Marek Borzyszkowski 184266
Wojciech Baranowski 184574

Uczenie miar odległości

1. Opis problemu:

Celem projektu było:

a. Przetestowanie metod na zmierzenie odległości w przestrzeni n wymiarowej

b. Sprawdzenie potencjału generalizacji wybranych metod w wyższe wymiary.

c. Sprawdzenie potencjału generalizacji wybranych metod w tym samym wymiarze, bazując na limitowanej liczbie punktów.

Zastosowane algorytmy zostały zaimplementowane przy użyciu języka interpretowanego Python, wraz z pakietami:

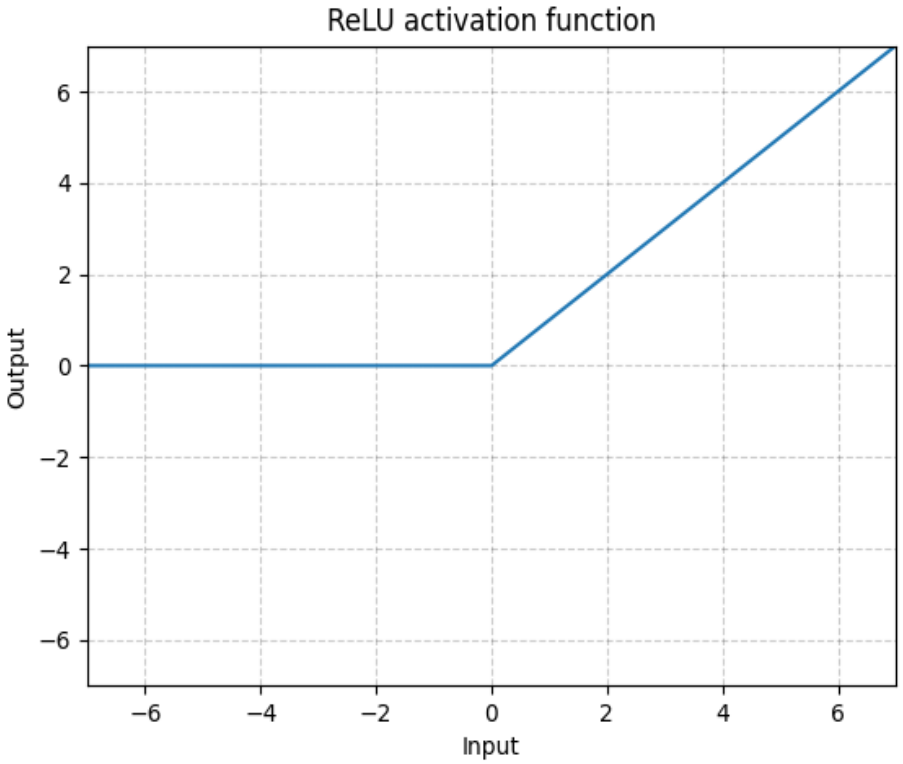
- Tensorflow
- Keras
- Sklearn
- Pandas
- Numpy

2. Metody użyte w problemie:

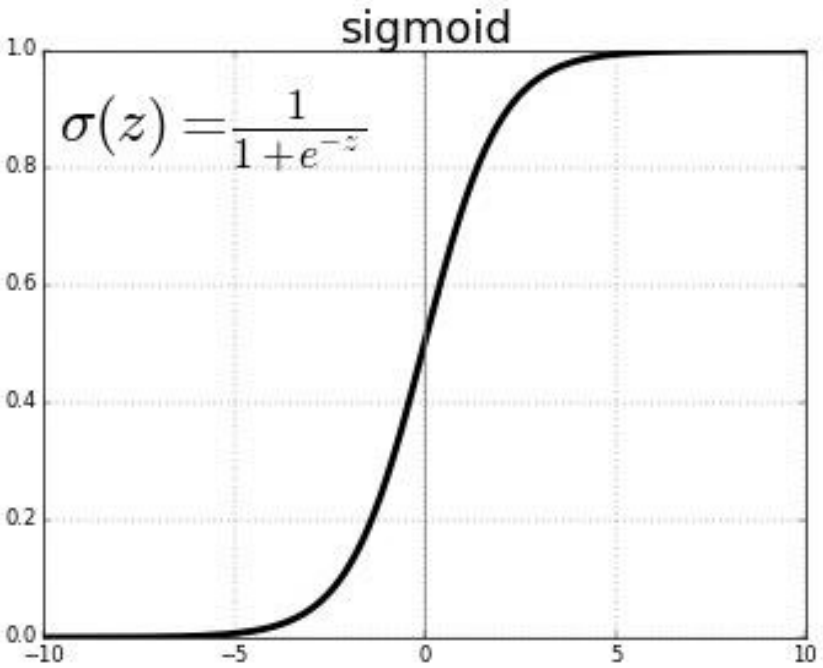
Przetestowano wykorzystanie w tym celu modelu sieci sekwencyjnej Fead Forward, jak i zgłębiliśmy temat sieci syjamskich.

