Deep Learning

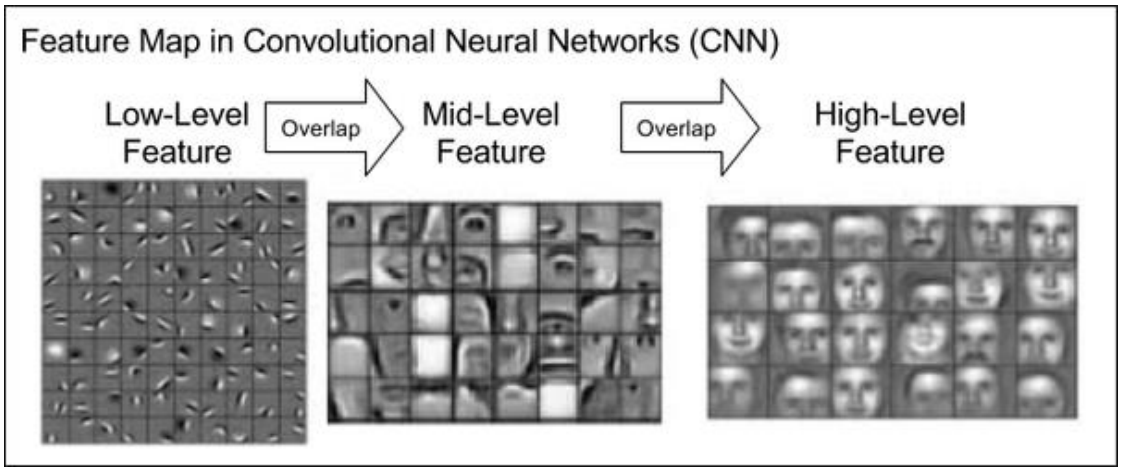
- **Multi-layer (L-layer) neural networks**

Zbiory danych

Podstawowe typy zbiorów danych:

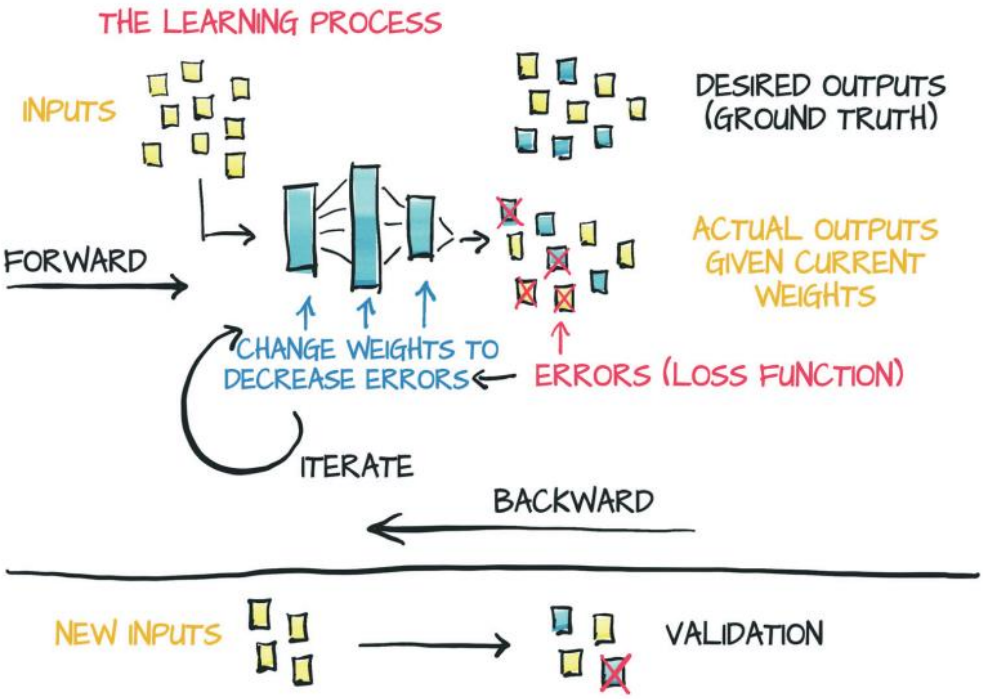
- **Treningowy**: dane używane do uczenia sieci
- **Walidacyjny**: dane używane do “unbiased” (nie stronniczej) ewaluacji modelu podczas tuningu hiperparametrów. Są one de facto dopasowywane do zbioru ewaluacynego
- **Testowy**: dane używane do “unbiased” ewaluacji ostatecznego modelu (jak nauczył się generalizować)

