AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

AGH UNIVERSITY OF SCIENCE
AND TECHNOLOGY

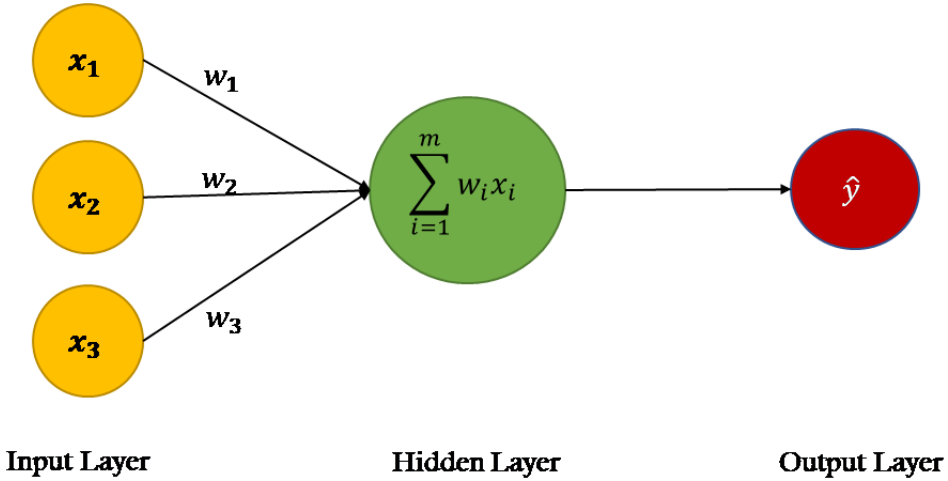
Głębokie sieci neuronowe

Dariusz Kucharski
Katedra Automatyki i Robotyki

24.10.2019, Kraków

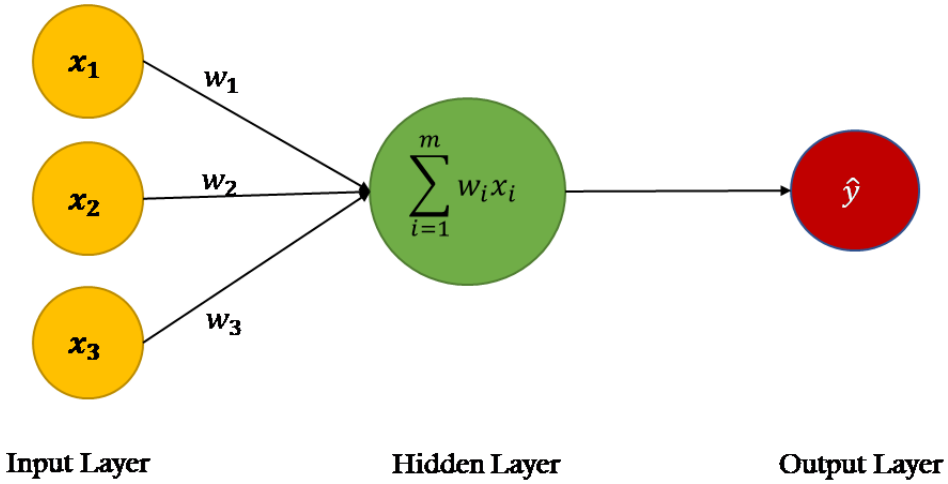
Sztuczny neuron

- Regresja liniowa dla liniowej funkcji aktywacji
- Regresja logistyczna dla funkcji aktywacji sigmoid



Schemat sieci z jednym neuronem [4]

Parametry sieci

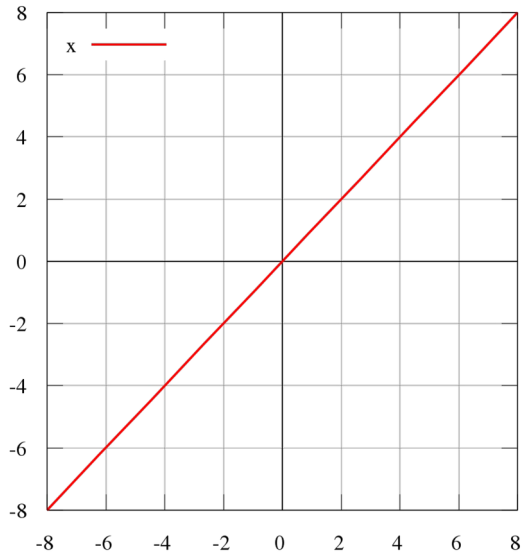


Schemat sieci z jednym neuronem [4]

- Ilość parametrów równa się długości wektora wejściowego
- W przypadku wielu neuronów, ich liczba zwielokrotnia ilość parametrów do optymalizacji

Funkcje aktywacji

- Liniowa funkcji aktywacji

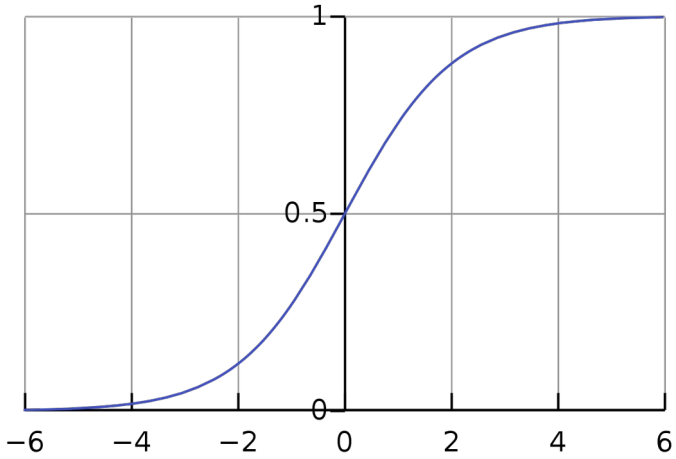


https://en.wikipedia.org/wiki/Identity_function

- Funkcja sigmoidalna

$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$$

https://en.wikipedia.org/wiki/Sigmoid_function



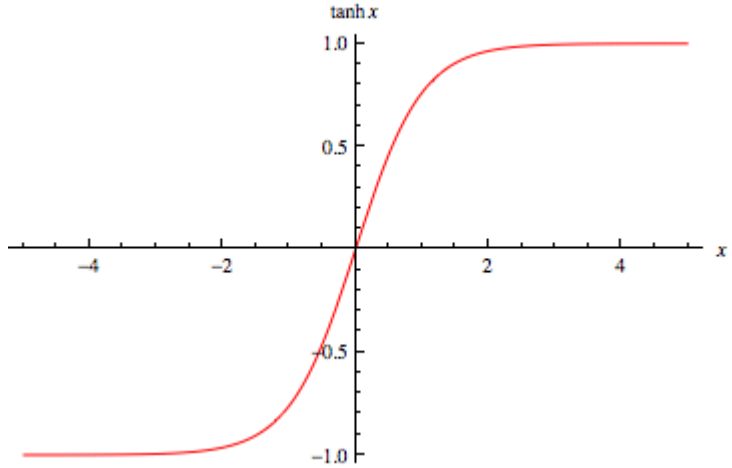
https://en.wikipedia.org/wiki/Sigmoid_function

Funkcje aktywacji

» Tangens hiperboliczny

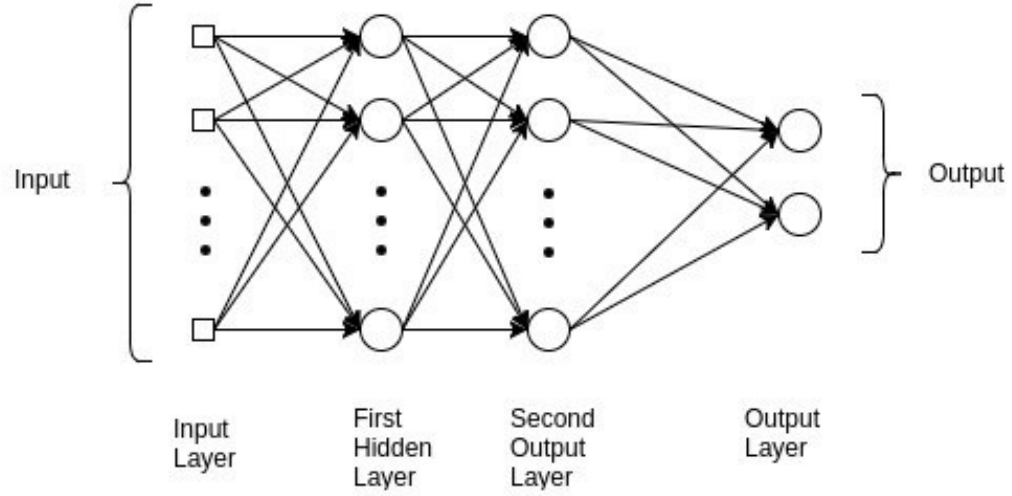
$$\operatorname{tgh} x = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

https://pl.wikipedia.org/wiki/Funkcje_hiperboliczne



http://mathworld.wolfram.com/images/interactive/TanhReal.gif

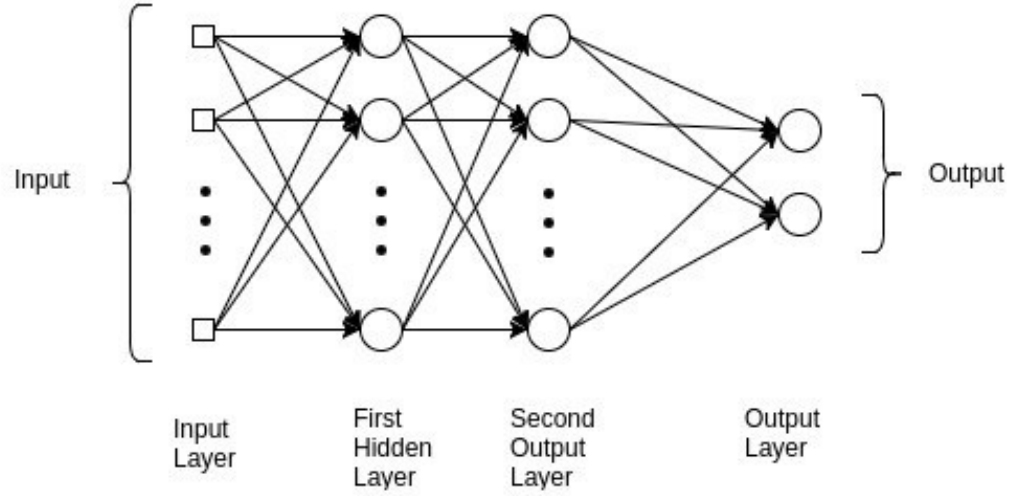
Sieci MLP



Schemat sieci mlp [12]

