W ramach naszych poniedziałkowych spotkań poszerzania horyzontów

Ten zestaw ze sobą działa:

Wersja tensorflow: 2.18.0
Wersja pythona: 3.12.13
Wersja CUDA Toolkit: 12.4

Znane twierdzenia o sieciach neuronowych:

Twierdzenie o uniwersalnej aproksymacji:
- dowolny problem można rozwiązać za pomocą 1 warstwy ukrytej i pewnej nie nieliniowej funkcji aktywacji

Efektywność optymalizacji powinna być gradientowa.
- im większy błąd tym tym optymalizacja powinna być bardziej agresywna. Choć matematycy zwykle skupiają się na problemie osiągania minimów lokalnych zamiast globalnych.

Efektywność sieci głębokich jest większa niż płytkich:
- dowody (matematyczne i praktyczne) pokazują, że sieci wielowarstwowe, uczą się efektywniej niż sieci jednowarstwowe, w których liczba neuronów musiała by być ogromna.

Twierdzenie mówiące o tym, że dla sieci neuronowej istnieją optymalne wartości wag. Ale problem jest z gatunku trudnych, bo sieć może wpadać w minima lokalne zamiast globalnego.

Warunek Lipschitza, mówi o tym, że niektóre funkcje aktywacji dają ograniczoną regularność sieci, a spełnienie warunku Lipschitza zapewnia stabilność sieci.

$$|f(x_1) - f(x_2)| \leq L \cdot |x_1 - x_2|$$

Rodzaje sieci:

Typ sieci	Zastosowanie
Sequential	Proste modele, dane tabelaryczne, klasyfikacja
Functional API	Modele wielowymiarowe, hybrydowe
CNN	Obrazy, przetwarzanie danych przestrzennych
RNN, LSTM, GRU	Prognozy szeregów czasowych, przetwarzanie sekwencji
Hybrydowe (CNN + RNN)	Sekwencje obrazów, audio, wideo
GAN	Generowanie danych, obrazów
ResNet	Bardzo głębokie sieci neuronowe

Rodzaje warstw:

Typ danych	Popularne warstwy
Dane proste	Dense, Dropout, BatchNormalization, Activation, LayerNormalization
Szeregi czasowe	LSTM, GRU, SimpleRNN, Bidirectional, Conv1D, TimeDistributed
Obrazy	Conv2D, MaxPooling2D, Flatten
Dane tekstowe	Embedding, LSTM, GRU, Bidirectional

Popularne funkcje aktywacji dla warstw ukrytych:

Funkcja aktywacji	Zastosowanie	Zadanie	Wzór i właściwości
ReLU (Rectified Linear Unit)	Sieci neuronowe (CNN, RNN), zadania ogólne	Klasyfikacja, regresja	$\mathrm{ReLU}(x) = \max(0, x)$
Leaky ReLU	Gdy ReLU powoduje martwe neurony	Klasyfikacja, regresja	$\mathrm{Leaky\ ReLU}(x) = x \text{ dla } x > 0\ \alpha x \text{ dla } x \leq 0$
tanh	Sieci rekurencyjne (RNN, LSTM), przetwarzanie tekstu	Klasyfikacja, przetwarzanie sekwencji	$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
Sigmoid	Gdy potrzebujesz wyjścia w zakresie [0, 1]	Klasyfikacja binarna	$\sigma(x) = \frac{1}{1+e^{-x}}$
Swish	Sieci głębokie (transformery, modele NLP)	Klasyfikacja, zadania z dużą liczbą cech	$\mathrm{Swish}(x) = x \cdot \sigma(x)$

Popularne funkcje aktywacji stosowane w warstwach wyjściowych:

Funkcja aktywacji	Zastosowanie	Rodzaj zadania	Zakres wartości
Sigmoid	Klasyfikacja binarna	Wyjście w postaci prawdopodobieństwa	[0, 1]
Softmax	Klasyfikacja wieloklasowa	Przypisanie próbki do jednej z klas	[0, 1] (suma = 1)
Linear (brak aktywacji)	Regresja	Przewidywanie wartości ciągłych	Dowolna wartość