Fead Forward – sieć neuronowa składająca się z warstw.
Warstwa wejścia, zawierająca określoną liczbę węzłów wejściowych na dane, które na podobieństwo grafu dwudzielnego pełnego, połączone są z następną z warstw.
Warstwa ukryta, składa się z n>=0 warstw węzłów ukrytych, które kaskadowo połączone są na wzór grafu dwudzielnego pełnego.
Warstwa wyjścia, połączona z ostatnią warstwą ukrytą, zawierająca 1 lub więcej węzłów jako wyjście.

Zarówno warstwa ukryta, jak i wyjścia posiada przypisaną funkcję aktywacji. Do najczęściej używanych zalicza się ReLU (Rectified Linear Unit), jak i funkcji sigmoidalnych.



https://pytorch.org/docs/stable/generated/torch.nn.ReLU.html



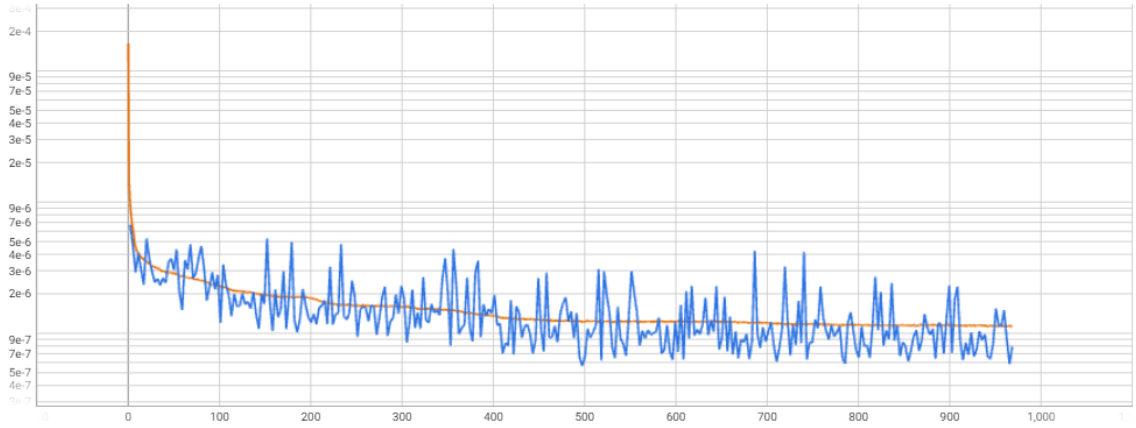
https://www.kaggle.com/general/197117

Każde połączenie między węzłami posiada swoją wagę, przez którą wykonuje się transformację na liczbie w węźle wychodzącym(zazwyczaj korzysta się z operacji mnożenia). Po transformacji następuje zsumowanie wszystkich tak powstałych wartości wchodzących do węzła. Do oceny modelu wykorzystuje się funkcję straty(w przypadku tego projektu skorzystano z błędu średniokwadratowego), która następnie uwzględnia się przy obliczaniach nowych wag sieci. W projekcie użyto optymalizacji stochastycznego spadku gradientu do obliczenia nowych wag.

Sieć syjamska – zawiera ona dwie, lub więcej identycznych sieci neuronowych, gdzie identyczne oznacza taką samą budowę, przypisane wagi i parametry. Stosuje się ją do znalezienia podobieństw między dwoma obiektami wejściowymi, poprzez zamianę ich na wektory cech, a następnie obliczenie odległości między wynikowymi wektorami każdej z sieci neuronowych. Sieci syjamskie są często wykorzystywane przy rozpoznawaniu twarzy, bądź też podpisów, bazując na zadanej bazie danych, ze względu na prostotę określenia wielkości różnicy między danymi.