- Wiele neuronów, połączonych warstwami każdy z każdym
- f(x) = f1(f2(f3(x))) – f(x) = x*W1=y1; y1*W2 = y2; y2*W3=y3=f(x)
- Dla sieci o 2 warstwach ukrytych, oraz wejściu n-elementowym ilość parametrów do optymalizacji wynosi:
  - Zaczynając od wejścia: n * ilość neuronów warstwie ukrytej (h1)
  - h1 * ilość neuronów w drugiej warstwie ukrytej (h2)
  - h2 * ilość wyjść (o)
- W sumie: n * h1 + h1 * h2 + h2 * o

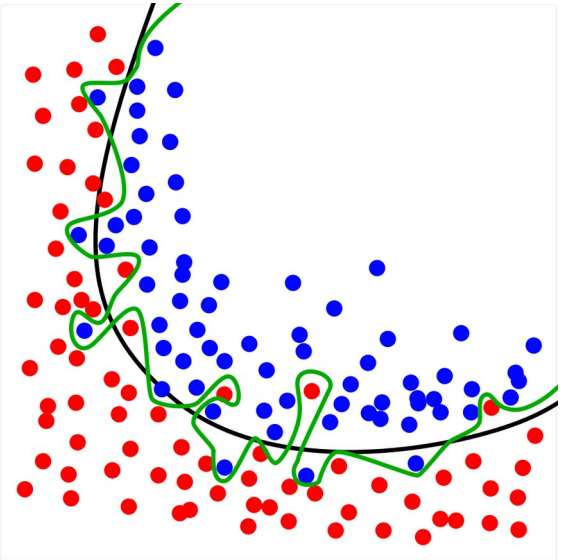
Parametry sieci MLP

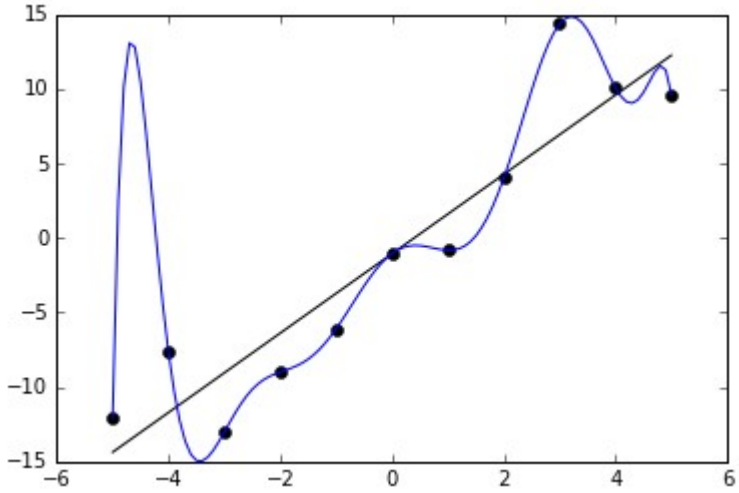


Schemat sieci z mlp [12]

- Przykład: dla sieci, która ma 32 neurony w pierwszej warstwie ukrytej, 8 neuronów w drugiej warstwie ukrytej, wektor cech o długości 10, jedno wyjście:
  - 10 cech * 32 neurony = 320
  - 32 wejścia na drugą warstwę ukrytą * 8 neuronów = 256
  - 8 * 2 = 16
- W sumie: 320 + 256 + 8 = 584

Ilość parametrów vs ilość przykładów



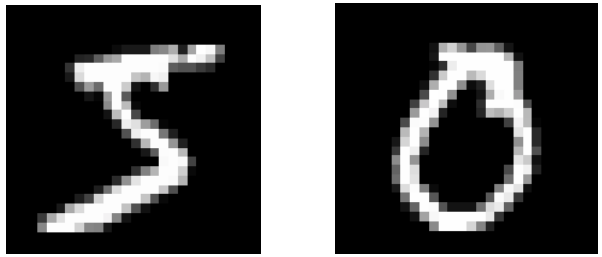


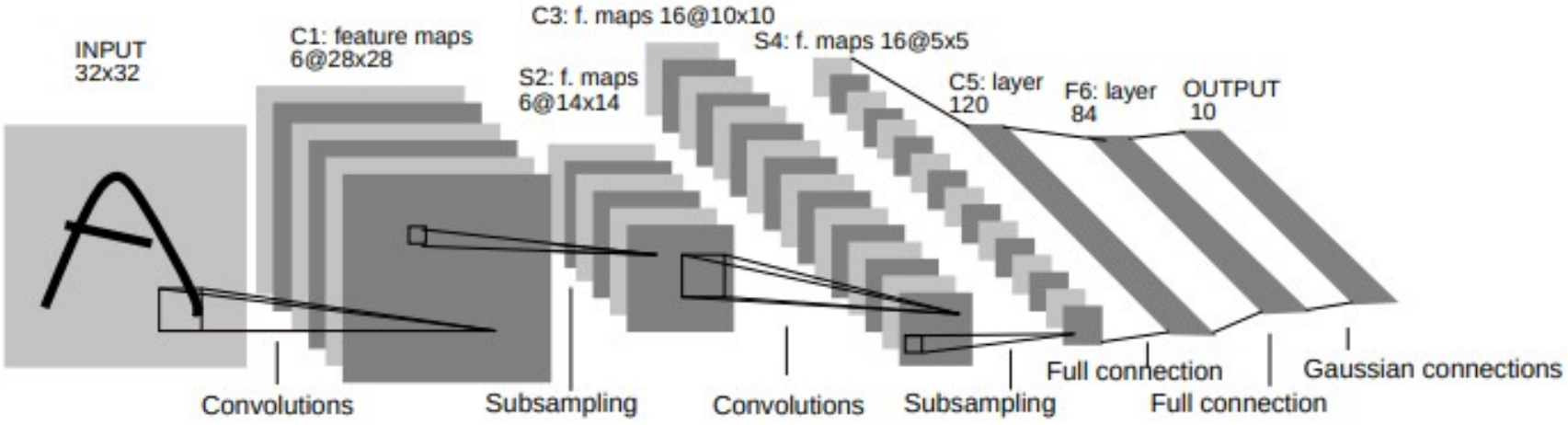
https://en.wikipedia.org/wiki/Overfitting

» Często zbyt mała ilość przykładów uczących w stosunku do ilości parametrów, może prowadzić do dopasowania się modelu do danych (overfitting)

Sieci konwolucyjne

- Yann Lecun
- baza Mnist [1]

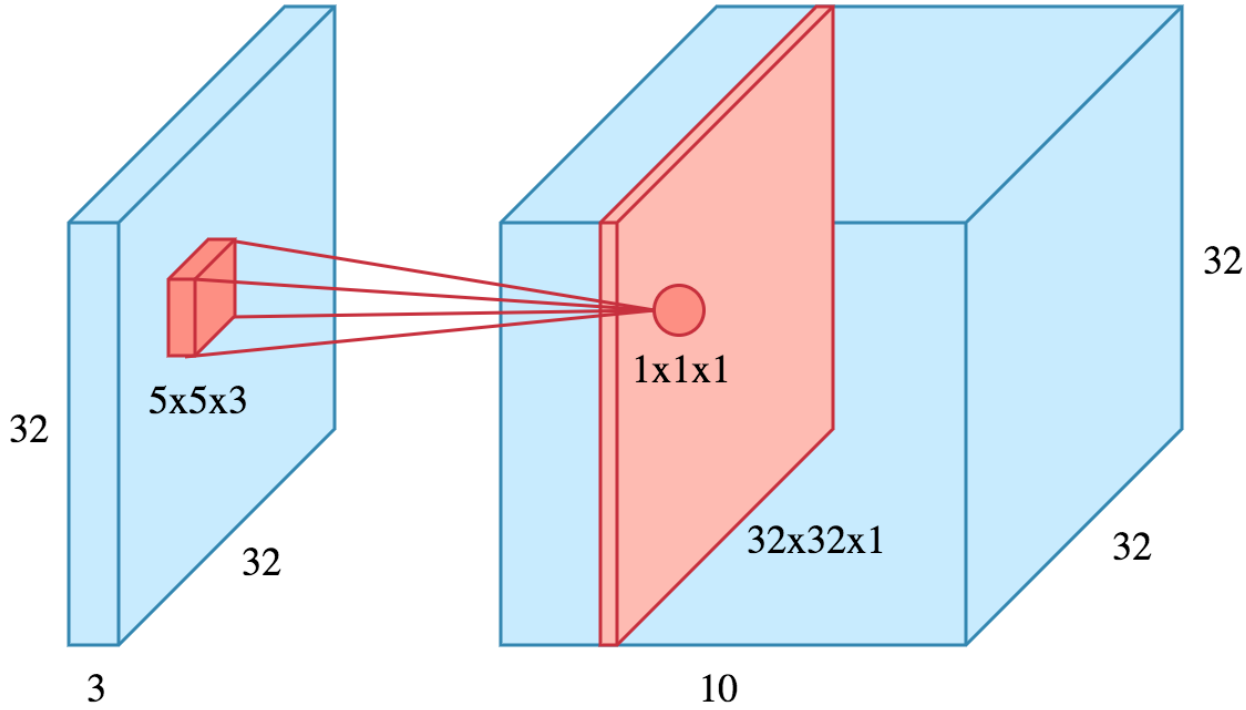




Architektura sieci do klasyfikacji odręcznie pisanych cyfr [1]

Sieci konwolucyjne

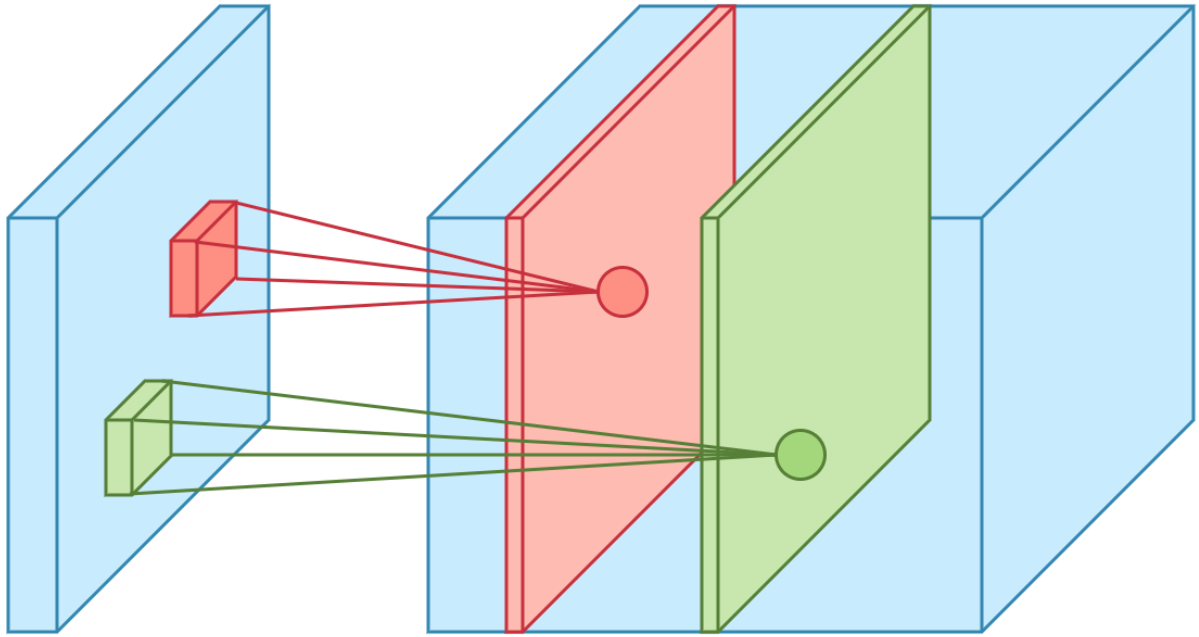
» Warstwa konwolucyjna



Konwolucja w sieciach neuronowych [2]