Sieci głębokie

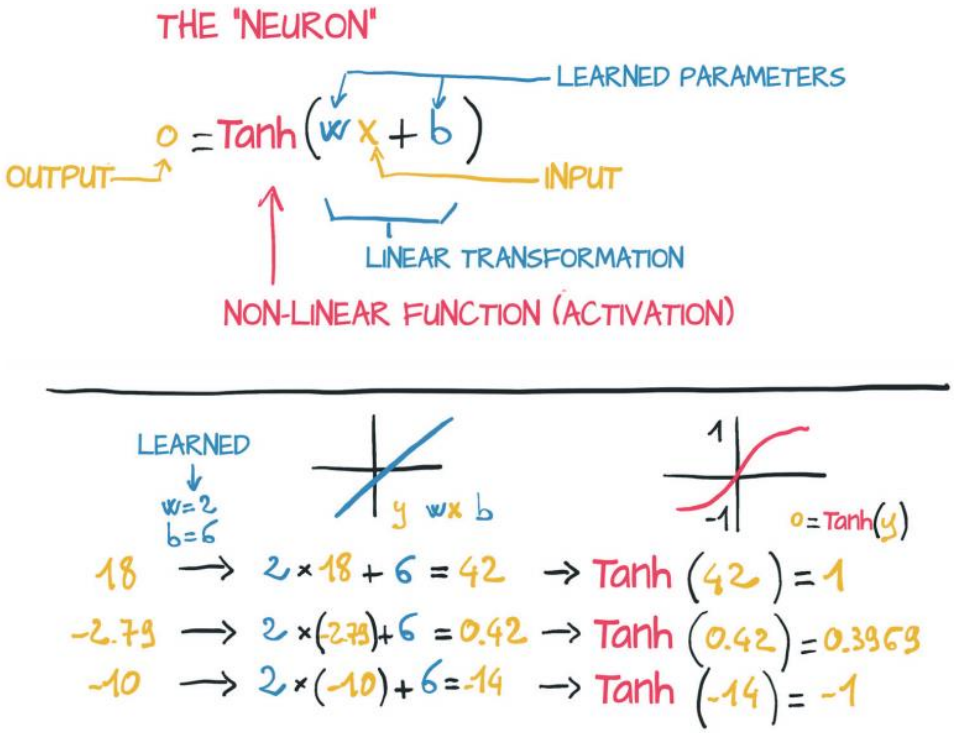


Uproszczona reprezentacja wyszukiwania wzorców przez sieć głęboką. Pierwsza warstwa poszukuje prostych wzorców, np. krawędzi. Kolejna warstwa wyszukuje konkretne części twarzy, np. Część nosa. Ostania warstwa łączy wyniki w twarz. Skomplikowanie rozpoznawanych wzorców rośnie wraz z warstwami.

- Warstwy bliższe wejściu sieci uczą się znajdować ogólne cechy (*features)* takie jak krawędzie. Następne warstwy łączą wyniki poprzedniej, stąd skomplikowanie rośnie. Ogólna reguła mówi, aby następna warstwa miała mniej neuronów niż poprzednia. Dzięki temu sieć nauczy się kompresować informacje i generalizować cechy.



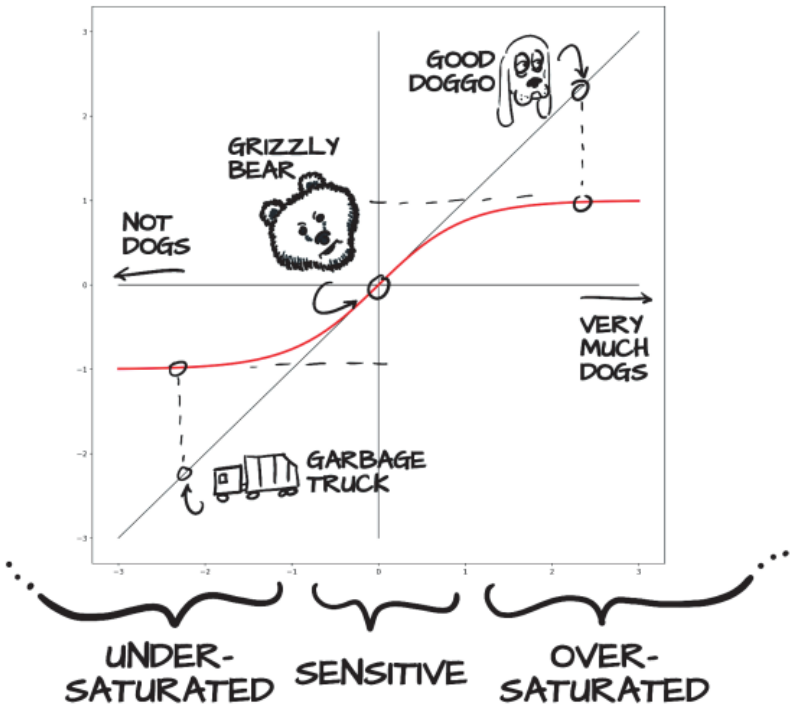
Proces uczenia sieci neuronowej.



