

# LAB5

## Wstęp

Implementacja algorytmu znajduje się w pliku MLP, generowanie wykresów w pliku graphs.

Użyte biblioteki/funkcje:

- sklearn
- numpy
- ProcessPoolExecutor z concurrent.futures
- Pyplot

W implementacji grafów została użyta ProcessPoolExecutror w celu znaczącego przyspieszenia wykonywania się wykresów

Nazwa	Stan	100% Procesor...	33% Pamięć	0% Dysk	0% Sieć	Zużycie energii	Trend zużycia ...
Python		7,1%	104,6 MB	0 MB/s	0 Mb/s	Bardzo wysokie	Bardzo niskie
Python		6,6%	102,5 MB	0 MB/s	0 Mb/s	Bardzo wysokie	Bardzo niskie
Python		6,3%	104,5 MB	0 MB/s	0 Mb/s	Bardzo wysokie	Bardzo niskie
Python		6,3%	104,0 MB	0 MB/s	0 Mb/s	Bardzo wysokie	Bardzo niskie
Python		6,3%	102,4 MB	0 MB/s	0 Mb/s	Bardzo wysokie	Bardzo niskie
Python		6,3%	103,5 MB	0 MB/s	0 Mb/s	Bardzo wysokie	Bardzo niskie
Python		6,2%	102,4 MB	0 MB/s	0 Mb/s	Bardzo wysokie	Bardzo niskie
Python		6,1%	102,1 MB	0 MB/s	0 Mb/s	Bardzo wysokie	Bardzo niskie
Python		5,9%	102,7 MB	0 MB/s	0 Mb/s	Bardzo wysokie	Bardzo niskie
Python		5,8%	102,6 MB	0 MB/s	0 Mb/s	Bardzo wysokie	Bardzo niskie
Python		5,7%	102,6 MB	0 MB/s	0 Mb/s	Bardzo wysokie	Bardzo niskie
Python		5,7%	102,4 MB	0 MB/s	0 Mb/s	Bardzo wysokie	Bardzo niskie
Python		5,7%	102,5 MB	0 MB/s	0 Mb/s	Bardzo wysokie	Bardzo niskie
Python		5,7%	102,3 MB	0 MB/s	0 Mb/s	Bardzo wysokie	Bardzo niskie
Python		5,7%	102,1 MB	0 MB/s	0 Mb/s	Bardzo wysokie	Bardzo niskie
Python		5,6%	102,8 MB	0 MB/s	0 Mb/s	Wysoki	Bardzo niskie
> Menedżer zadań		0,5%	37,9 MB	0 MB/s	0 Mb/s	Bardzo niskie	Bardzo niskie

Część przedstawionych wykresów została wykonana na stacjonarnym komputerze z procesorem z 16 dostępnymi wątkami i wykonywanie ich z podanymi parametrami może skutkować bardzo długim czasem wykonania na jednostce obsługującej mniej wątków lub posiadającej znacząco słabsze rdzenie.

## Opis algorytmu

Zadanie polegało na zaimplementowaniu perceptronu wielowarstwowego oraz wybranego algorytmu optymalizacji gradientowej z algorytmem propagacji wstecznej.

Perceptron wielowarstwowy został wytrenowany do klasyfikacji zbioru danych MNIST (<http://yann.lecun.com/exdb/mnist/>). Zbiór MNIST został załadowany poprzez (`sklearn.datasets.load_digits()`).

## Przygotowania

Domyślne parametry dla dalszych eksperymentów, zostały określone metodą prób i błędów i wynoszą:

epochs: 100

learning rate: 0,01

batch\_size: 8

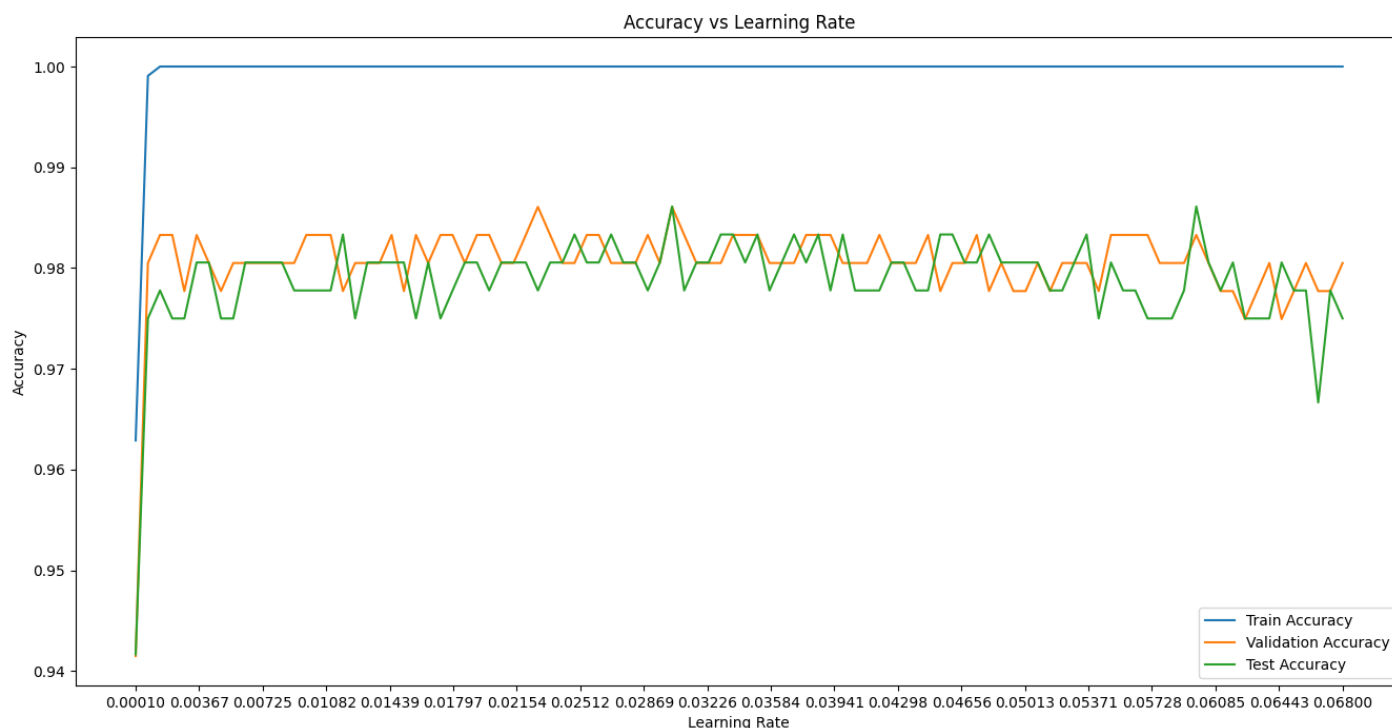
layers: 64, 128, 64, 10

Warstwy w dalszej części będą odnosić się do wartości między 64 i 10, takie liczby neuronów na początkowej i ostatniej warstwie są wymagane do poprawnego działania programu.

## Wpływ learning rate na accuracy:

### Pierwszy test:

Parametry: learning rate z zakresu 0.001 – 0.068 – 100 punktów na wykresie



### Obserwacje:

Bardzo małe wartości learning rate osiągają mniejsze wartości accuracy

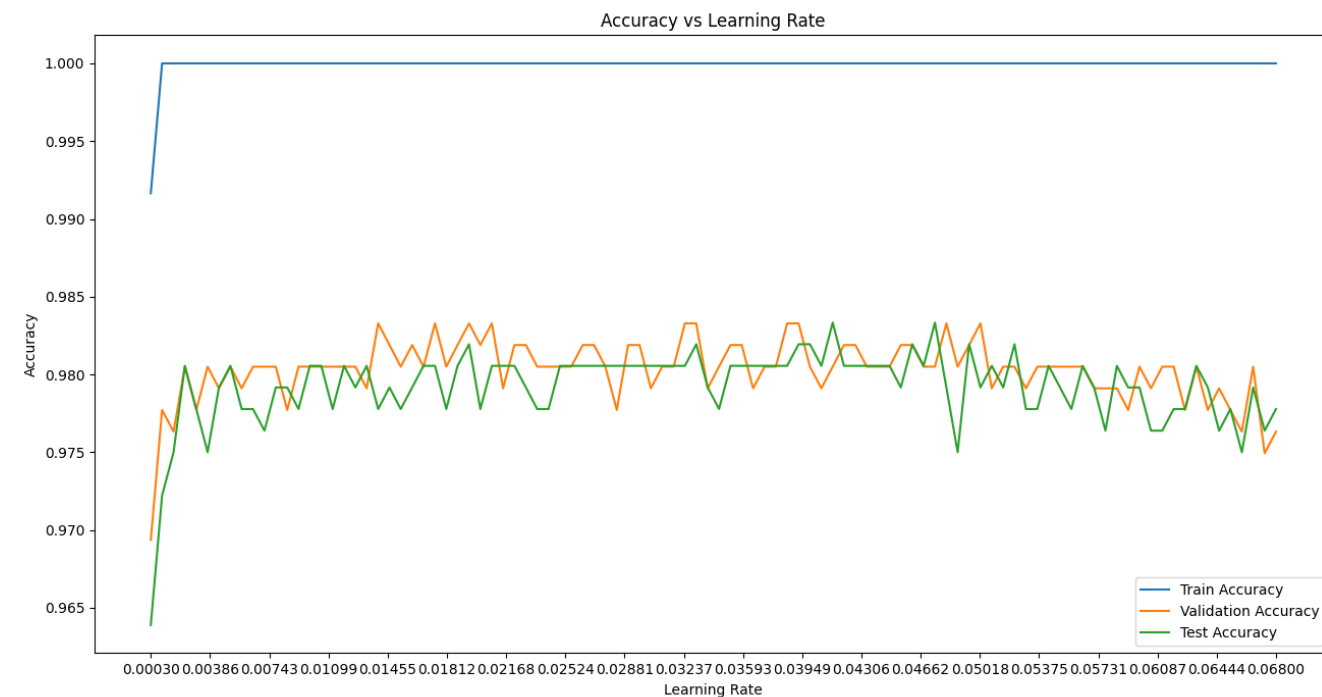
## Drugi test:

Te same parametry wykres jest medianą 5 wykonań



## Trzeci test:

Początkowa wartość learning rate została ustawiona na 0,003, aby zapewnić dokładniejszy wykres

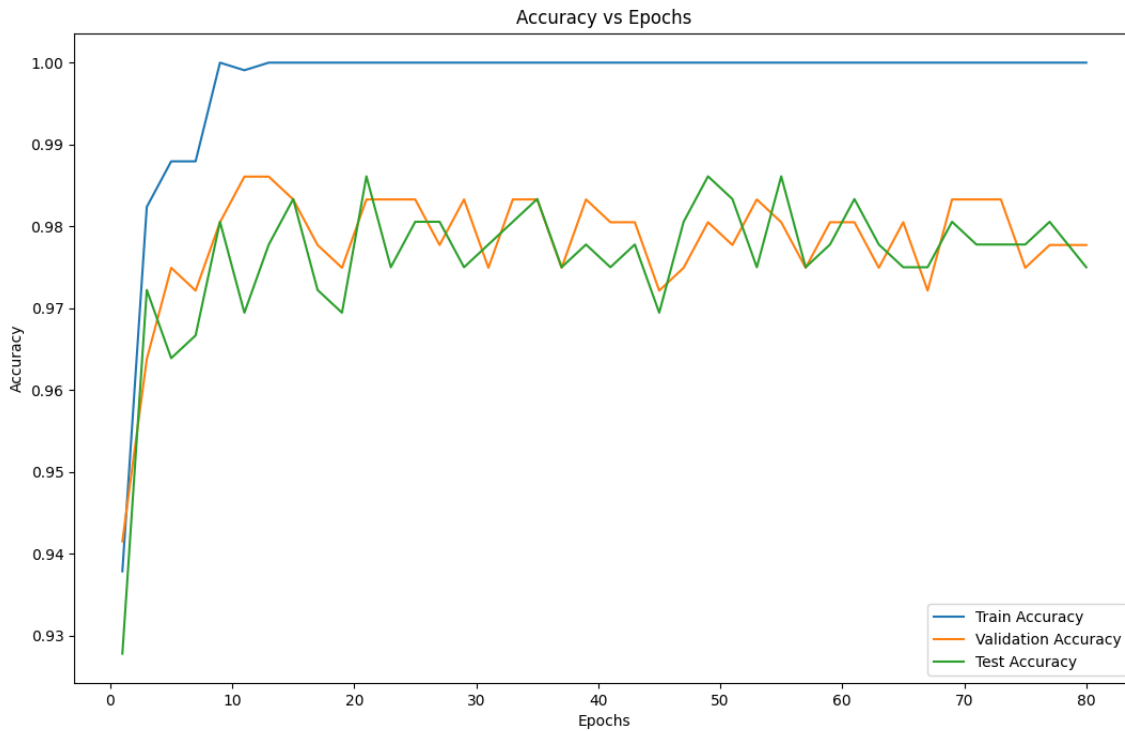


## Obserwacje:

Zauważalny mały trend spadkowy zaczynający się około 0.05

## Wpływ epochs na accuracy:

Ustawiona została zakres epochs między 1 i 80, badane jest 40 punktów, zakres dobrany dla optymalnej czytelności wykresu, wartości poniżej są medianą 10 wywołań