Sieci konwolucyjne

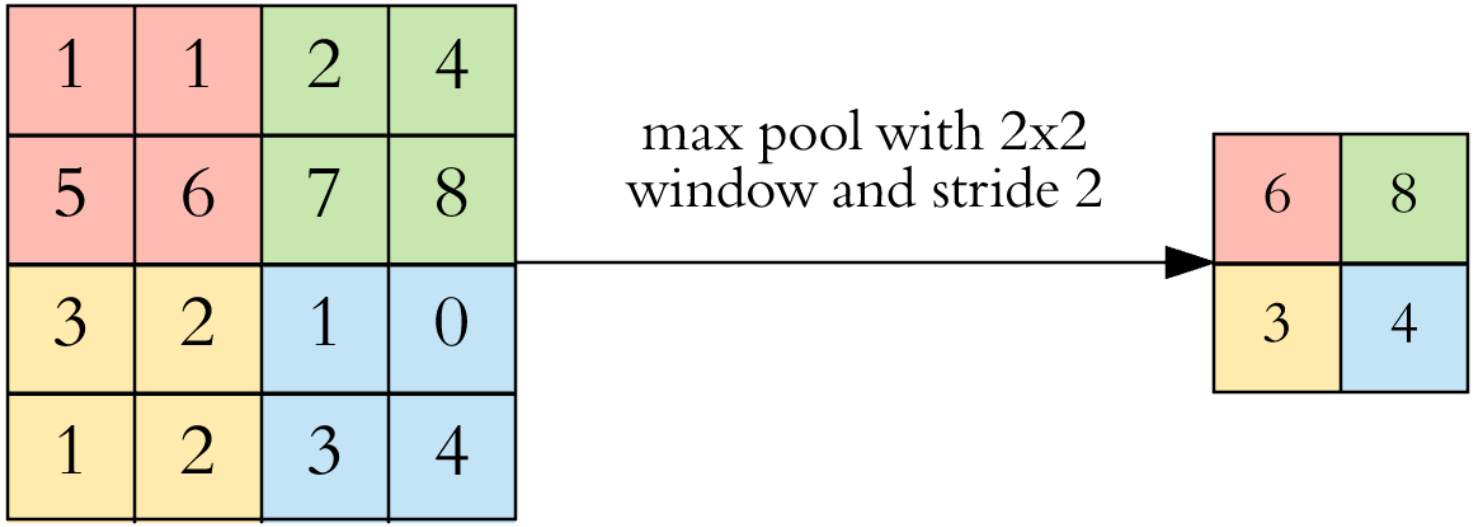
» Warstwa konwolucyjna



Każdy filtr tworzy kolejną feature mapę [2]

Sieci konwolucyjne

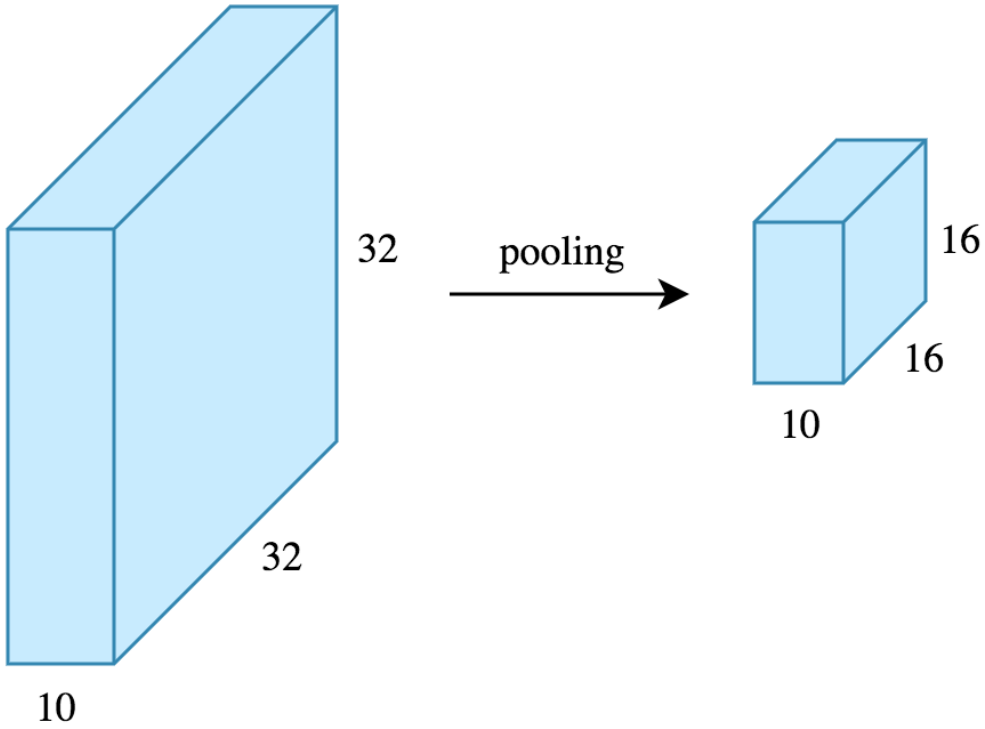
» Maxpooling



Maxpooling w sieciach neuronowych [2]

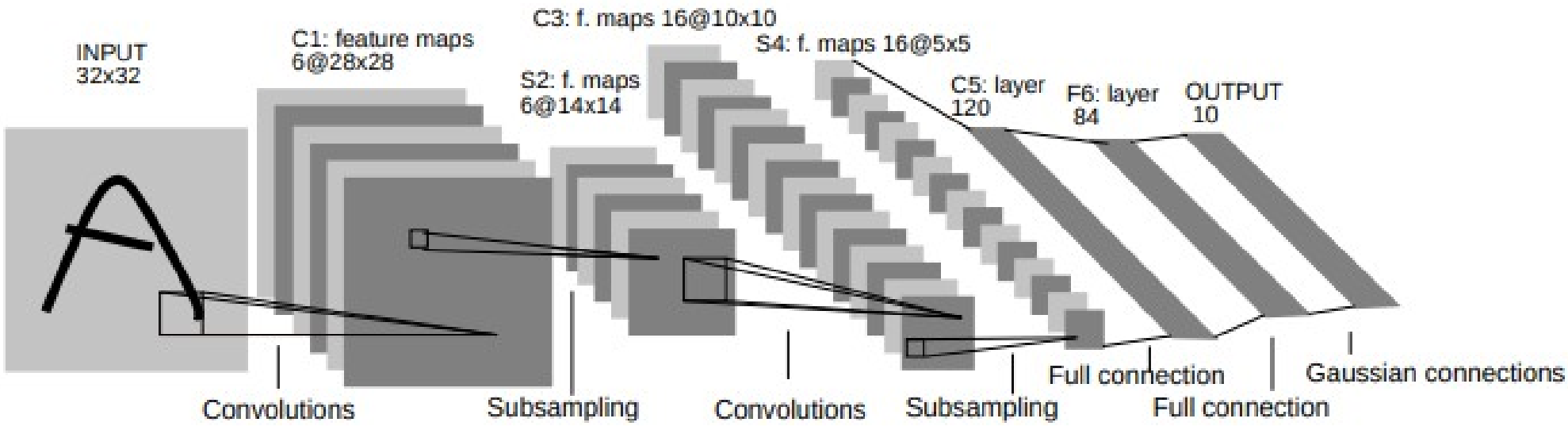
Sieci konwolucyjne

» Maxpooling



Redukcja wymiarów o połowę [2]

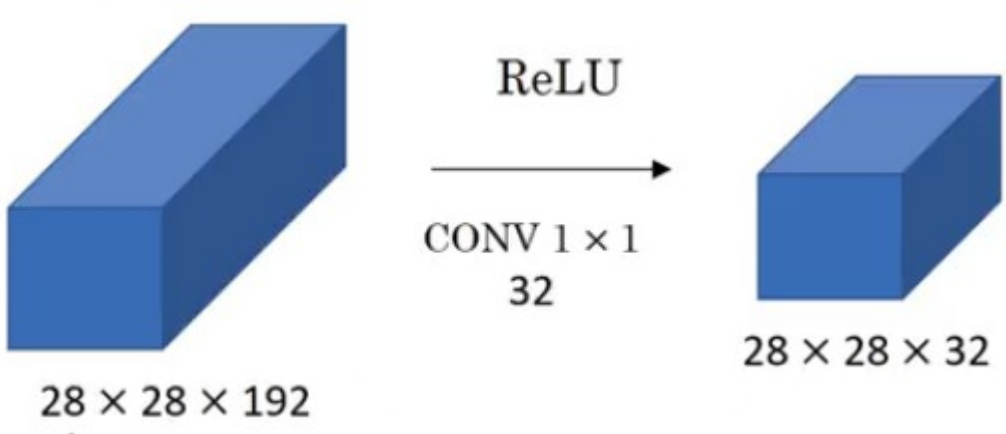
Sieci konwolucyjne



Architektura sieci do klasyfikacji odręcznie pisanych cyfr [1]

Sieci konwolucyjne

- Konwolucja 1x1
  - Pozwala zredukować rozmiar feature map w przestrzeni kanałów
  - Działa jak dense layer „w głąb”



Redukcja wymiaru dzięki operacji konwolucji 1x1 [3]