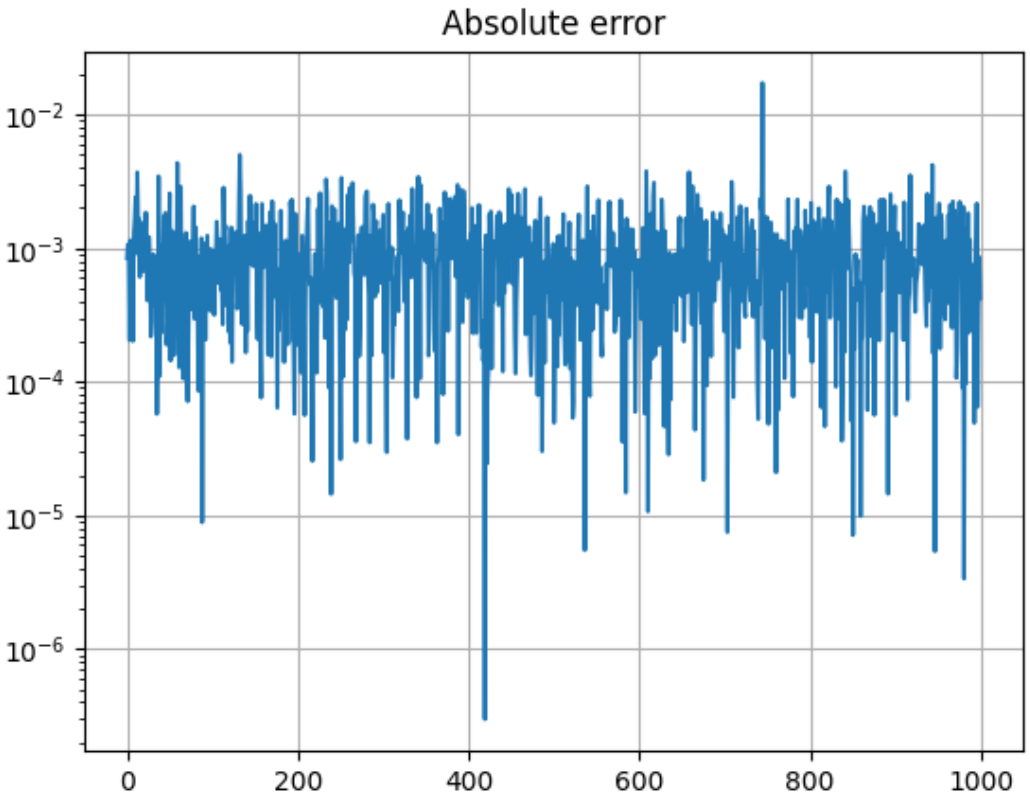
3. Opis realizacji i prezentacja wyników:

Pierwszy z pomysłów przetestowanych w projekcie było wygenerowanie wszystkich par kombinacji z powtórzeniami punktów w 3 wymiarze, na zadanym dyskretnym zbiorze $\{0, 0.1, 0.2 \dots 0.8, 0.9, 1\}$. Tak powstałe punkty były zbiorem treningowym, a łączna liczba par punktów wyniosła dokładnie 1.000.000. Następnie wygenerowano 1.000.000 par punktów w 3 wymiarze, ze składowymi w przedziale ciągłym <0, 1>, który wykorzystywano jako zbiór weryfikujący postępy.
Poniższy wykres przedstawia postęp uczenia się modelu sieci Fead Forward, która na wejściu przyjmuje 2 wektory w 3 wymiarze, a na wyjściu podaje odległość między nimi. Na pomarańczowo podano błąd średniokwadratowy w danej chwili dla predykcji na zbiorze treningowym, natomiast na niebiesko dla zbioru weryfikującego.