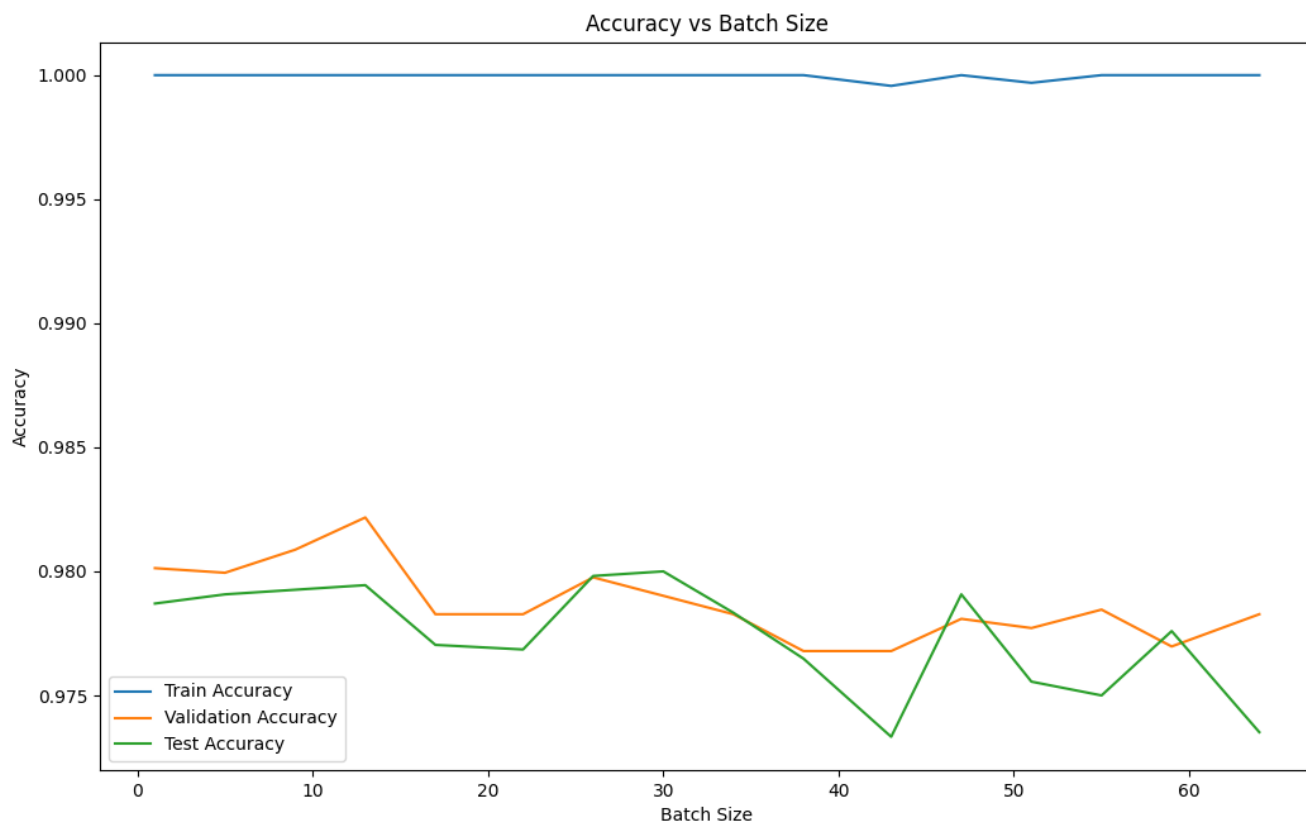


Obserwacje:

Małe wartości osiągają mniejsze wartości accuracy

## Wpływ batch size na accuracy:

Ustawiona został zakres epochs między 1 i 64, badane jest 16 punktów, wartości poniżej są średnią 15 wywołań



Obserwacje:

Wykonywanie treningu dla bardzo małych wartości zwiększa czas wykonania, natomiast generuje minimalnie lepsze wyniki