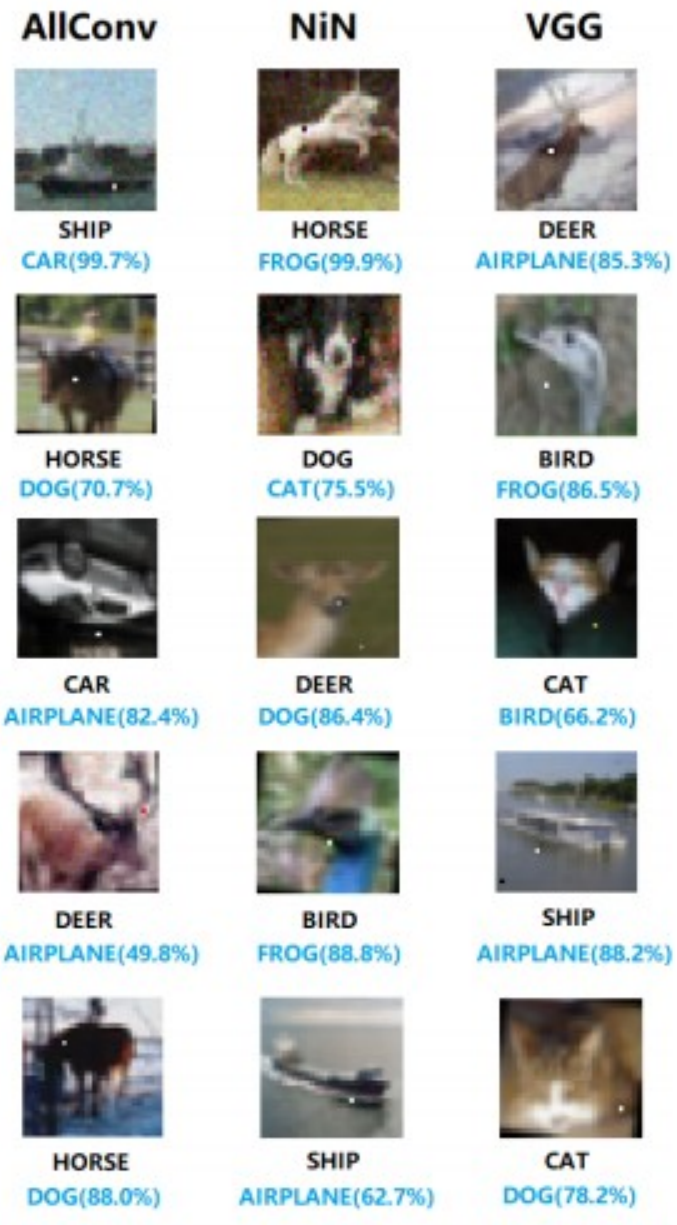
Sieci konwolucyjne

- Konsekwencje stosowania
  - Badanie wystąpienia lokalnych korelacji
  - Redukcja ilości parametrów do optymalizacji

  np.: out: 64x64x16 → 128 neuronów w warstwie dense

  ~65k * 128 ~= 8 milionów parametrów, redukcja do 4M po zastosowaniu MP, zastosowanie konwolucji 1x1, np. 8 filtrów da 32x32x8, to już 2M parametrów (oczywiście nie zawsze redukcja parametrów jest możliwa)
  - Lokalne uniezależnienie od wystąpienia cechy
  - Eliminacja z przestrzeni cech nieznaczących (w przypadku gdy szukamy wartości maksymalnych)

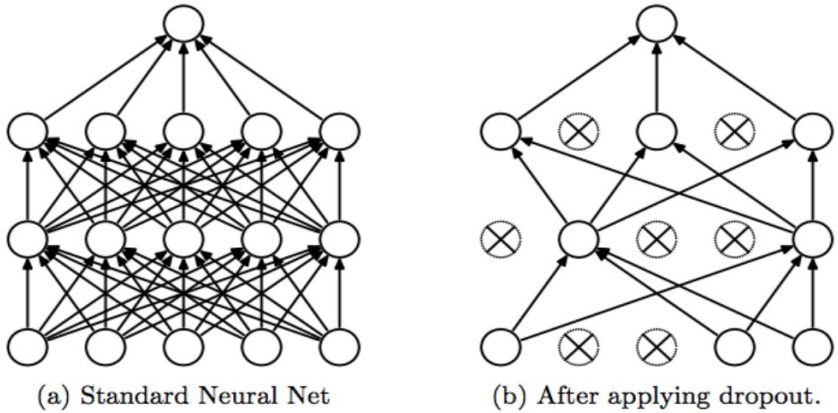
Ataki na sieci konwolucyjne



Zmiana wartości danego piksela na maksymalna, może spowodować propagowanie się tej informacji w głąb sieci powodując na końcu błędną klasyfikację [13] - One pixel attack for fooling deep neural networks.

Sieci konwolucyjne

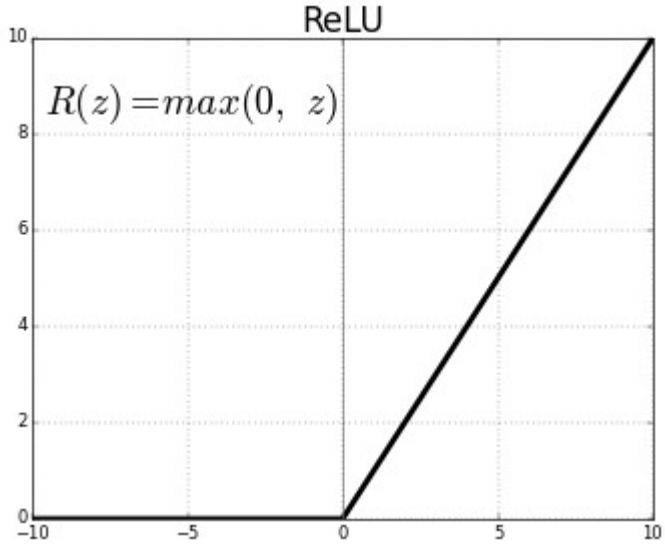
- Dropout
  - Metoda regularyzacji
  - Polega na odłączeniu losowych neuronów w celu uniezależnienia kolejnej warstwy od informacji przekazywanej przez te neurony (podobieństwo do ensemble methods)
  - Wyłączenie niektórych neuronów w trakcie uczenia powoduje, że kolejne warstwy nie polegają jedynie na informacji przekazanej z konkretnego neuronu



Schematyczne przedstawienie działania dropoutu [5]

Sieci konwolucyjne

- Funkcja aktywacji Relu
  - x dla x > 0, 0 w przeciwnym razie
  - Najczęściej występuje po warstwach konwolucyjnych
  - Choć mogłoby się wydawać (szczególnie dla obrazów), że nie wprowadza nieliniowości, jako funkcja aktywacji osobnych filtrów stanowi pewnego rodzaju zbiór liniowych aproksymacji funkcji nieliniowej



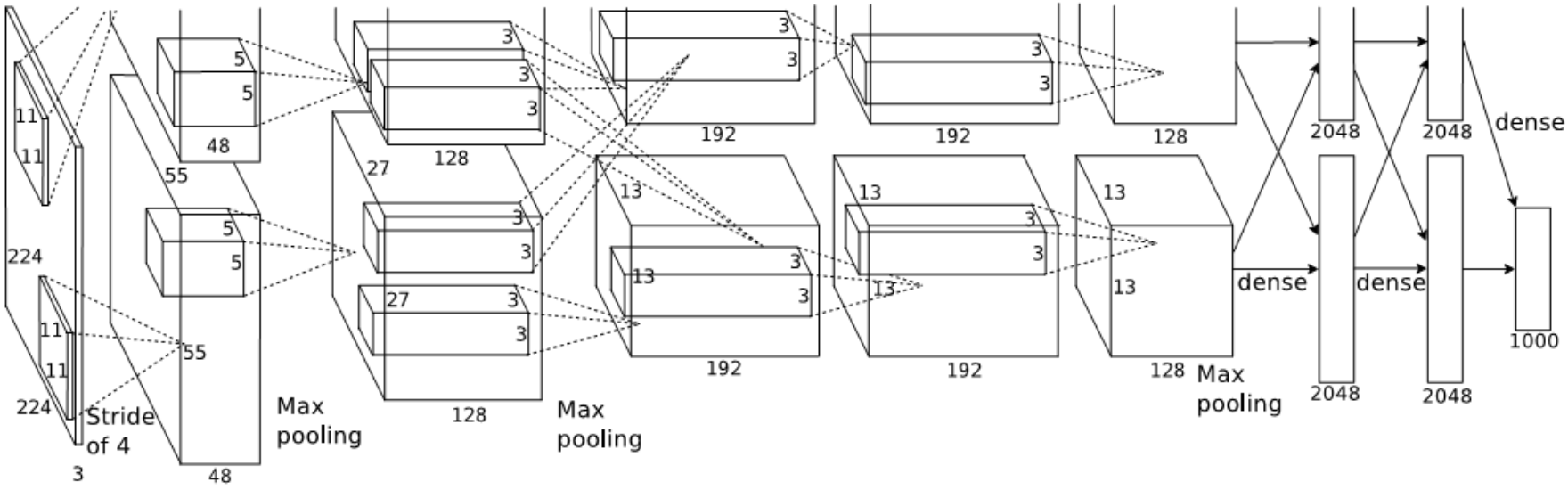
https://cdn-images-1.medium.com/max/800/1*oePAhrm74RNnNEolprmTaQ.png

Sieci konwolucyjne

- ILSVRC – ImageNet Large Scale Visual Recognition Challenge
- AlexNet – 2012
  - Konwolucja + maxpooling
  - 3 warstwy dense + softmax
  - Zaproponowanie ReLU jako funkcji aktywacji
  - Uczenie na dwóch GF GTX 580 przez 6 dni
  - Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton
  - Top-5 error – 15,3% w porównaniu do 26,2% za drugie miejsce