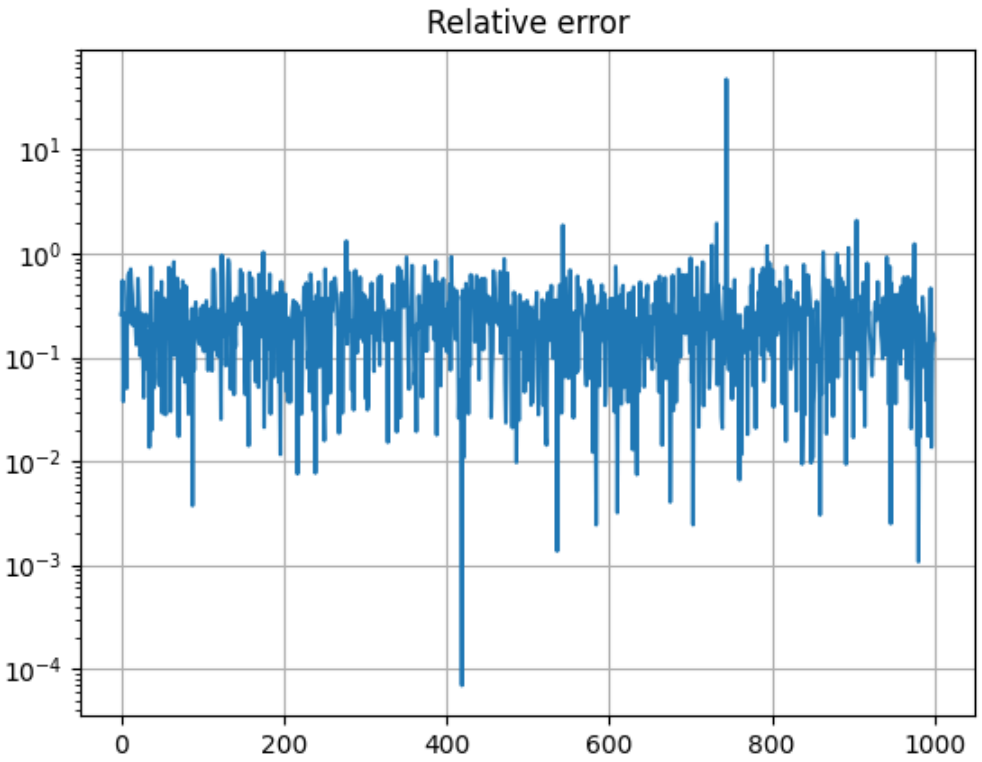


OX – numer epoki nauczania
OY – błąd średniokwadratowy

Każda z zaimplementowanych metod, po skończeniu procesu uczenia, została poddana testowi na 1000 losowych parach punktów w 3 wymiarze. Pary te nie pojawiły się wcześniej na jakimkolwiek z etapów treningu. Dla powyższego modelu wyniki były następujące:



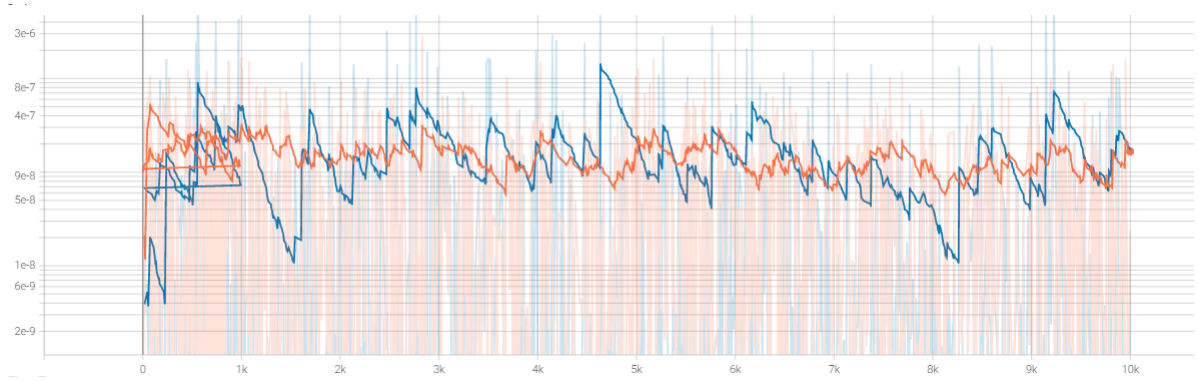
Średni błąd bezwzględny: 0.00093



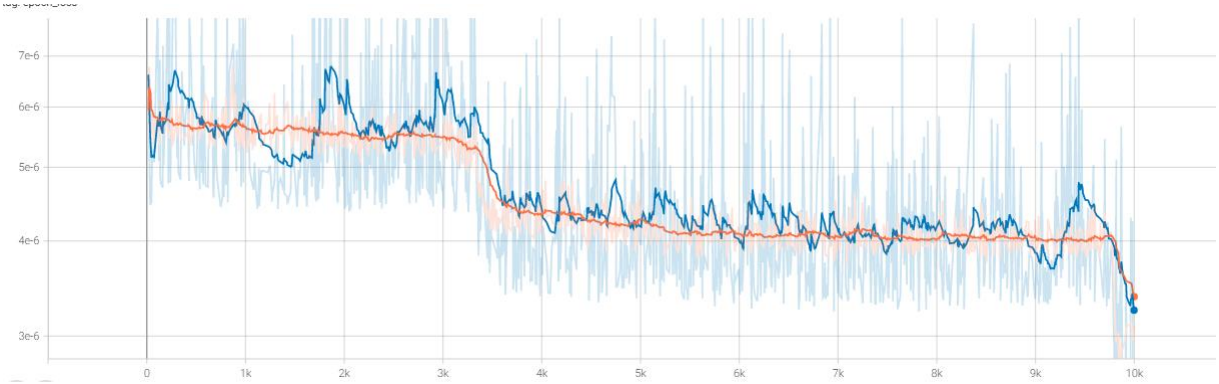
Średni błąd względny: 0.313%

Kolejnym pomysłem z próbą implementacji było wyliczenie normy Manhattan[1], euklidesowej[2], oraz nieskończonej[3] , jednakowoż błąd średniokwadratowy tego rozwiązania w fazie uczenia cały czas utrzymywał się w okolicach 0.166. Wynik ten wykluczył tą metodę jako jedną z potencjalnych sposobów podejścia do problemu, mimo że posiadała ona duży potencjał generalizacji w wyższe wymiary, poprzez brak wymagania przebudowywania całej sieci, w wypadku zwiększenia wektorów wejściowych, co mogłoby się przełożyć na brak konieczności douczenia jej w owych warunkach.

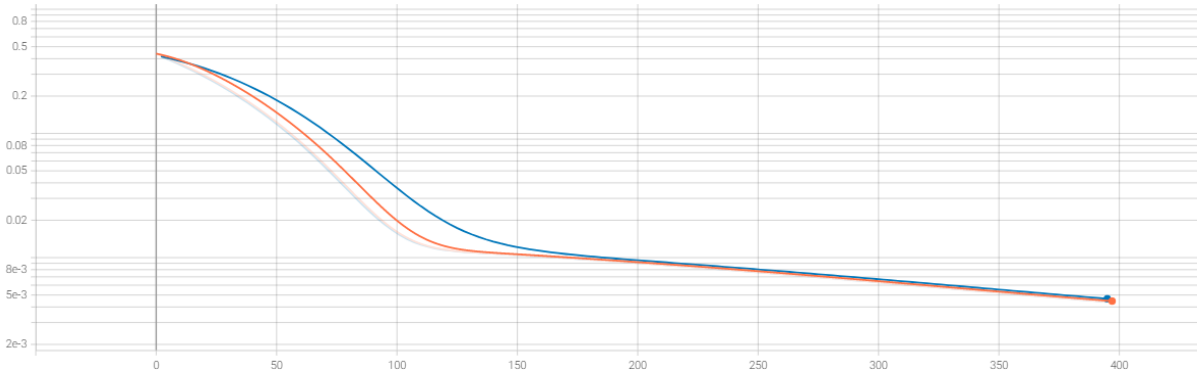
Ostatnim podejściem w pełni zaimplementowanym, było stworzenie sieci Fead Forward do odejmowania 2 liczb, mnożenia 2 liczb, oraz pierwiastkowania liczby. Zaczynając od sieci odejmującej, była ona uczona na zbiorze 10000 par punktów, powstałych na bazie dyskretnego zbioru {0, 0.01, 0.02 … 0.98, 0.99, 1}. Oto wykres przedstawiający proces uczenia się sieci, gdzie kolory, jak i osie wykresu pełnią analogiczną funkcję, co w przypadku poprzedniego wykresu uczenia. Dla ułatwienia zaobserwowaniu trendu, wykres został wygładzony.



W podobny sposób wytrenowano sieć mnożącą, korzystając z bliźniaczego zbioru danych.



