
- Podstawowym elementem sieci VGG jest sekwencja 2-4 konwolucji 3x3, po której umieszczona jest typowa warstwa zbierająca. Sieć składa się z wielu takich „bloków”.
- Po warstwie zbierającej głębokość wolumenów jest (z reguły) zwiększana dwukrotnie.



Współczesne Architektury CNN: sieci VGG

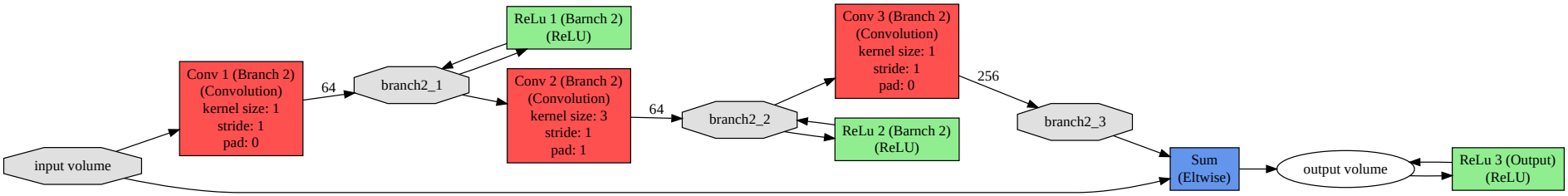
Sieci VGG charakteryzują się bardzo jednorodną architekturą.

- Podstawowym elementem sieci VGG jest sekwencja 2-4 konwolucji 3x3, po której umieszczona jest typowa warstwa zbierająca. Sieć składa się z wielu takich „bloków”.
- Po warstwie zbierającej głębokość wolumenów jest (z reguły) zwiększana dwukrotnie.
- Na końcu sieci VGG umieszczony jest klasyczny MLP, przewidujący prawdopodobieństwo przynależności obrazu do poszczególnych klas.



Współczesne Architektury CNN: sieci ResNet

Podstawową jednostką funkcjonalną w sieciach ResNet są bloki residualne:



Blok residualny propaguje aktywacje dwoma ścieżkami: poprzez sekwencję konwolucji i poprzez funkcję identycznościową.

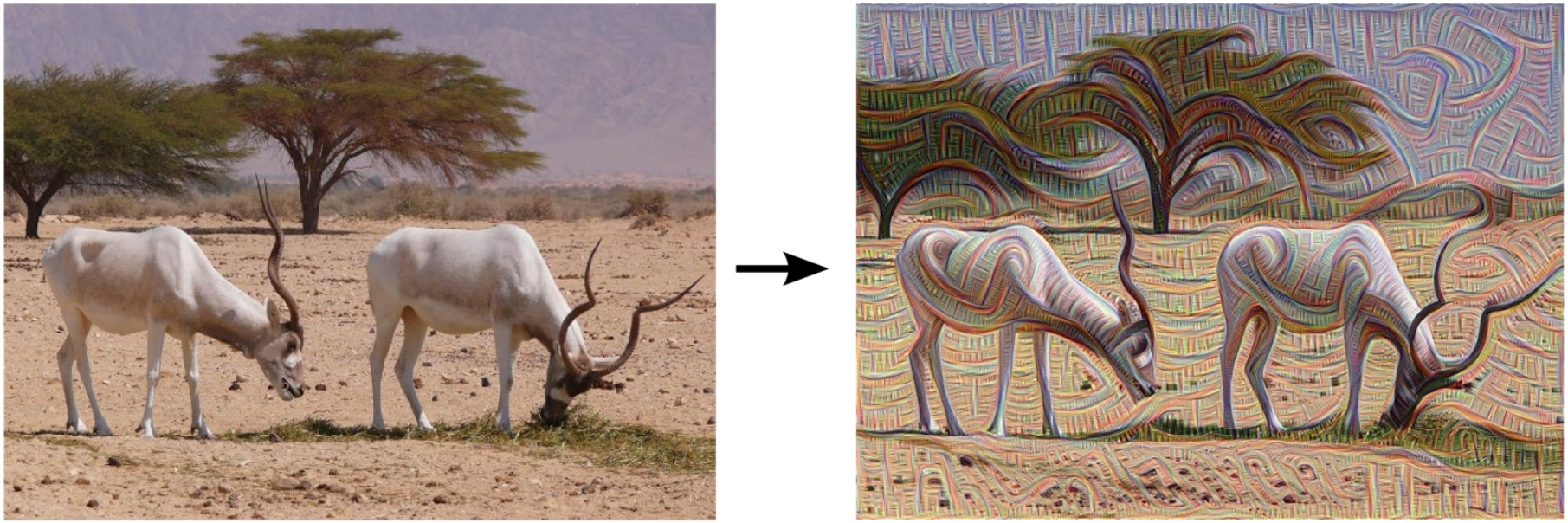
- Aktywacje na końcu tych ścieżek są sumowane.

K. He, X. Zhang, S. Ren, J. Sun, "Deep Residual Learning for Image Recognition", IEEE Conference on Computer Vision and Pattern Recognition, CVPR2016, 2016

Jakie cechy wykrywają CNN?

Co będzie, gdy zmodyfikujemy wejściowy obraz w taki sposób, aby wzmocnić jego reprezentację w wybranej warstwie konwolucyjnej?