Sieci konwolucyjne

- ILSVRC – ImageNet Large Scale Visual Recognition Challenge
- AlexNet – 2012



Sieci konwolucyjne

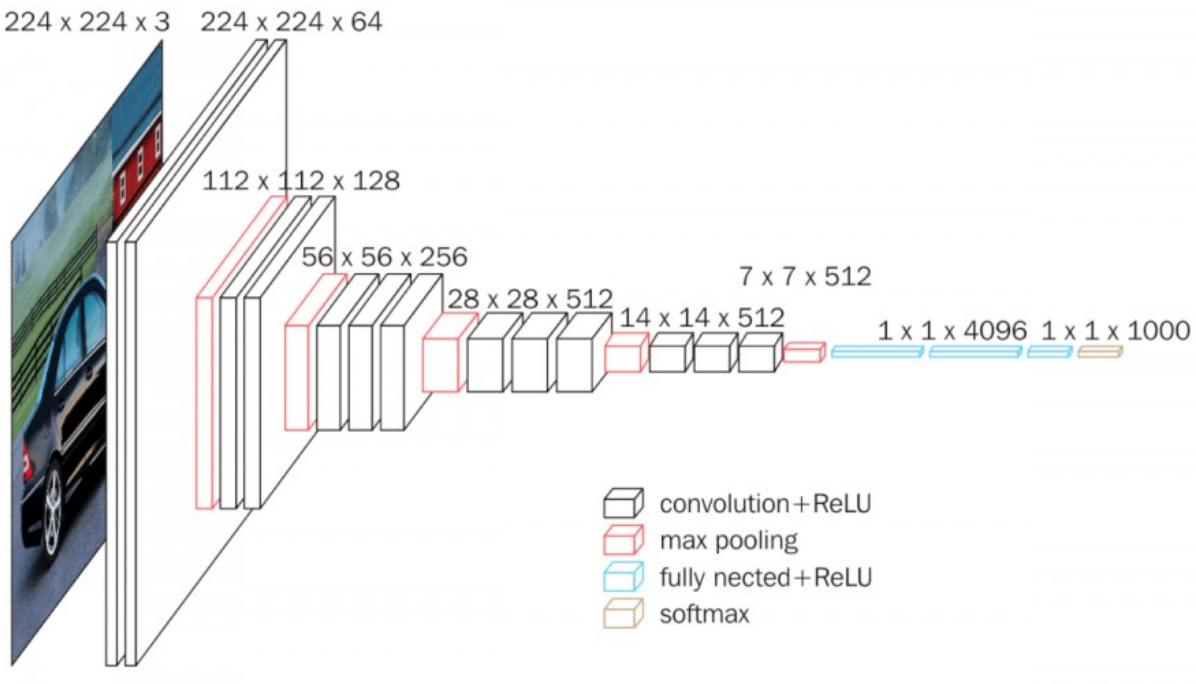
- ILSVRC – ImageNet Large Scale Visual Recognition Challenge
- ZFNet – 2013
  - Drobne usprawnienia
  - Top-5 error – 11,2%

Sieci konwolucyjne

- ILSVRC – ImageNet Large Scale Visual Recognition Challenge
- VGG – 2014
  - Zastosowanie warstw jedna po drugiej z filtrami 3x3 (szybsze uczenie, mniej parametrów do optymalizacji)
  - Im głębiej tym więcej filtrów (od 64 do 512)
  - Obecnie uważa się, że VGG jest jedną z lepszych architektur dla uczenia z transferem wiedzy (transfer learning) ze względu na swoja prostą architekturę
  - Top-5 error – 7,3%
  - I zajęła drugie miejsce

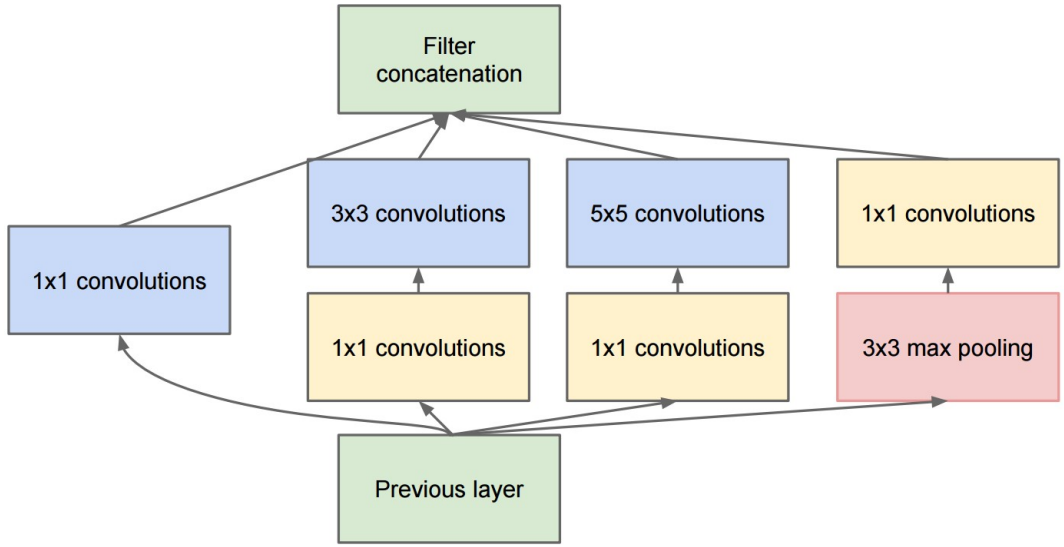
Sieci konwolucyjne

- ILSVRC – ImageNet Large Scale Visual Recognition Challenge
- VGG – 2014



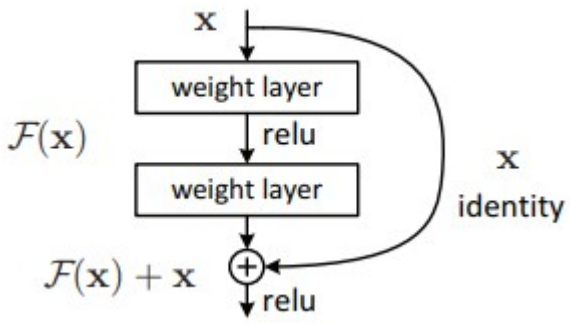
Sieci konwolucyjne

- ILSVRC – ImageNet Large Scale Visual Recognition Challenge
- GoogLeNet – 2014
  - Koncepcja incepcji – kilka bloków połączonych rownolegle – sieć uczy się swojej własne architektury
  - Konwolucja 1x1
  - Top-5 error – 6,67%



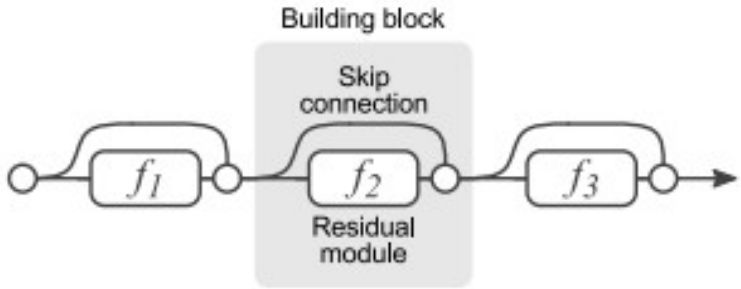
Sieci konwolucyjne

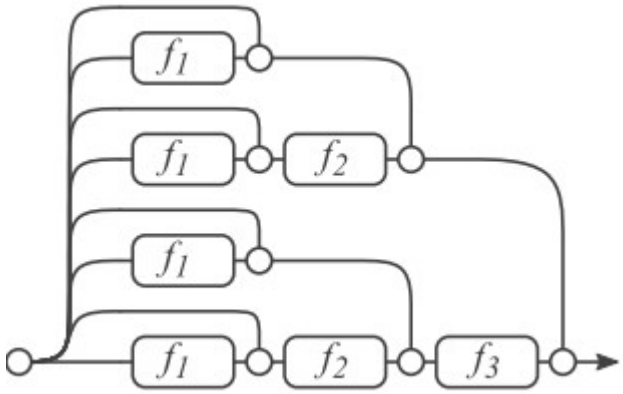
- ILSVRC – ImageNet Large Scale Visual Recognition Challenge
- ResNet – 2015
  - Residual block
  - Wyjście sieci rozgałęzia się i prowadzi do kolejnej warstwy
  - 152 warstwy
  - Top-5 error – 3,6%



Sieci konwolucyjne

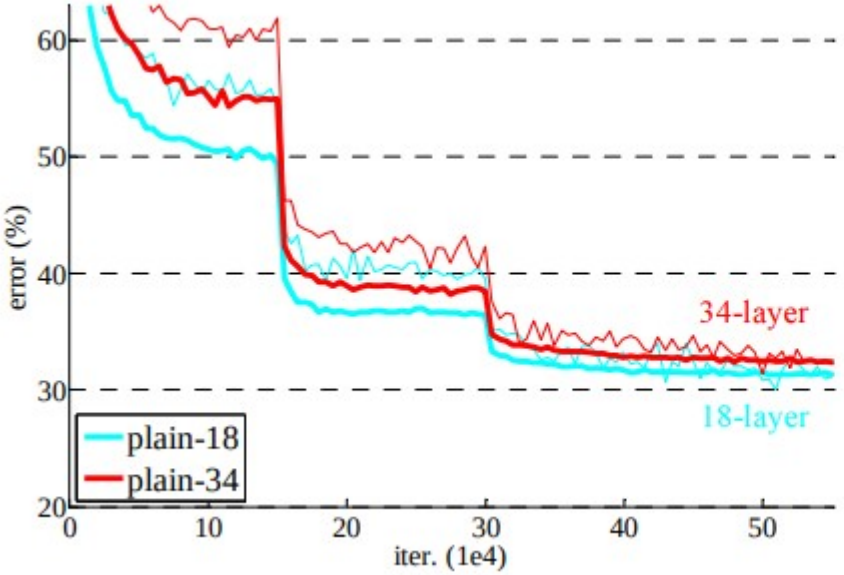
- Zestaw płytkich sieci neuronowych (ensemble) [11]
- Brak założenia hierachii warstw
- Przez co warstwy nie są od siebie zależne w takim stopniu, usuniecie jednego bloku nie pogarsza znacznie skuteczności sieci – usunięcie warstwy z sieci o architekturze typu VGG znacznie obniży skuteczność
- Rożne ścieżki gradientu w propagacji wstecznej – brak zanikania gradientu ze względu na ilość warstw