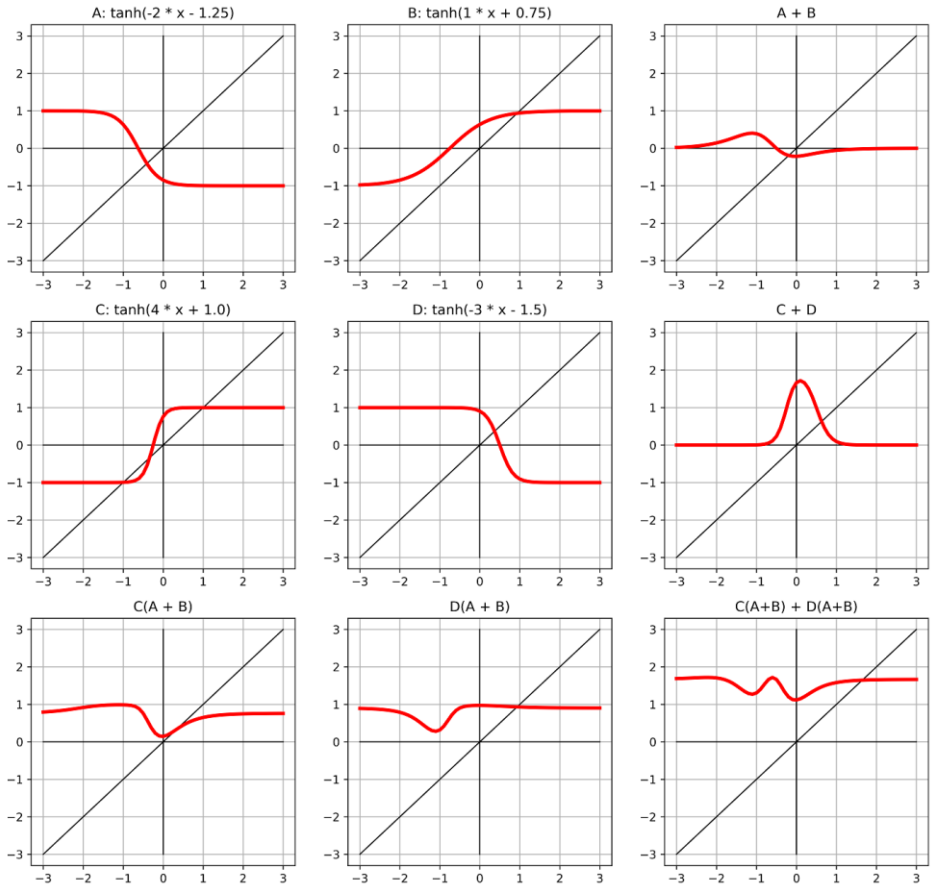
Pojedynczy neuron. ***X*** - wejście, ***W*** – wagi, ***b*** – bias (przesunięcie). ***Tanh*** jest funkcją aktywacji *g*.

X, b *oraz* ***o*** *(*u nas ***y*** lub ***a***) mogą być skalarami lub wektorami. Podobnie ***W*** może być wektorem lub macierzą (innymi słowy wyliczenia możemy przeprowadzać dla pojedynczego neuronu lub **całej warstwy jednocześnie**). Obliczenia na wektorach i macierzach są efektywniejsze aniżeli przeliczanie wartości dla każdego przypadku uczącego.

Funkcje aktywacji cd.



- Liniowa operacja *(wagi + bias*) może zwrócić dowolną wartość rzeczywistą. Potrzebujemy skoncentrować wyjście (*z*) do określonego zbioru wartości, który będzie reprezentacyjny - stąd funkcje aktywacji.
- Funkcje aktywacji są:
    - Nieliniowe – pozwala to na przybliżanie skomplikowanych funkcji
    - Różniczkowalne - aby można było wyliczyć dla nich gradienty.