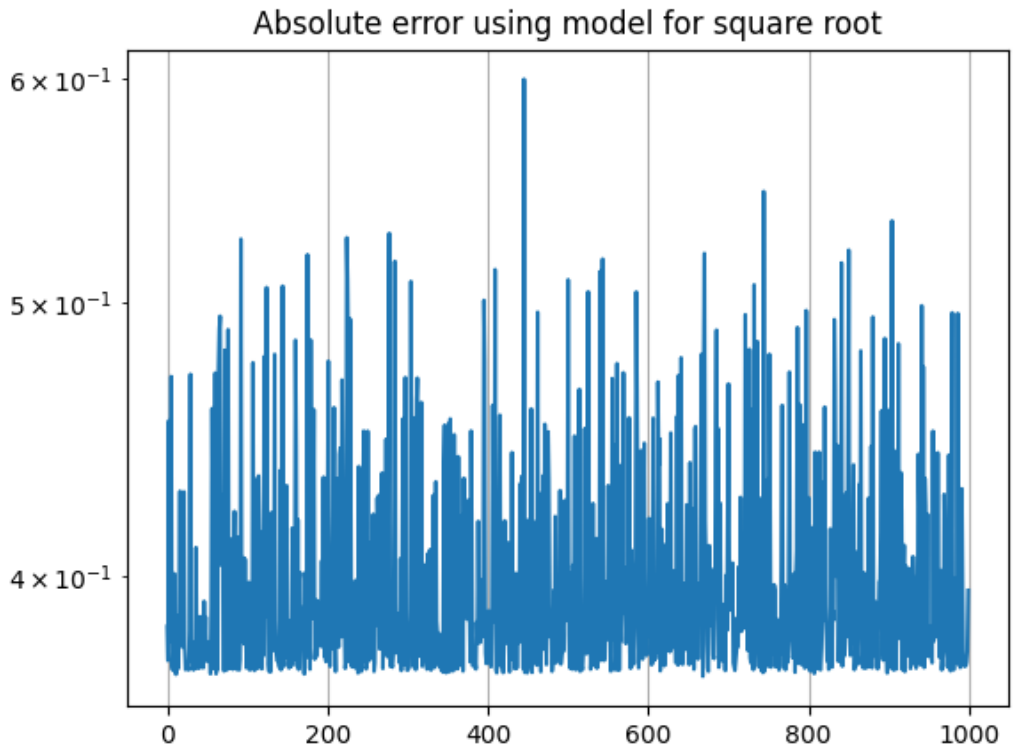
Komplikacje pojawiły się przy sieci, która miała za zadanie obliczyć pierwiastek drugiego stopnia z zadanej liczby. Problem dotyczył implementacji sieci, gdyż niezależnie od kombinacji zadanych parametrów, bądź też przetestowania prawidłowości danych wejściowych. W trakcie uczenia program dochodził do wartości poniżej zera, co automatycznie przerywało dalsze jego wykonywanie. Oto wykres procesu uczenia się sieci, na danych z takiego samego zbioru dyskretnego, jak poprzednie 2 sieci:



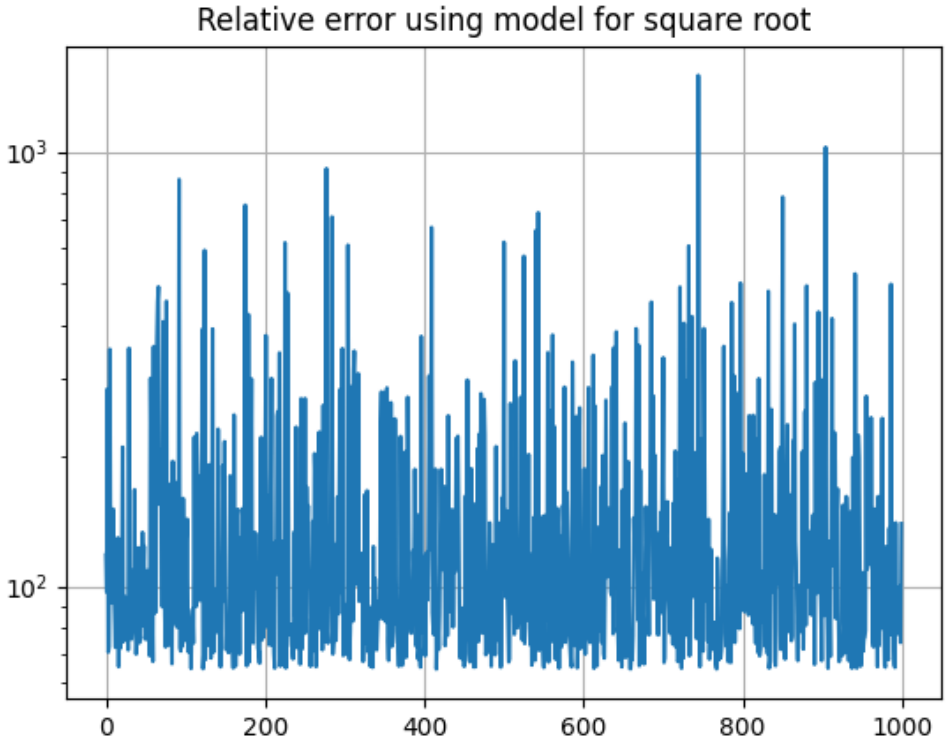
Powyższa komplikacja doprowadziła do powstaniu dwóch wariacji metody, wykorzystującą sieć do obliczania pierwiastka kwadratowego, oraz wykorzystującą wbudowane narzędzia w języku Python do obliczenia pierwiastka kwadratowego.