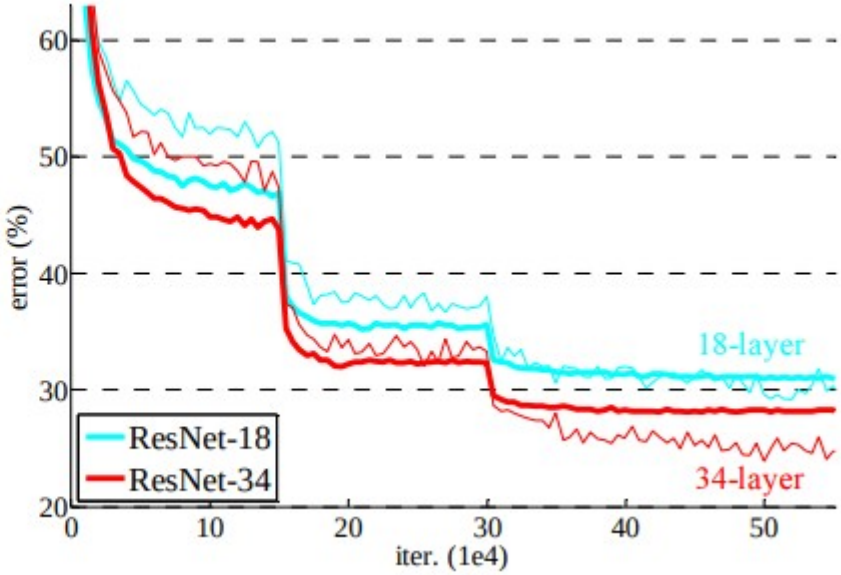




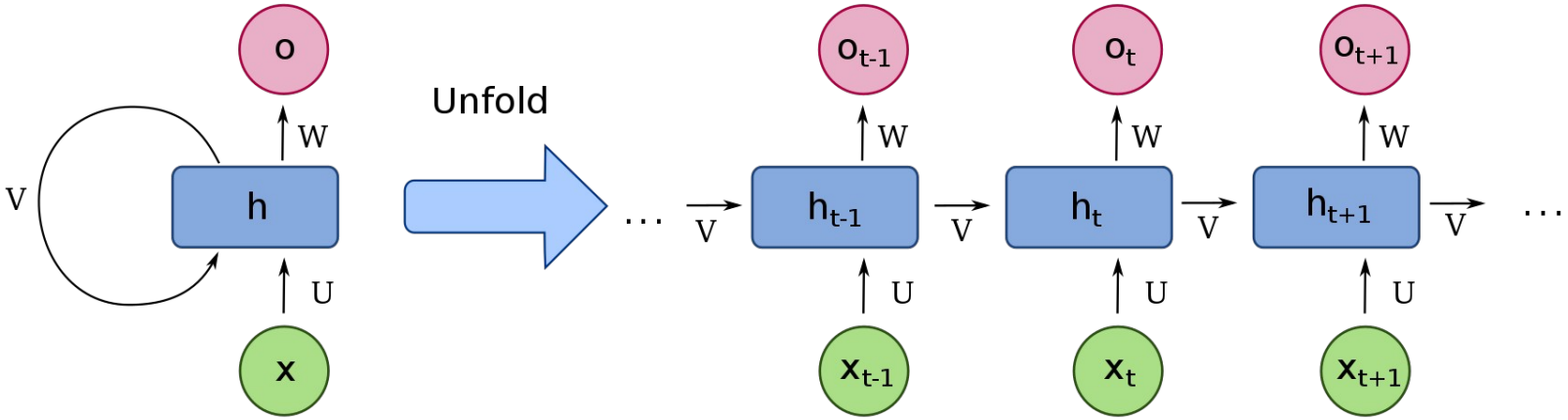
Sieci konwolucyjne

» Lepszy spadek gradientu [10]





Sieci rekurencyjne



https://en.wikipedia.org/wiki/Recurrent_neural_network

$$h^{(t)} = f\left(w_{in}^{(t)} i^{(t)} + w_{rec}^{(t-1)} h^{(t-1)}\right)$$ [5]

Sieci rekurencyjne

$$\frac{\partial h^{(t)}}{\partial i^{(t-k)}} = f'\left(w_{in}^{(t)} i^{(t)} + w_{rec}^{(t-1)} h^{(t-1)}\right) \frac{\partial}{\partial i^{(t-k)}} \left(w_{in}^{(t)} i^{(t)} + w_{rec}^{(t-1)} h^{(t-1)}\right)$$

$$\frac{\partial h^{(t)}}{\partial i^{(t-k)}} = f'\left(w_{in}^{(t)} i^{(t)} + w_{rec}^{(t-1)} h^{(t-1)}\right) w_{rec}^{(t-1)} \frac{\partial h^{(t-1)}}{\partial i^{(t-k)}}$$

$$\left|\frac{\partial h^{(t)}}{\partial i^{(t-k)}}\right| \leq \left|w_{rec}^{(t-1)}\right| \cdot \left|\frac{\partial h^{(t-1)}}{\partial i^{(t-k)}}\right|$$

$$\left|\frac{\partial h^{(t)}}{\partial i^{(t-k)}}\right| \leq \left|w_{rec}^{(t-1)}\right| \cdot \ldots \cdot \left|w_{rec}^{(t-k)}\right| \cdot \left|\frac{\partial h^{(t-k)}}{\partial i^{(t-k)}}\right|$$

[5]

Sieci rekurencyjne

$$h^{(t-k)} = f\left(w_{in}^{(t-k)} i^{(t-k)} + w_{rec}^{(t-k-1)} h^{(t-k-1)}\right)$$

$$\frac{\partial h^{(t-k)}}{\partial i^{(t-k)}} = f'\left(w_{in}^{(t-k)} i^{(t-k)} + w_{rec}^{(t-k-1)} h^{(t-k-1)}\right) \frac{\partial}{\partial i^{(t-k)}} \left(w_{in}^{(t-k)} i^{(t-k)} + w_{rec}^{(t-k-1)} h^{(t-k-1)}\right)$$

$$\frac{\partial h^{(t-k)}}{\partial i^{(t-k)}} = f'\left(w_{in}^{(t-k)} i^{(t-k)} + w_{rec}^{(t-k-1)} h^{(t-k-1)}\right) w_{in}^{(t-k)}$$

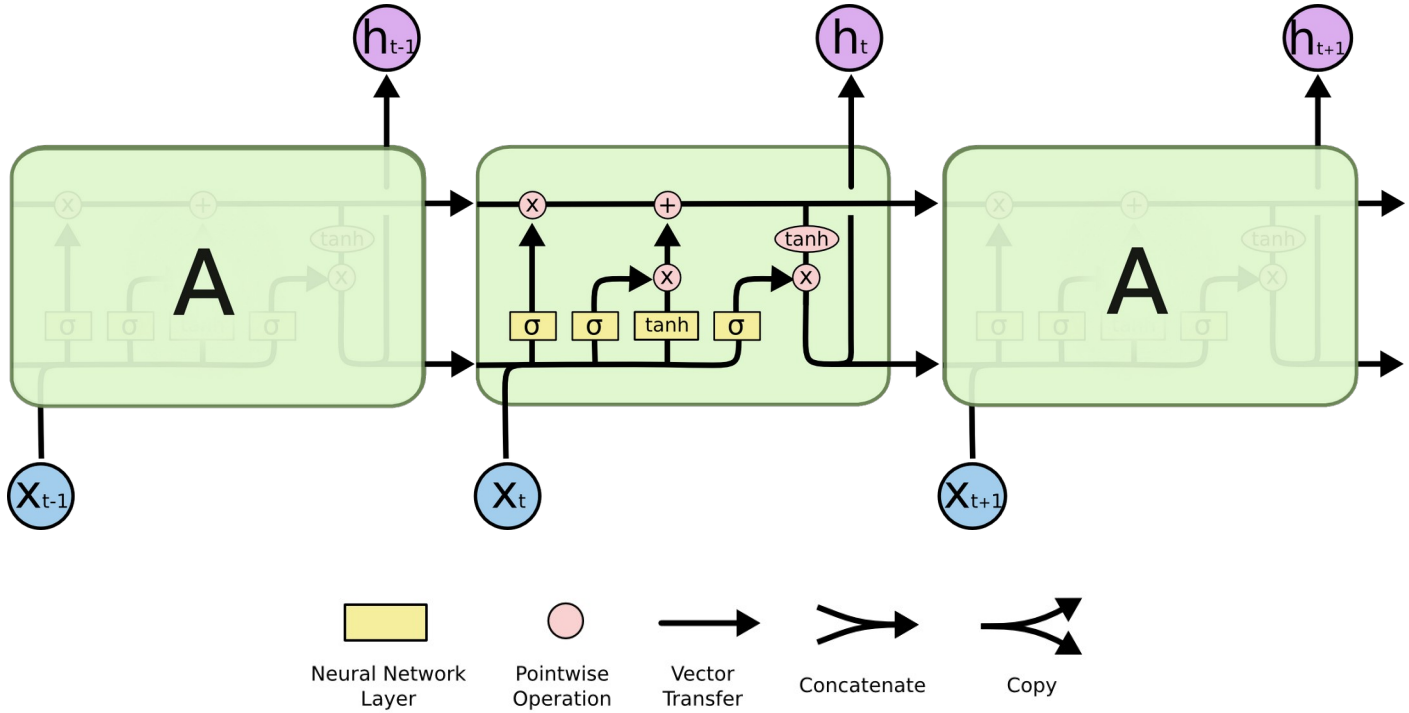
$$\left|\frac{\partial h^{(t-k)}}{\partial i^{(t-k)}}\right| \leq \left|w_{in}^{(t-k)}\right|$$

$$\left|\frac{\partial h^{(t)}}{\partial i^{(t-k)}}\right| \leq \left|w_{rec}^{(t-1)}\right| \cdot \ldots \cdot \left|w_{rec}^{(t-k)}\right| \cdot \left|w_{in}^{(t-k)}\right| = \left|w_{rec}\right|^{k} \cdot w_{in}$$

[5]

Sieci rekurencyjne

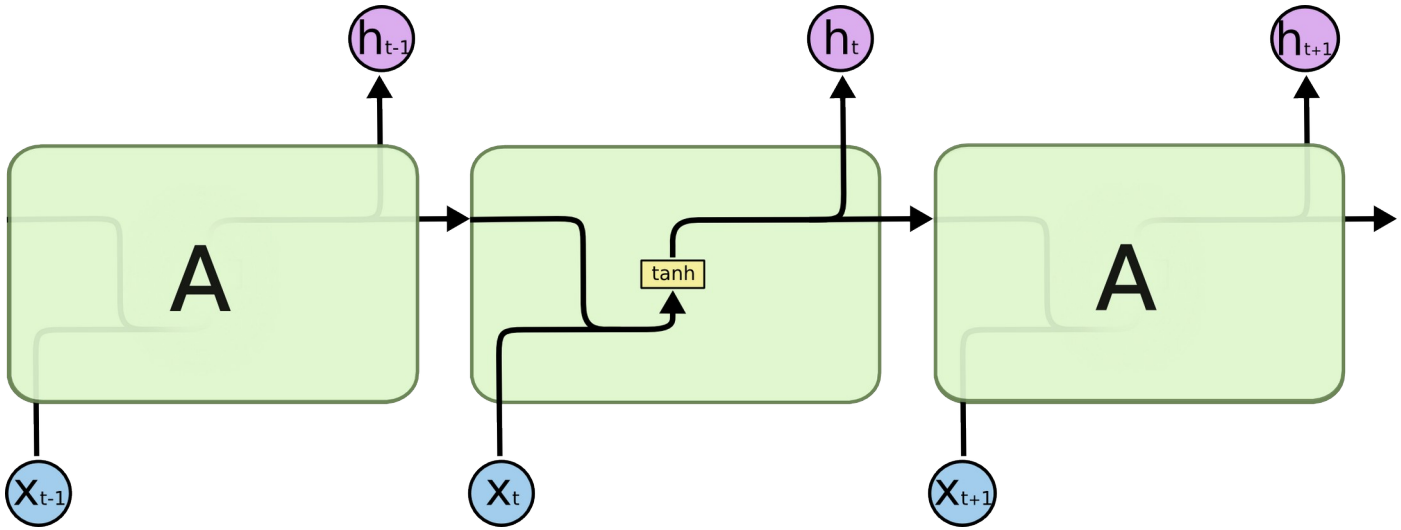
» Long-short term memory unit [7]