A, B, C i D reprezentują pojedyncze neurony. A + B reprezentuje połączenie dwóch neuronów, a C(A + B) - kompozycję neuronów (połączenie dwóch warstw). C(A+B) + D(A+B)

- Głęboka sieć neuronowa potrafi przybliżyć dowolnie skomplikowaną funkcję poprzez połączenie wielu funkcji aktywacji poprzedzonych liniowymi transformacjami.
- Model danych z początku jest nie znany - sieć “odkrywa” model reprezentujący dane na podstawie danych oraz *cost function*

Forward Propagation

- Dla pojedynczej pary uczącej *x*:
  - Dla pierwszej warstwy:

$$a^{[1]} = g^{[1]}(z^{[1]}) = g^{[1]}(W^{[1]}x + b^{[1]})$$

    - *g*– funkcja aktywacji, *W* - macierz parametrów, *b - bias vector, x = $a^{[0]}$*
  - Dla drugiej warstwy:

$$a^{[2]} = g^{[2]}(z^{[2]}) = g^{[2]}(W^{[2]}a^{[1]} + b^{[2]})$$

  - Itd... Wyjście warstwy ostatniej: *y_hat = $a^{[l]}$*
- Wszystkie powyższe obliczenia należy wykonać w pętli dla wszystkich par uczących. Wersja zwektoryzowana wygląda następująco:

$$A^{[l]} = g^{[l]}(Z^{[l]}) = g^{[l]}(W^{[l]}A^{[l-1]} + b^{[l]})$$

- Powyższe wyliczenia należy zastosować oczywiście dla każdej warstwy (tego nie możemy przyspieszyć, musimy zastosować pętlę *for* po warstwach)

$$W^{[l]}: (n^{[l]}, n^{[l-1]})$$

Rozmiar macierzy W: tyle wierszy, ile neuronów w l-tej warstwie oraz kolumn, ile elementów wejściowych

Backward Propagation

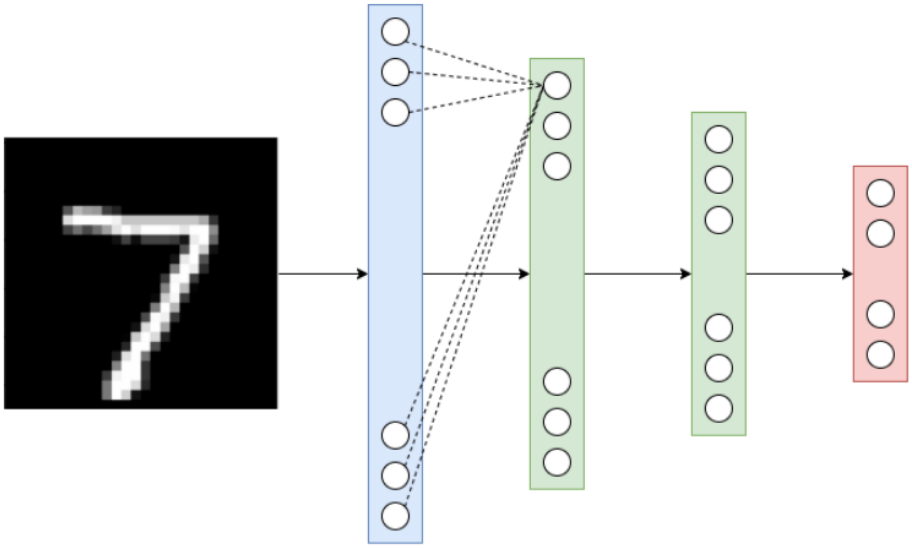
https://www.youtube.com/watch?v=Ilg3gGewQ5U

https://www.youtube.com/watch?v=tIeHLnjs5U8

Parametry vs Hiperparametry

- Parametry:
  - W,
  - b
- Hiperparametry
  - Learning rate,
  - Liczba iteracji w GD,
  - Liczba warstw ukrytych,
  - Liczba neuronów w warstwach,
  - Funkcja aktywacji
- Implementowanie metod DL jest procesem empirycznym. Wyniki polepszamy poprzez eksperymentację wartościami hiperparametrów.

Zadanie



Obrazek 28x28, dwie warstwy ukryte.

Multi_layer_nn.py - kod programu