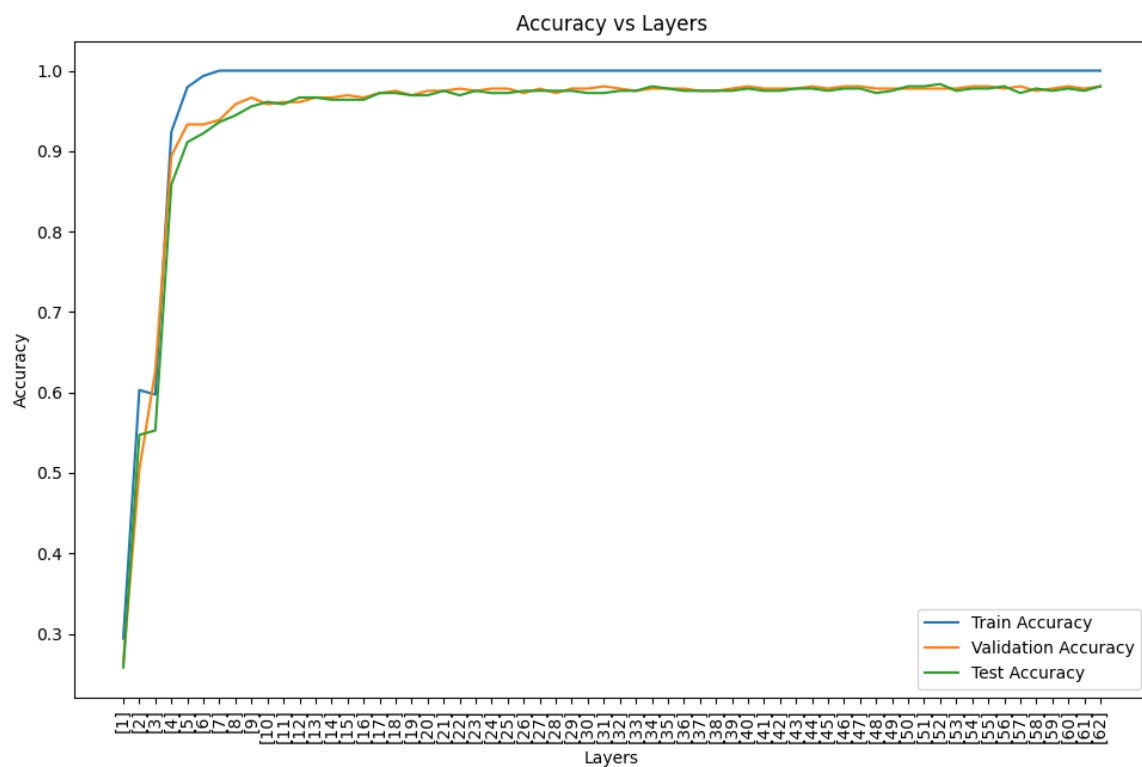
## Wpływ ilości neuronów i ich ustawienie na accuracy :

Wszystkie testy dla warstw są medianą 5 wykonań

Pojedyncza warstwa:

**Pierwszy test:**

Warstwa o wielkości od 1 do 62

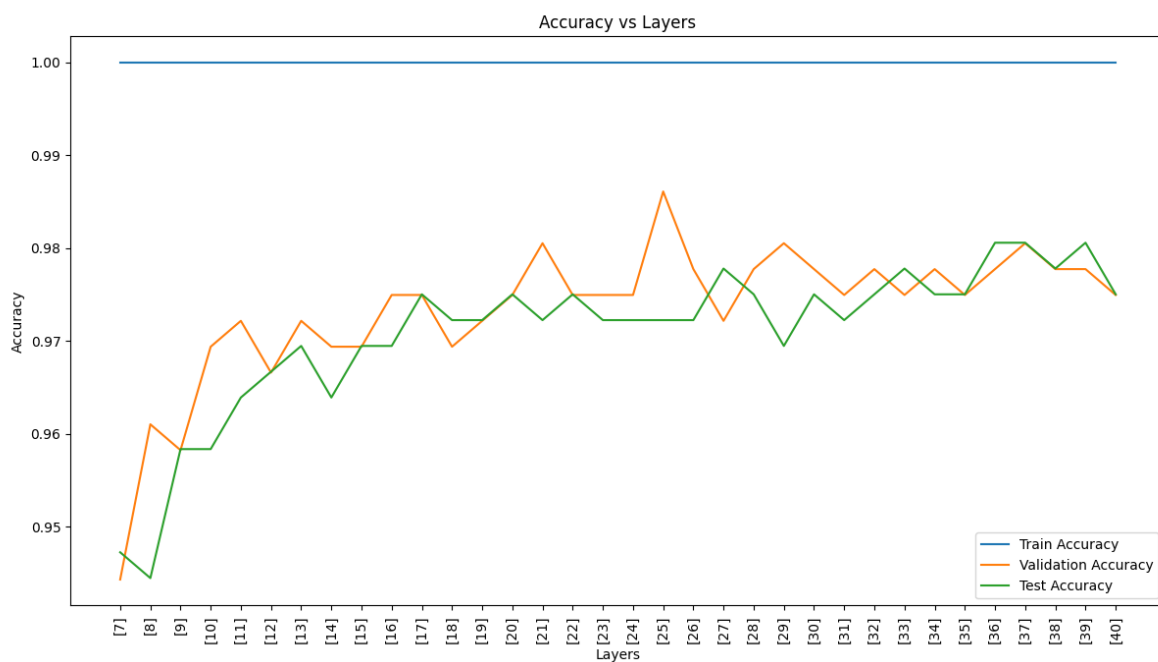


Obserwacje:

Bardzo mała liczba neuronów, uzyskuje bardzo małe wartości accuracy

### Drugi test:

Wartości między 7 i 40

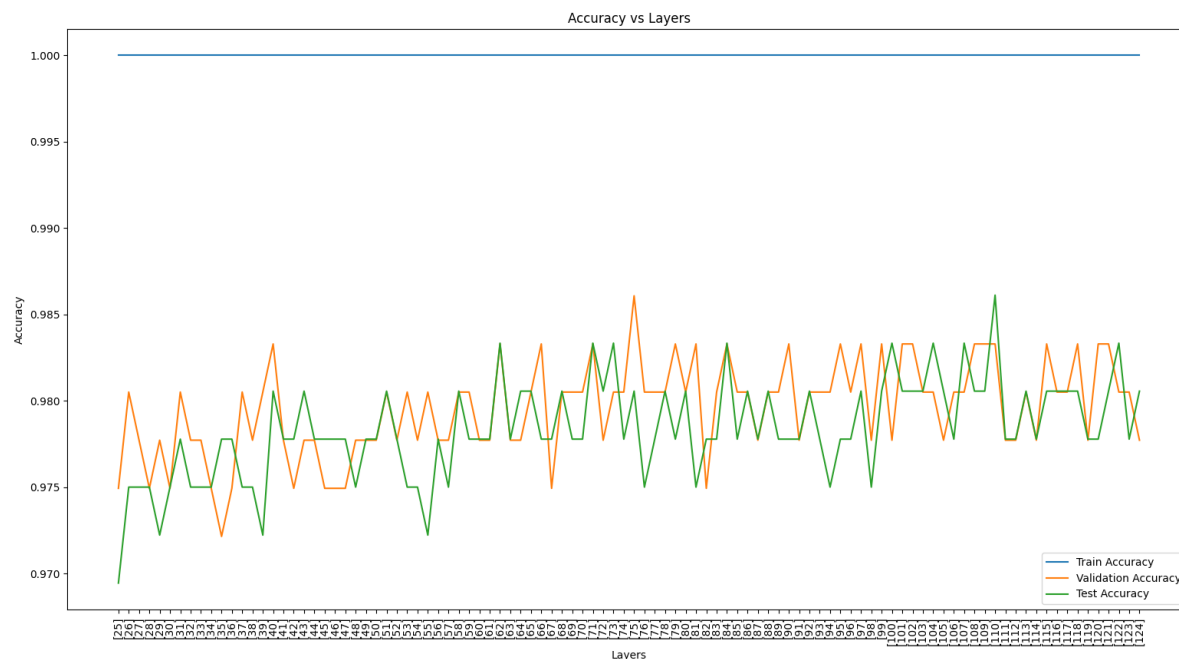


Obserwacje:

Wzrost accuracy wraz z liczbą neuronów

### Trzeci test:

Warstwa o wielkości od 25 do 124



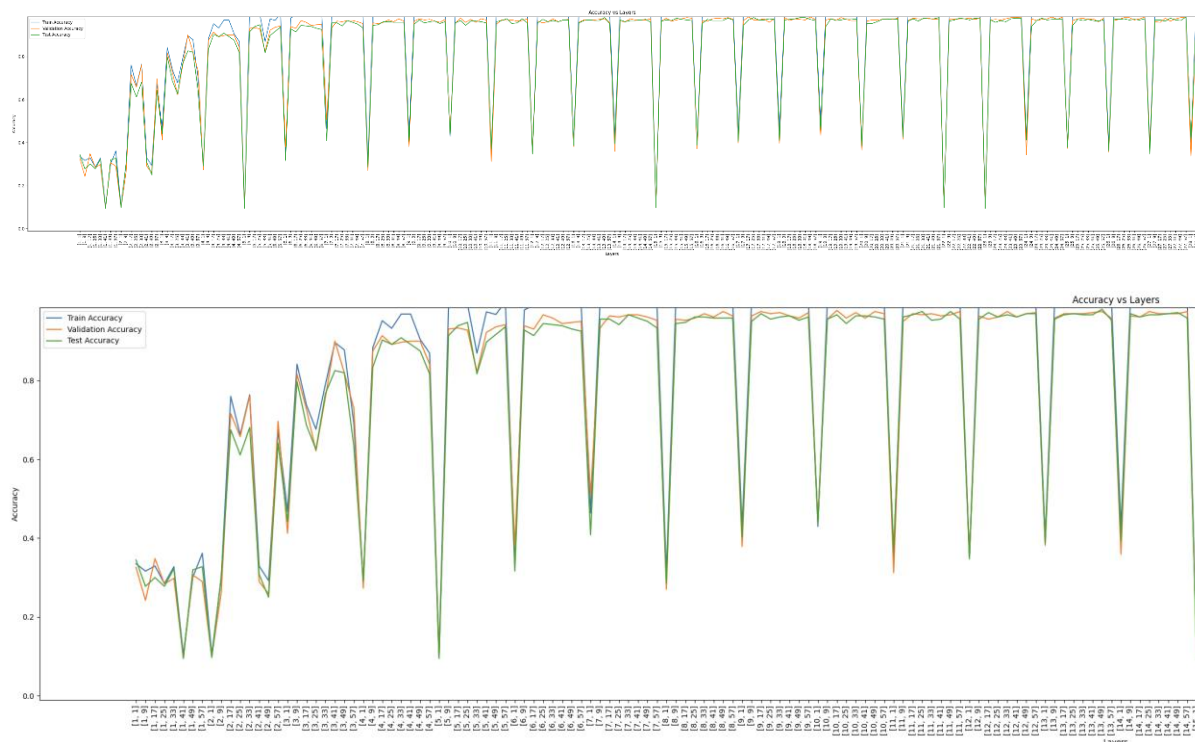
Obserwacje:

Mały wzrost dokładności wraz ze zwiększającą się liczbą neuronów

### Podwójna warstwa:

Pierwszy test:

1/8 kombinacji 2 warstw o zakresach od 1 do 64



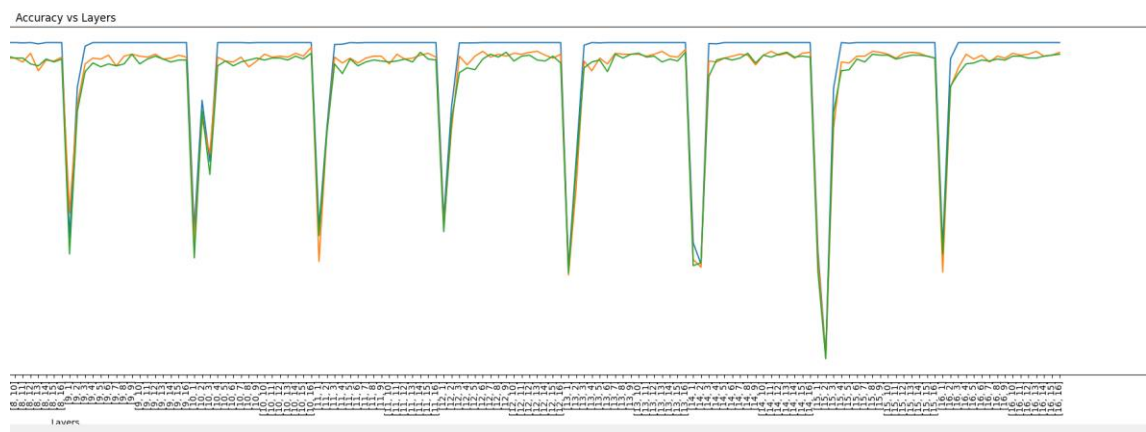
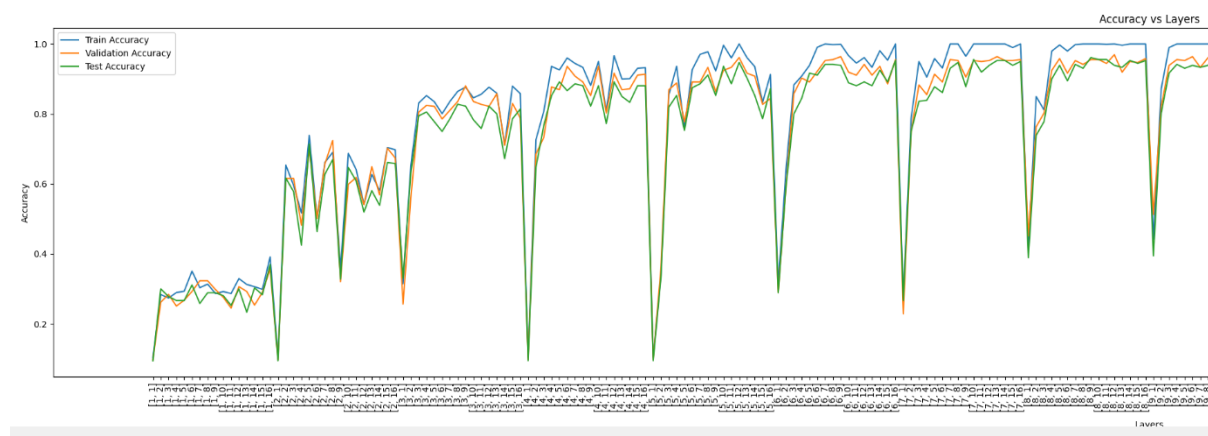
Mateusz Daszewski i Miłosz Andryszczuk