Po obliczeniu powyższych sieci skonstruowano program obliczający odległość między dwoma wektorami, gdzie do odejmowania, mnożenia(i pierwiastkowania), wykorzystano sieci neuronowe. Operatory sumy wykonano przy pomocy wbudowanych operacji matematycznych. Sumę przed przepuszczeniem przez sieć pierwiastkującą podzielono przez wymiar wektora wejściowego, a następnie wynik z sieci przemnożono przez pierwiastek z wymiaru. Oto jak plasują się wyniki:

Dla wariantu z siecią do obliczania pierwiastka kwadratowego:

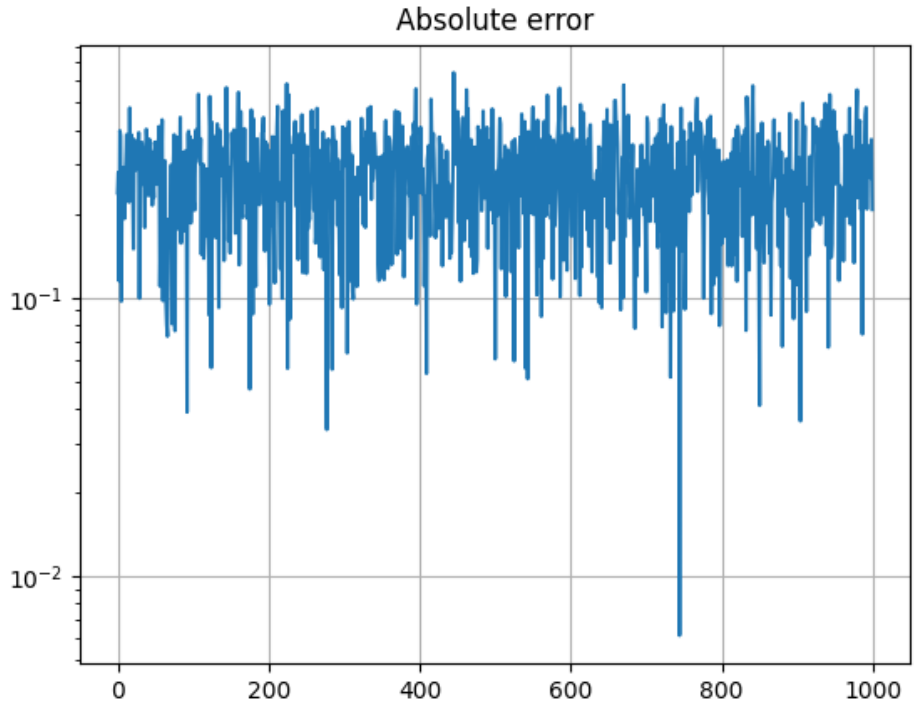


Średni błąd bezwzględny: 0.399

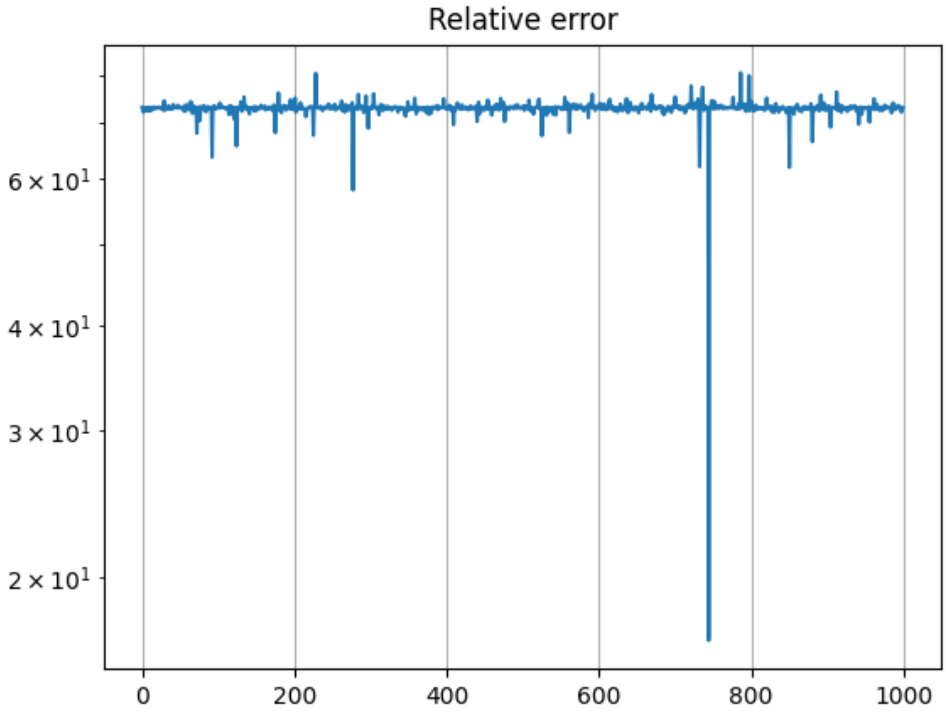


Średni błąd względny: 138.79%

Dla wariantu z wykorzystaniem wbudowanej operacji pierwiastka kwadratowego:



Średni błąd bezwzględny: 0.276



Średni błąd względny: 72.88%

Jednym z pomysłów z dużym potencjałem było zastosowanie sieci syjamskich do problemu. Z powodu komplikacji w implementacji w języku Python, nie doszła ona jednakowoż do skutku. Dobrym źródłem ukazujący użycie tej metody w bliźniaczym problemie można znaleźć w podanym artykule naukowym[4] . Natomiast na tej stronie[5] można znaleźć wytłumaczenie 3 sposobów na zastosowanie sieci syjamskich do danego problemu.

4. **Dyskusja:**

Po przetestowaniu wspomnianych metod, można dojść do następujących wniosków:

- **a.** Sieć Fead Forward bezpośrednio obliczająca dystans między dwoma zadanymi wektorami nadaje się dobrze do generalizacji w obrębie wyuczonego wymiaru i niższych. Ucząc się na danych dyskretnych, była w stanie dokonać predykcji odległości z niskimi wartościami błędu względnego, na punktach ze zbioru liczb <0, 1>. Posiada ona jednak wadę w postaci trudności(a na ten moment niemożliwości) generalizacji modelu w wyższe wymiary, gdyż wymagałoby to zmiany budowy sieci, a co za tym idzie, ponownego przejścia procesu uczenia.
