**Podstawy Sztucznej Inteligencji – Laboratorium nr 3**

Wykonał: Kamil Wieniecki

Temat ćwiczenia: Budowa i działanie sieci wielowarstwowej typu feedforward.

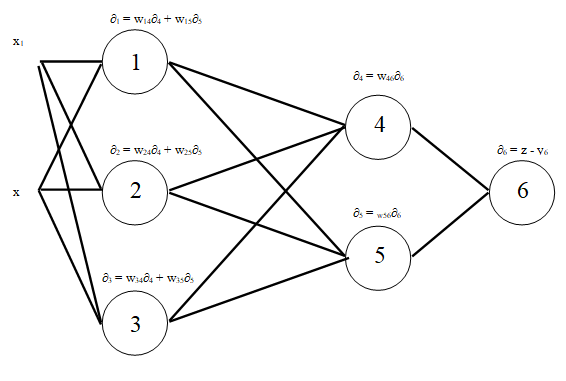
1. Cel ćwiczenia

Celem ćwiczenia jest poznanie budowy i działania wielowarstwowych sieci neuronowych poprzez uczenie kształtu wykresu funkcji matematycznej z użyciem wstecznej propagacji błędu.

1. Realizacja ćwiczenia

Wybrany przeze mnie język programowania do wykonania laboratorium to **Python.** Implementacja wykorzystuje bibliotekę open-source o nazwie **Keras**, która służy do deep neural learningu.

Biblioteka ta wykorzystuje algorytm wstecznej propagacji błedów (backpropagation), która przedstawia się następująco (przykład trójwarstwowej sieci neuronowej z dwoma wejściami i jednym wyjściem):



,gdzie:

δi – wartość błędu i-tego neuronu,

wab – waga neuronu a w neuronie b,

z – wartość oczekiwana,

yi – wartość wyjścia neuronu,

Testowane przeze mnie struktury sieci to:

- 30 - 1

- 30 – 10 - 1

- 30 – 30 – 30 - 1

Oraz kolejno współczynniki uczenia dla tych prób:

- 0.01

- 0.1

- 0.4

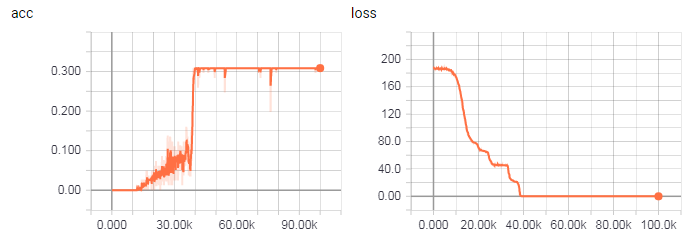
Każda próba uczenia opierała się na XXXX epokach, dla XXXX punktów wygenerowanych losowo dla tychże testów. Walidacja opierała się na siatce punktów ∂x = 0.5 oraz ∂y = 0.5. Biblioteka pozwala na określenie wielkości batch\_size, która to odpowiedzialna jest za aktualizacje wag po ilości próbek podanych jako ten właśnie argument. Implementacja wykorzystuje średni błąd kwadratowy, czyli różnicę pomiędzy estymatorem (wartością policzoną) i wartością estymowaną (wartością szacunkową).

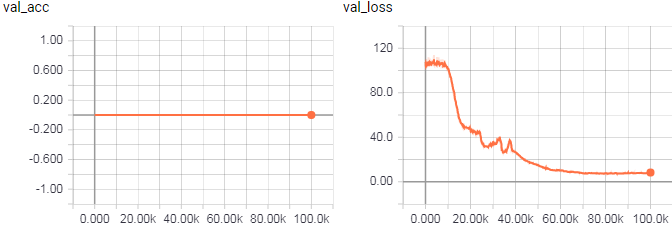
Każdy model został zapisany do pliku .h5, który może zostać w łatwy sposób odczytany przy pomocy metody bibliotecznej metody model\_load. Wyniki zostały zapisane w katalogu z logami, gdzie korzystając z narzędzia **TensorBoard** możemy stworzyć na ich podstawie wykresy.