Struktura LSTM [8]

Sieci rekurencyjne

» Klasyczne sieci rekurencyjne

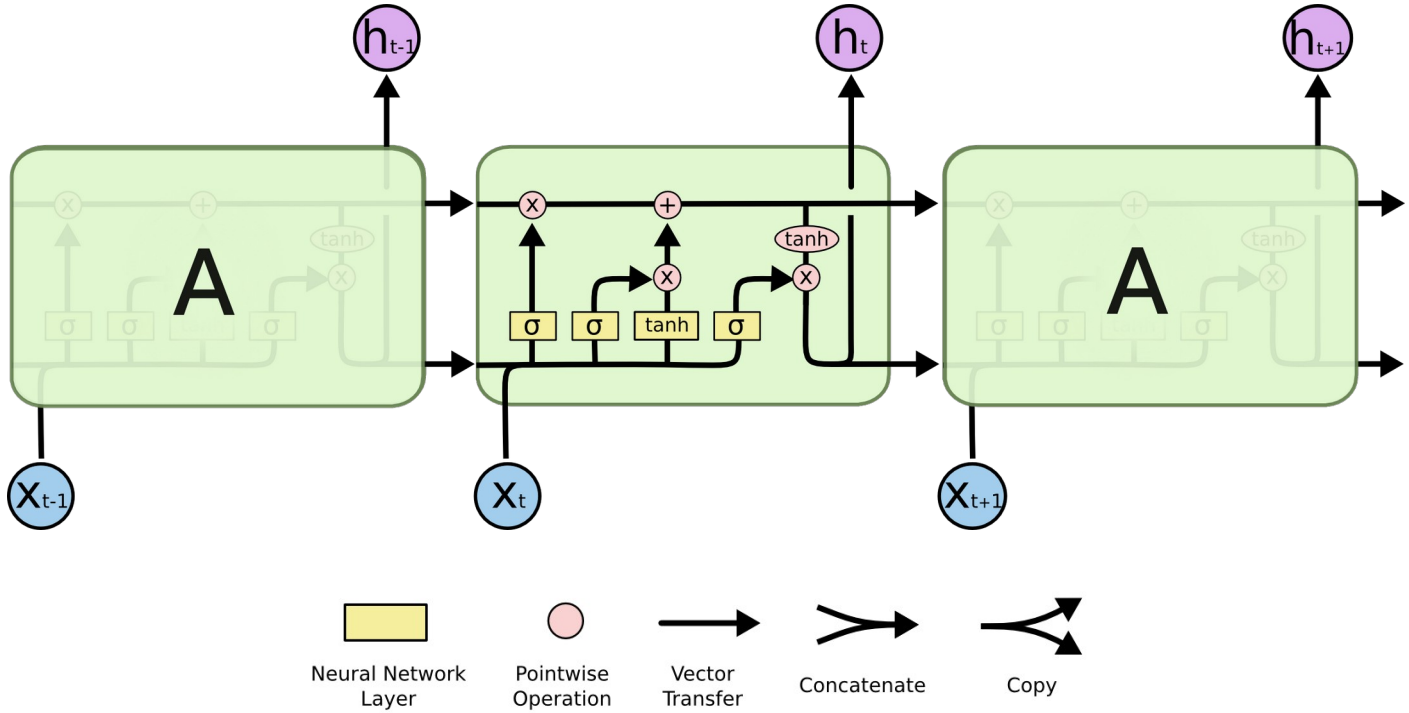


$$h^{(t)} = f\left(w_{in}^{(t)} i^{(t)} + w_{rec}^{(t-1)} h^{(t-1)}\right)$$

Vanilla RNN [8]

Sieci rekurencyjne

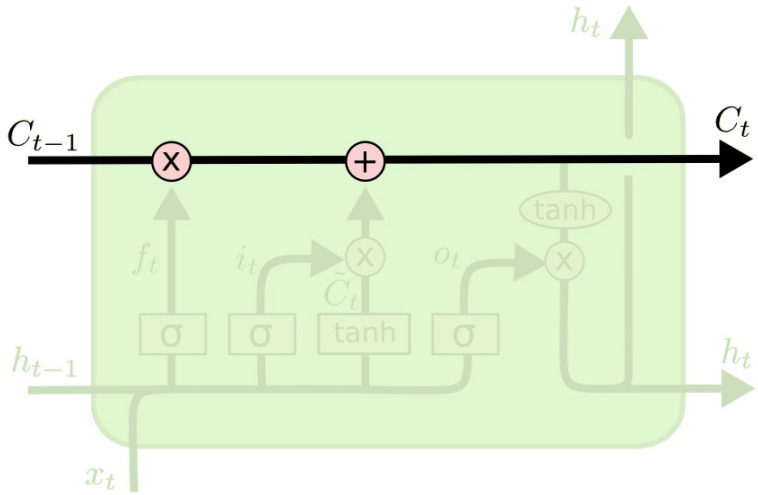
» Long-short term memory unit [7]



Struktura LSTM [8]

Sieci rekurencyjne

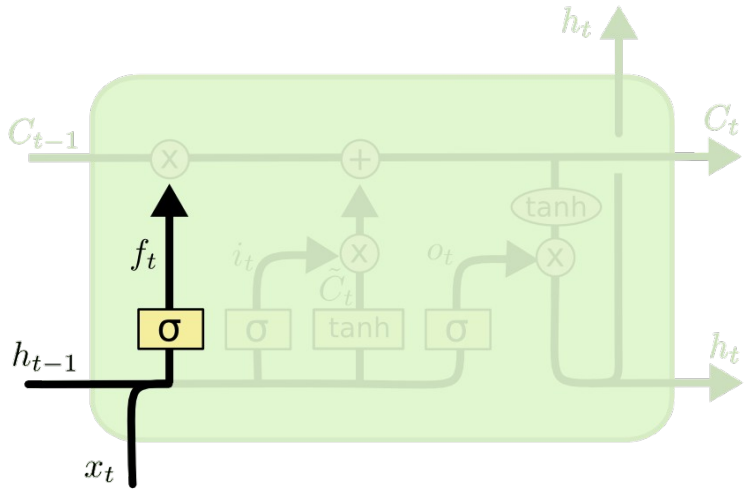
» Long-short term memory unit



Cell state [8]

Sieci rekurencyjne

» Long-short term memory unit

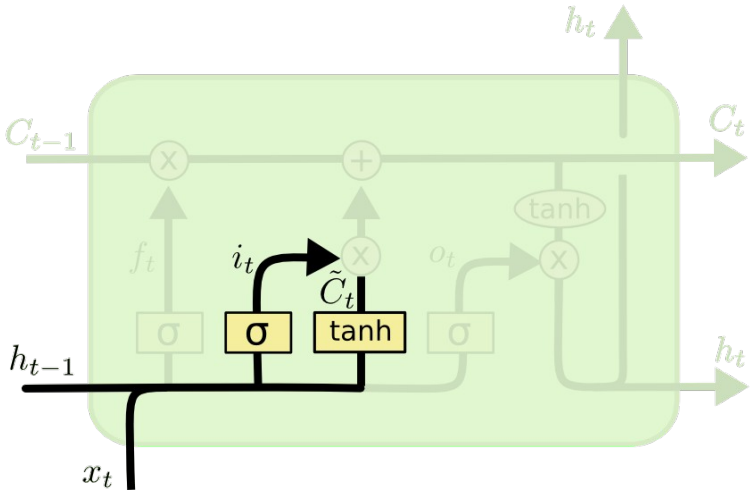


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Cell state [8]

Sieci rekurencyjne

» Long-short term memory unit



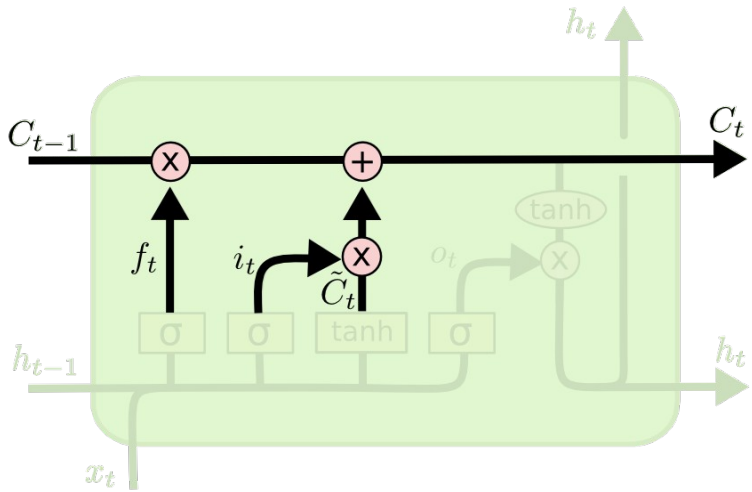
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Cell state [8]

Sieci rekurencyjne

» Long-short term memory unit

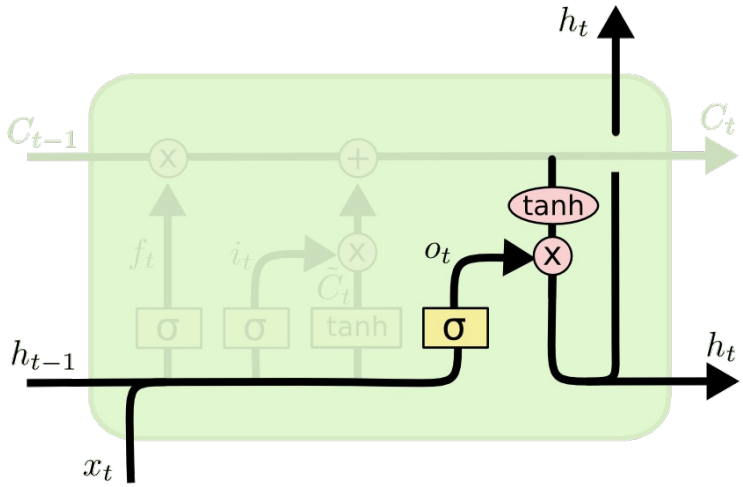


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Cell state [8]