- **b.** Pomysł wykorzystujący sieci do operacji podstawowych, taki jak odejmowanie, mnożenie, pierwiastkowanie, zawiera duży potencjał do generalizacji w dowolny wymiar, poprzez brak teoretycznego ograniczenia wymiaru wektorów wejściowych. Posiada on jednak dużą wadę, mianowicie narastające błędy numeryczne, które potrafią doprowadzić błąd względny do 70%, a nawet od 140%.

5. **Potencjalny dalszy rozwój:**

Najbardziej obiecującym pomysłem jest poprawienie wydajności metody stworzonej z podsieci, szczególnie naprawienie błędu uniemożliwiającego poprawne uczenie sieci pierwiastkowania. Kolejnym z etapów mogło by być wykorzystanie tych sieci do konstrukcji większej, gdzie wszelkie operacje odbywały by się przy wykorzystaniu podsieci, jak i przy pomocy transformacji wartości za pomocą wag.

Następną z opcji na dalszą analizę, jest przetestowanie wariacji sieci syjamskich na zadanym problemie, a następnie dokonania porównania wyników z już istniejącymi sieciami.

6. Bibliografia:

[1]https://en.wikipedia.org/wiki/Norm_(mathematics)#Taxicab_norm_or_Manhattan_norm

[2]https://en.wikipedia.org/wiki/Norm_(mathematics)#Euclidean_norm

[3]https://en.wikipedia.org/wiki/Norm_(mathematics)#Maximum_norm_(special_case_of:_infinity_norm,_uniform_norm,_or_supremum_norm)

[4]https://arxiv.org/pdf/2001.09189.pdf

[5]https://towardsdatascience.com/neural-networks-intuitions-9-distance-metric-learning-dae7c3a0ebdf