1. Wyniki

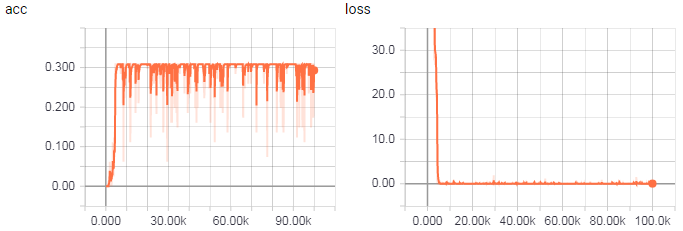
Wyniki z uczenia prezentują się następująco:

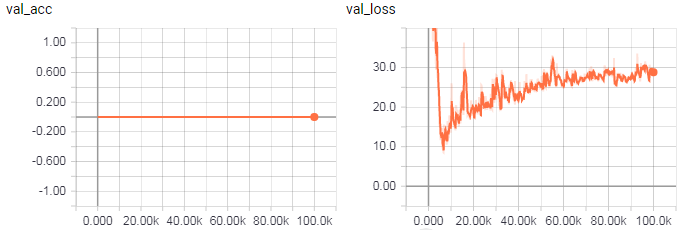
Dla struktury 30-1, learning rate = 0.01



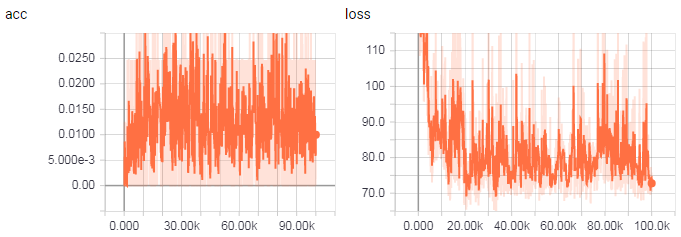


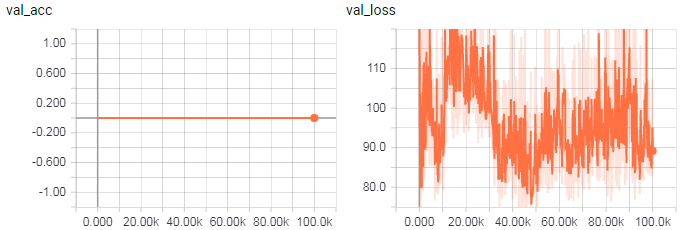
Dla struktury 30-1, learning rate = 0.1



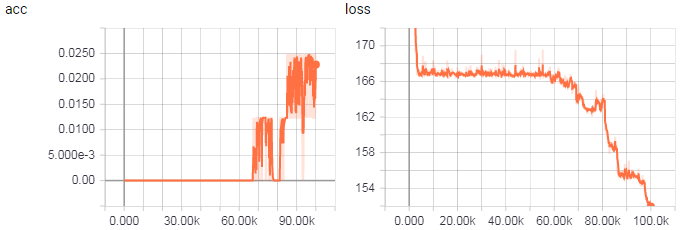
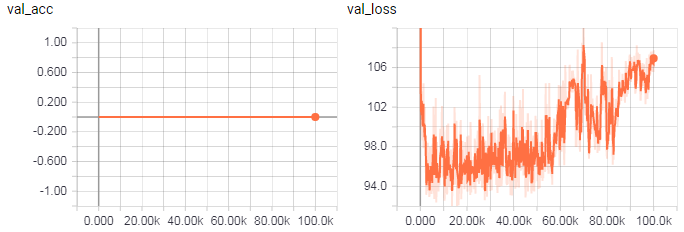