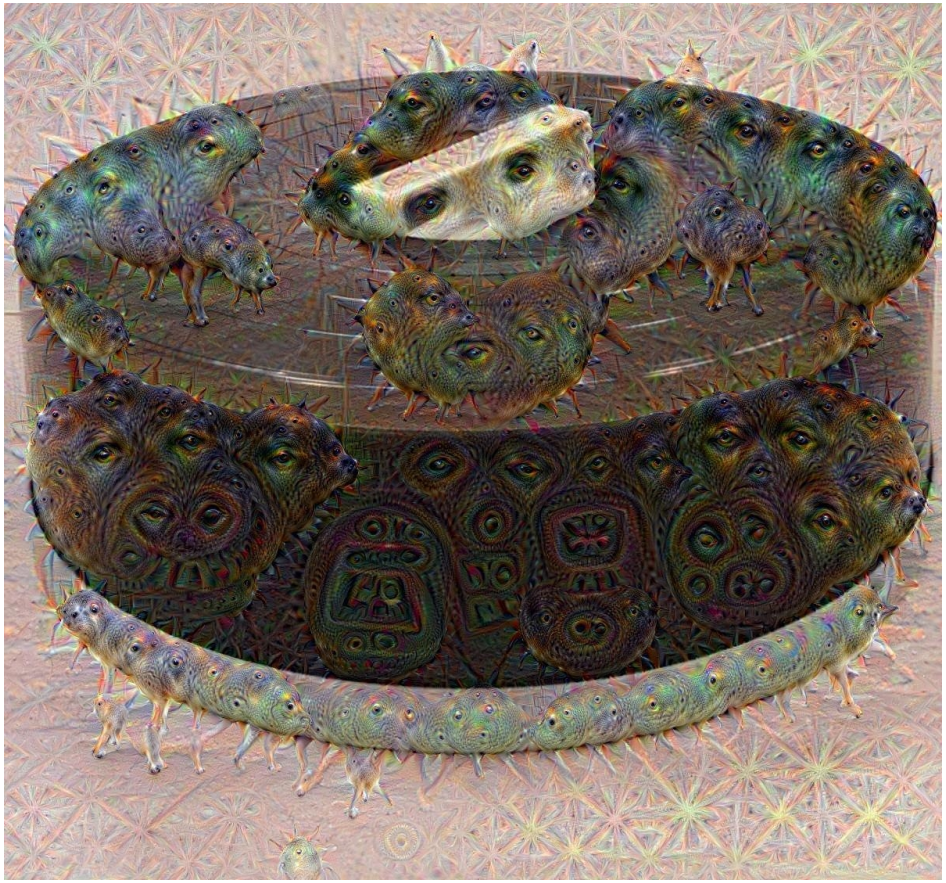
Cechy wykrywane w pierwszych warstwach sieci Inception Network (sieć Google).

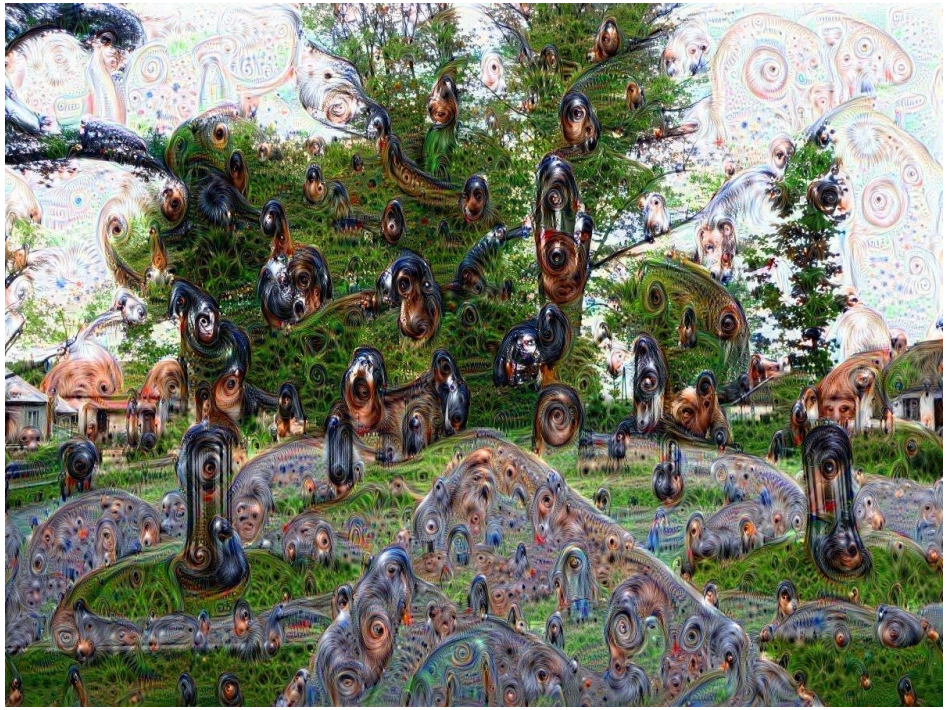


Left: Original photo by Zachi Evenor. Right: processed by Günther Noack.

Jakie cechy wykrywają CNN?

Co będzie, gdy zmodyfikujemy wejściowy obraz w taki sposób, aby wzmocnić jego reprezentację w wybranej warstwie konwolucyjnej?





Dziękuję za Uwagę!