Optymalizator aktualizuje wagi na podstawie funkcji straty.
Przegląd optymalizatorów:

Optymalizator	Zalety	Wady	Zastosowanie
Adam	Szybka konwergencja, uniwersalny	Czasami może przeuczać	Sieci CNN, RNN, ogólnie uniwersalny
SGD	Prosty i wydajny na dużych zbiorach danych	Powolna konwergencja bez `momentum`	Modele liniowe, sieci płytkie
RMSProp	Stabilna i szybka konwergencja	Wrażliwy na wybór parametrów	Sieci RNN, modele sekwencyjne
AdamW	Zapobiega przeuczeniu (L2)	Nieco większy narzut obliczeniowy	Głębokie modele, sieci transformers
Nadam	Szybsza konwergencja dzięki Nesterovowi	Większy narzut obliczeniowy niż Adam	Dynamiczne problemy
Adagrad	Dostosowuje współczynnik dla każdej cechy	Zbyt szybkie zmniejszanie learning rate	Dane rzadkie, NLP
Adadelta	Poprawiony Adagrad	Wolniejsza konwergencja w niektórych problemach	Duże sieci neuronowe
Ftrl	Skuteczny w danych rzadkich	Nie działa dobrze w gęstych danych	Systemy rekomendacyjne, dane rzadkie

Porównanie różnych funkcji straty:

Funkcja straty	Kiedy używać	Wzór	Właściwości
MSE (`mse`)	Problemy regresyjne	$\frac{1}{n}\sum(y - \hat{y})^2$	Karze większe błędy, bardziej wrażliwe na outliery
MAE (`mean_absolute_error`)	Problemy regresyjne, bardziej odporne na outliery	(\frac{1}{n} \sum	y - \hat{y}
Huber Loss	Problemy regresyjne z umiarkowanymi outlierami	Hybryda MSE i MAE	Ogranicza wpływ outlierów powyżej pewnego progu
Binary Crossentropy (`binary_crossentropy`)	Klasyfikacja binarna	---	Odpowiednia dla klasyfikacji binarnej
Categorical Crossentropy (`categorical_crossentropy`)	Klasyfikacja wieloklasowa	---	Odpowiednia dla klasyfikacji wieloklasowej

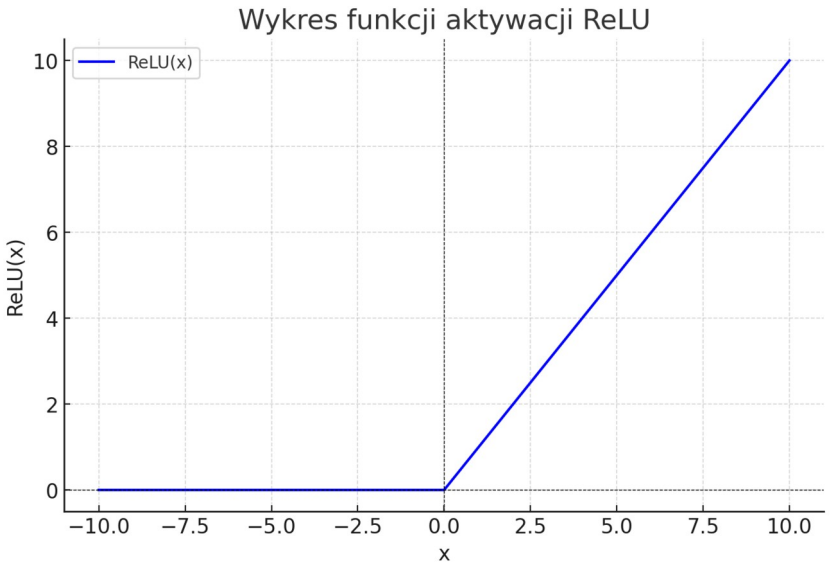
Interpretacja wyjścia uczenia się sieci:

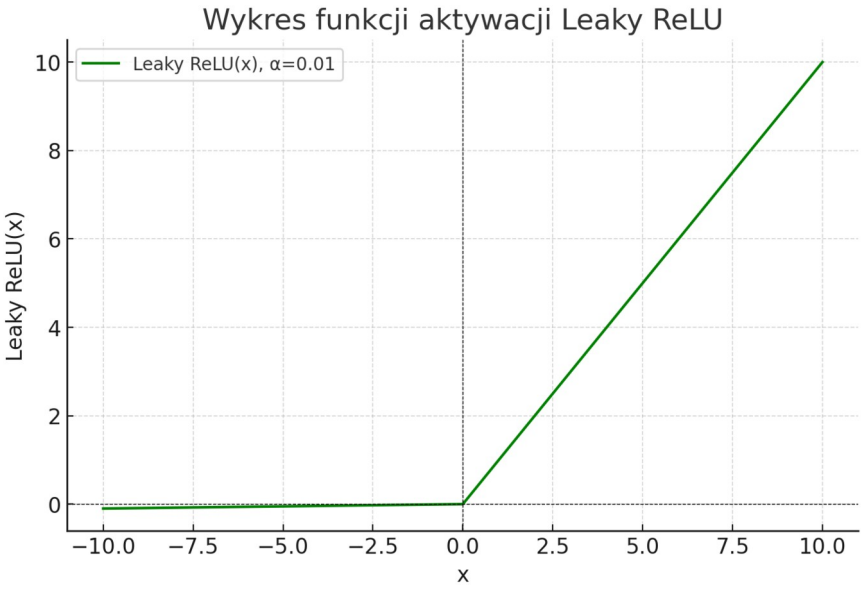
Metryka	Znaczenie	Zastosowanie
loss	Funkcja straty obliczona na danych treningowych (w Twoim przypadku MSE)	Pomaga zminimalizować różnice między przewidywaniami a rzeczywistością
mae	Średni błąd bezwzględny na danych treningowych	Interpretuje się jako średni błąd w jednostkach danych wyjściowych
val_loss	Funkcja straty obliczona na danych walidacyjnych	Ocena, jak dobrze model generalizuje wyniki na nieznanych danych
val_mae	Średni błąd bezwzględny na danych walidacyjnych	Pomaga ocenić praktyczny błąd w nowych danych

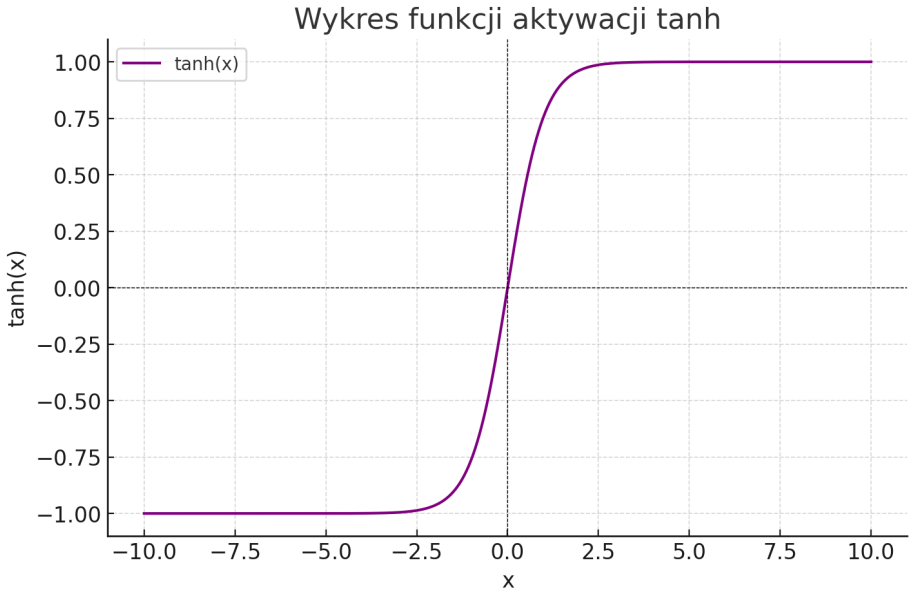
Alternatywy do Tensorflow:
- FANN (ale to do prostych sieci)
- PyTorch