Dla struktury 30-1, learning rate = 0.4

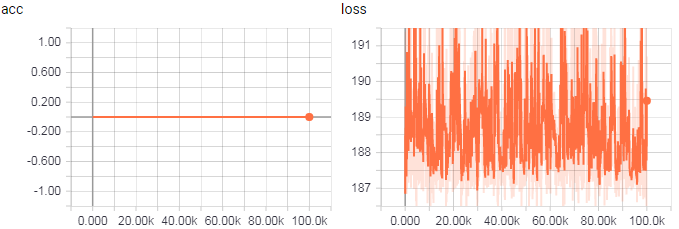


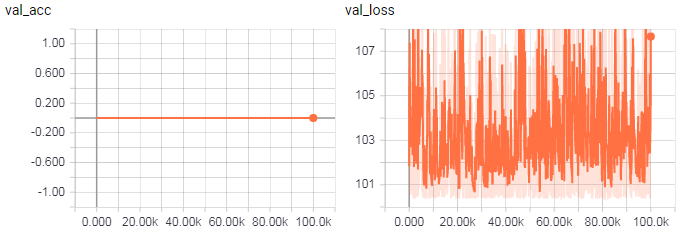


Dla struktury 30-10-1, learning rate = 0.01

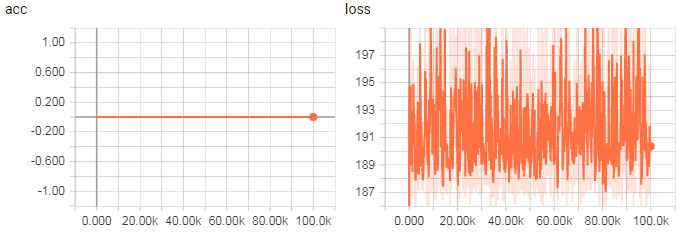
 

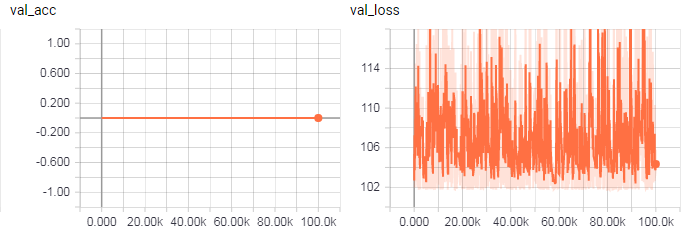
Dla struktury 30-10-1, learning rate = 0.1



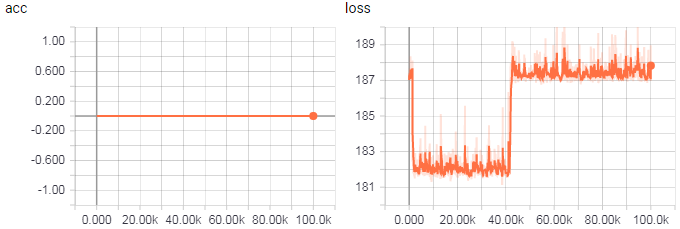


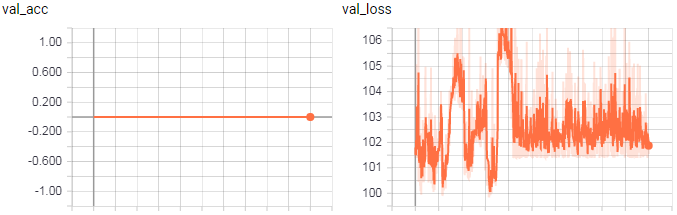
Dla struktury 30-10-1, learning rate = 0.4





Dla struktury 30-30-30-1, learning rate = 0.01





Dla struktury 30-30-30-1, learning rate = 0.1

Dla struktury 30-30-30-1, learning rate = 0.4

1. Analiza wyników
2. Wnioski

Przeciętny czas nauki jednej sieci to około 2,5h.

Niektóre struktury nie nadawały się całkowicie do nauki, gdyż implementacja biblioteki Keras nie radziła sobie z uczeniem na podstawie wprowadzonych danych. Najlepiej wypadła struktura perceptronów [ 30 – 1 ], w której to najlepiej widać. Chociaż współczynnik dokładności na poziomie około 30% nie jest najlepszym wynikiem. Funkcja Rastrigin jest również trudną do wyuczenia funkcją, ponieważ posiada bardzo mnóstwo minimów i maksimów lokalnych co znacznie utrudnia skuteczną naukę. Współczynnik szybkości nauki learning rate zdecydowanie wpływał na szybkość nauki sieci, co widać po zamieszczonych wykresach dla sieci.