(iii) opis struktury programu

Nasz program zasadniczo składa się z trzech części:

* Plik *datasetLoader.py* odpowiada za dostarczenie funkcji ładujących dane z plików .csv. Obecnie zawiera on możliwość załadowania zestawu danych (zarówno treningowych jak i testowych) dotyczących wytrzymałości materiałowych betonu. (Funkcja *readConcreteDataset()*). Dostarczone w ten sposób dane są już odpowiednio znormalizowane dla naszej sieci.
* Plik *test.py* odpowiada za zainicjalizowanie zmiennych przechowujących dane testowe i treningowe, inicjalizuje sieć neuronową, wywołuje funkcję uczącą, a następnie wypisuje wynik predykcji dla danych walidacyjnych oraz średni błąd w przewidywaniach.
* Plik *NetworkTwo.py* jest w zasadzie najważniejszą częścią projektu, gdyż to właśnie on zawiera całą implementację naszej sieci neuronowej:
  + *\_\_init\_\_(sizes)* – inicjuje sieć o warstwach podanych w formie listy

[ilość wejść, 1…n ,ilość wyjść] – gdzie 1…n – oznacza n wartości odpowiadających za ilości neuronów w kolejnych warstwach ukrytych (np. [5, 8, 8, 2] stworzy sieć o

5 wejściach, 2 wyjściach i o dwóch warstwach ukrytych po 8 neuronów każda).

* + *backprop(x, y)* – funkcja implementuje algorytm propagacji wstecznej. Zmienne *x* i *y* odpowiadają odpowiednio wejściom i oczekiwanym wyjściom sieci.
  + *SGD(trainingData, miniBatchSize, lRate, epochs, testData)* – funkcja implementuje algorytm stochastycznego spadku. Zmienne odpowiadają za:
    - *trainingData –* zawiera dane trenujące w formie (*wejścia, wyjścia*)
    - *miniBatchSize –* określa jak duże są partie danych dla których wyliczany jest gradient.
    - *lRate –* zmienna odpowiada jak duże „kroki” wykonuje algorytm podczas minimalizacji funkcji kosztu. Przez tą zmienną mnożony jest wynik algorytmu propagacji wstecznej podczas aktualizowania wag i biasów.
    - *Epochs –* ile iteracji dla danych testowych ma być wykonanych.
    - *testData –* zawiera dane testowe; na końcu każdej iteracji obliczany jest dla nich średni błąd i wypisywany w konsoli.
  + *Feedforward(x) –* funkcja implementująca wyliczanie wyjścia sieci dla wejścia x.
  + *ReLU(x):*
  + *ReLUprime(x):*