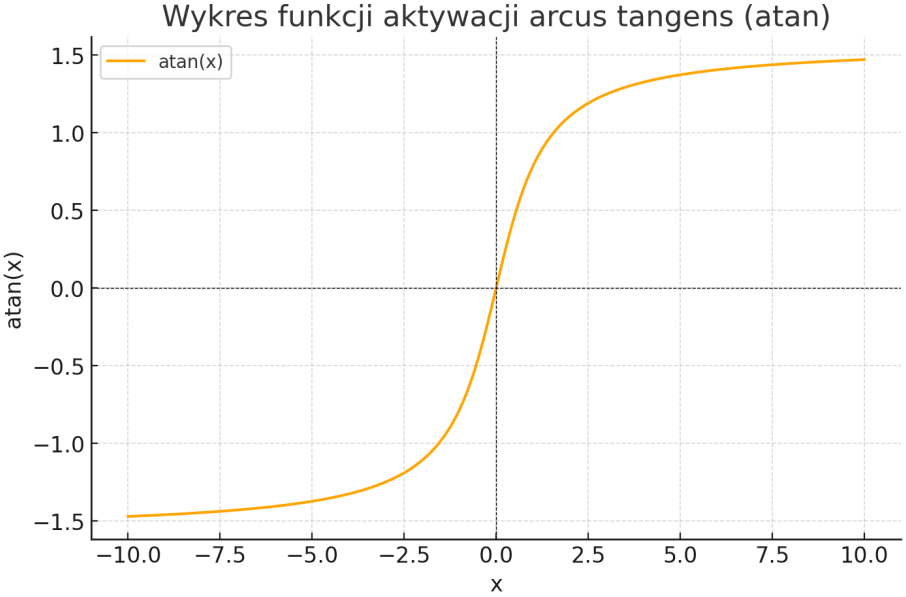
Tensorflow i PyTotch można uruchamiać na wielu kartach graficznych, na wielu komputerach w ramach klastra i jako rozwiązania chmurowe również.

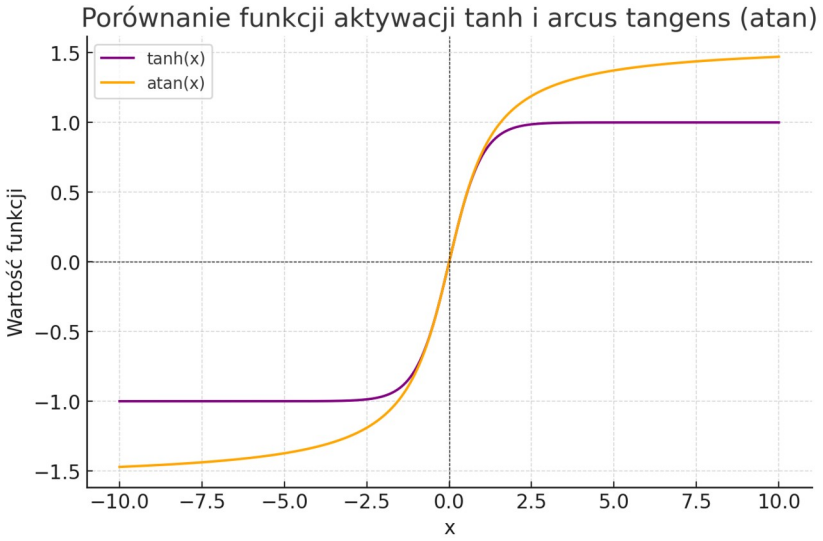
Tensorflow – rozwijany i udpstępniony przez Google
PyTorch – rozwijany i udostępniony przez Facebooka