Sieci rekurencyjne

» Long-short term memory unit



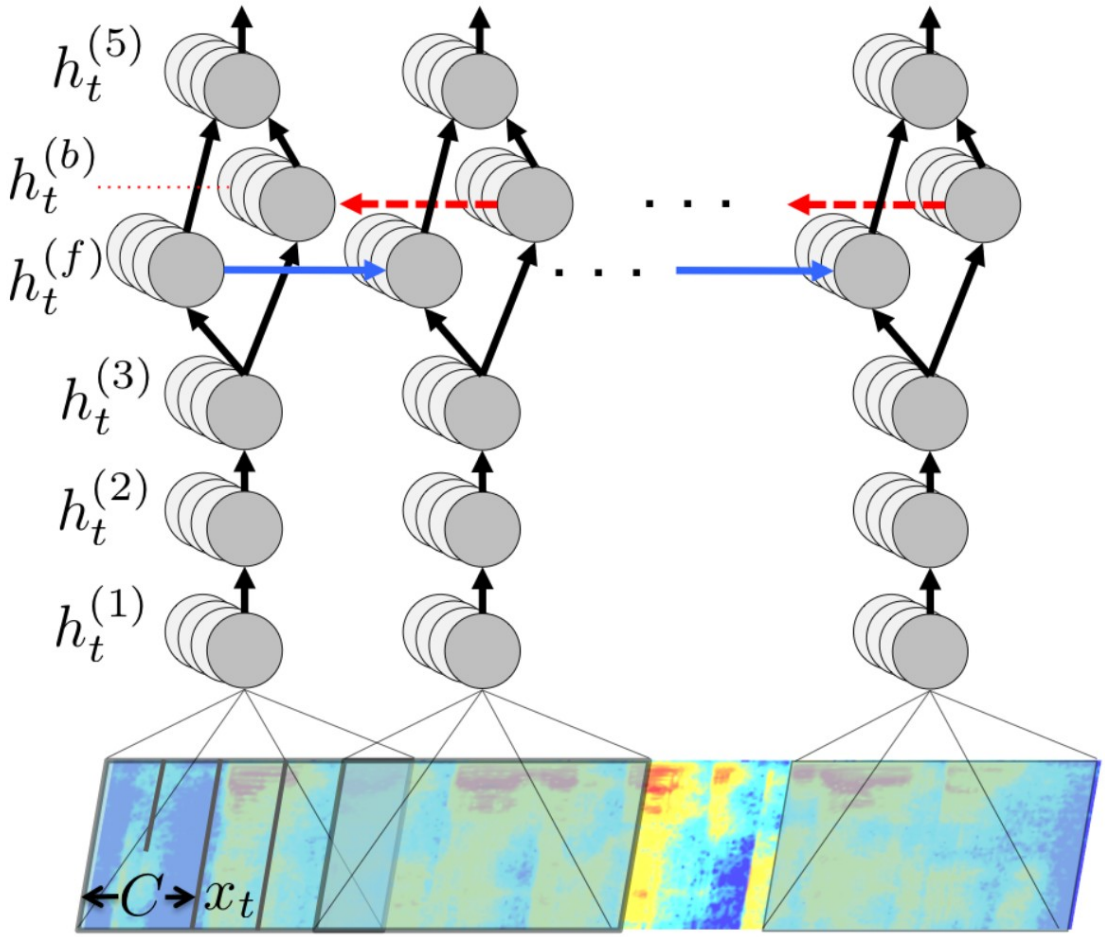
Cell state [8]

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

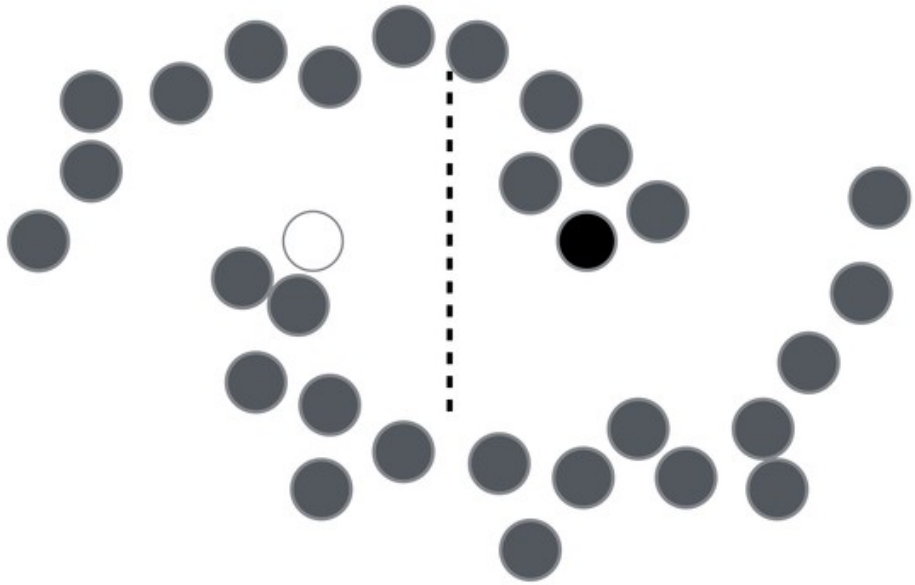
Deepspeech

» Transkrypcja wypowiedzi na podstawie spektrogramu



Uczenie półnadzorowane

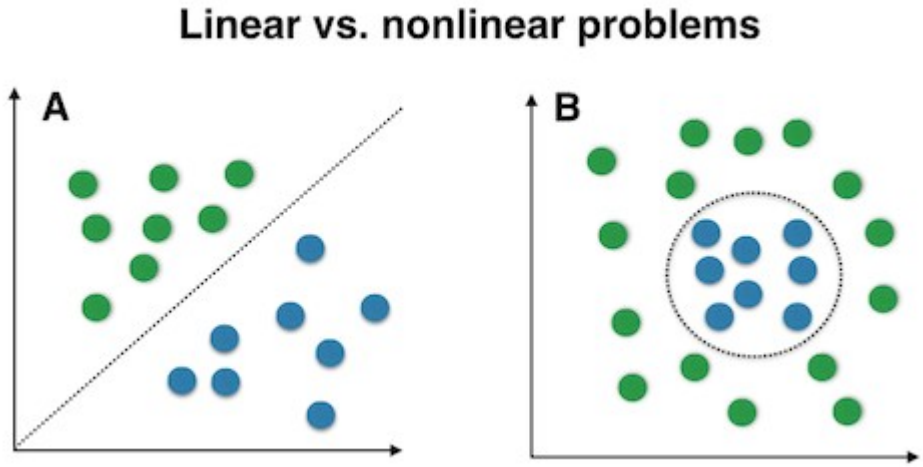
- Trudność w pozyskaniu opisanych danych (sieci neuronowe potrzebują dużo danych aby nie overfitować)
- Obecność nie opisanych danych poprawia skuteczność klasyfikacji



Potacjalny overfitting w przypadku użycia nie wystarczającej ilości danych [9]

Uczenie półnadzorowane

- Jednym z podejść może być zastosowanie PCA
- Jednak PCA nie radzi sobie z nieliniowymi danymi



https://sebastianraschka.com/images/blog/2014/kernel_pca/linear_vs_nonlinear.png

Autoenkodery

Sieć neuronowa, używana do efektywnego kodowania danych wejściowych (odnalezienia reprezentacji charakteryzującej dany zbiór, często o mniejszym wymiarze niż wejście) w sposób nienadzorowany.

Możemy wyróżnić enkoder (część sieci odpowiedzialna za kodowanie), który dokonuje transformacji wejścia do pewnej ukrytej reprezentacji z (stanowiącej pewnego rodzaju kod):

$$z = f_\theta(x) = \sigma(Wx + b)$$

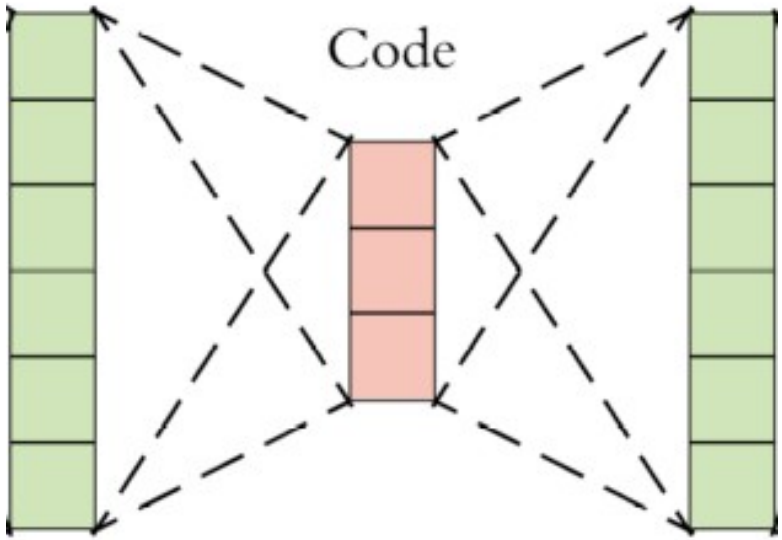
Oraz dekoder, który próbuje otworzyć te dane wejściowe na podstawie ich ukrytej reprezentacji z

$$x' = f_{\theta'}(h) = \sigma'(W'z + b')$$

Minimalizując funkcję błędu między wejściem i rekonstrukcją.

W, W', b, b' - parametry sieci; σ, σ' – funkcje aktywacji

Autoenkodery



Przykład autoenkodera[1]

Dla W’ = W^T w początkowej fazie uczenia – połowa mniej parametrów do optymalizacji

Dziękuję za uwagę

Bibliografia

- [1] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, 86(11):2278-2324, November 1998
- [2] https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2
- [3] https://www.analyticsvidhya.com/blog/2018/12/guide-convolutional-neural-network-cnn/
- [4] https://towardsdatascience.com/single-neuron-training-3fc7f84d67d
- [5] Srivastava, Nitish, et al. ”Dropout: a simple way to prevent neural networks from overfitting”, JMLR 2014
- [6] Buduma N., Fundamentals of Deep Learning, June 2017: First Edition, O’Reilly Media Inc.
- [7] Hochreiter, Sepp & Schmidhuber, Jürgen. (1997). Long Short-term Memory. Neural computation. 9. 1735-80. 10.1162/neco.1997.9.8.1735.
- [8] http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- [9] https://towardsdatascience.com/simple-explanation-of-semi-supervised-learning-and-pseudo-labeling-c2218e8c769b
- [10] He, Kaiming, Xiangyu Zhang, Shaoqing Ren and Jian Sun. “Deep Residual Learning for Image Recognition.” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016): 770-778.
- [11] Veit, Andreas & Wilber, Michael & Belongie, Serge. (2016). Residual Networks Behave Like Ensembles of Relatively Shallow Networks. Advances in Neural Information Processing Systems.
- [12] https://medium.com/@AI_with_Kain/understanding-of-multilayer-perceptron-mlp-8f179c4a135f
- [13] One pixel attack for fooling deep neural networks, Su, Jiawei; Vasconcellos Vargas, Danilo; Kouichi, Sakurai
- [14] Deep Speech: Scaling up end-to-end speech recognition, Hannun A., et al.