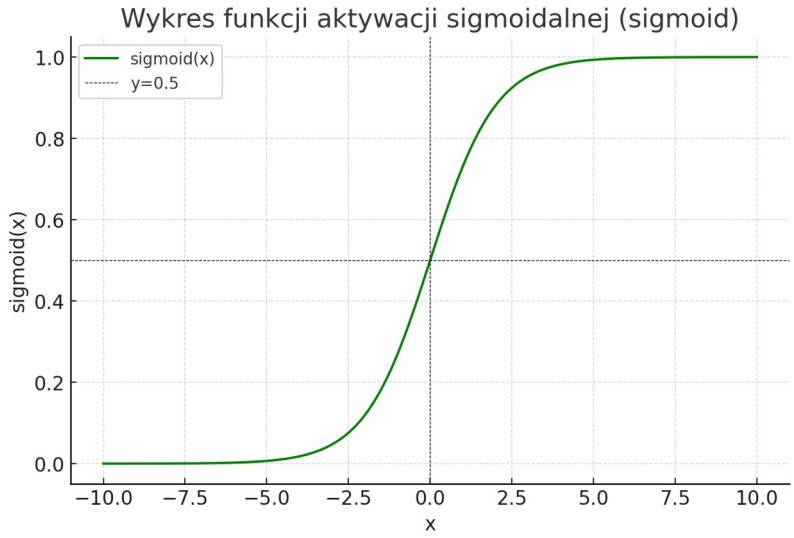



Wykres funkcji aktywacji Leaky ReLU
Leaky ReLU(x), α=0.01
Leaky ReLU(x)
x

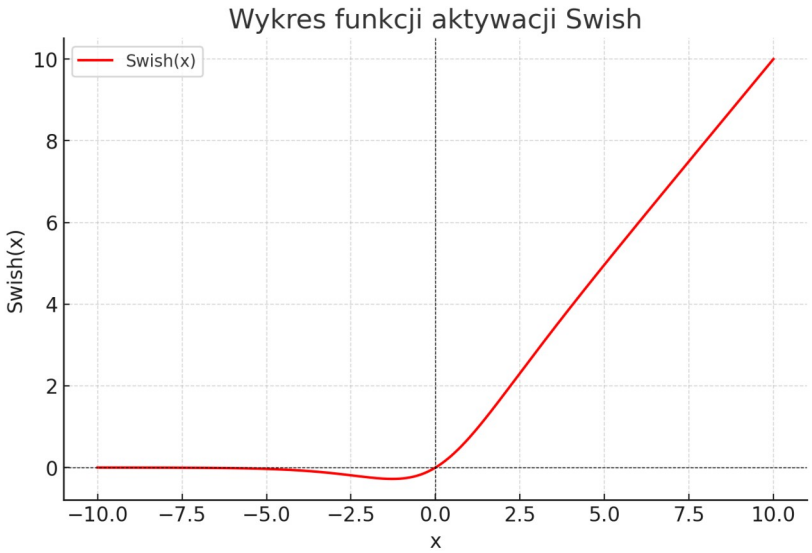

Wykres funkcji aktywacji tanh
tanh(x)
tanh(x)
x


Wykres funkcji aktywacji arcus tangens (atan)
atan(x)
atan(x)
x





Funkcja sigmoidalna, jest stosowana w klasyfikacja binarnej



Tensory szeregów czasowych dla warstwy ltsm podajemy w postaci:

[[f(cecha$_1$, t_1), f(cecha$_2$, t_1)…], [f(cecha$_1$, t_2), f(cecha$_2$, t_2)…] takiego, że $t_2 > t_1$ Chyba, że chcemy prognozować przeszłość a nie przyszłość :-)