Obserwacje:

Pojedynczy neuron na warstwie, obniża znacząco accuracy. Mała liczba neuronów obniża accuracy

**Drugi test:**

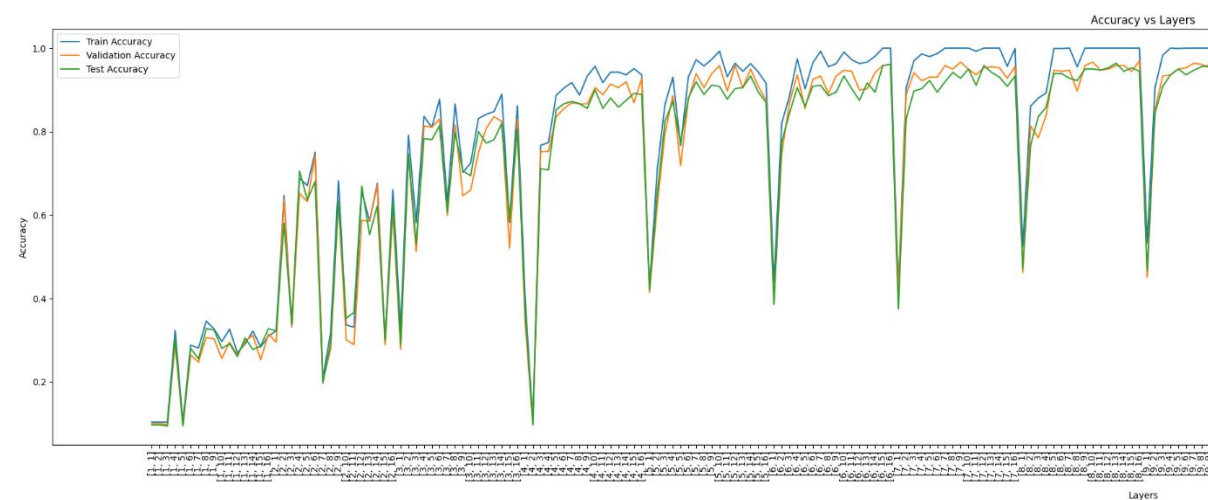
Wykres wszystkich kombinacji 2 warstw o wielkości z zakresu 1-16



Obserwacje:

Problem czytelności wykresu stanowią wartości z 1 neuronem

Przesortowany wykres względem  $a*b$ , a i b liczba neuronów na warstwach



Obserwacje:



Mateusz Daszewski i Miłosz Andryszczuk

Małe wartości oraz warstwy z jednym neuronem utrudniają czytelność

### Trzeci test:

Przesortowane wartości z poprzedniego testu, oraz ustawiono minimalną liczbę neuronów na warstwie na 4



### Obserwacje:

Mała liczba neuronów na dowolnej warstwie obniża accuracy. Wraz z wzrostem liczby neuronów na obydwu warstwach wzrasta accuracy.

### Czwarty test:

Wpływ batch size na poprzedni test, ustawiono wartość 2

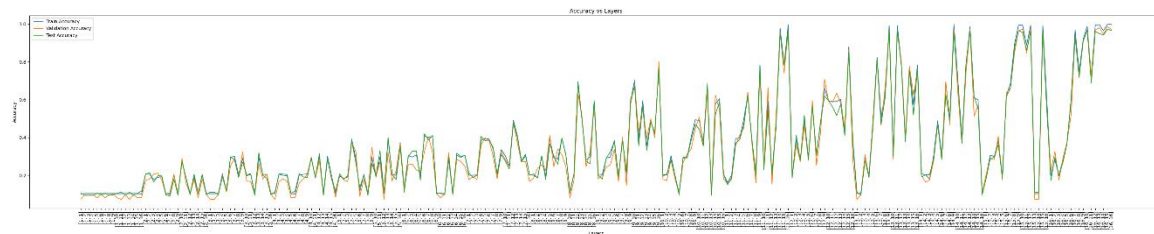


### Obserwacje:

Wykres jest bardziej „spłaszczony”. Czas obliczeniowy znacząco wzrósł

### Czwarty test:

Wpływ batch size=1, na ten sam zakres nieposortowany, wartości z raz wykonanego testu



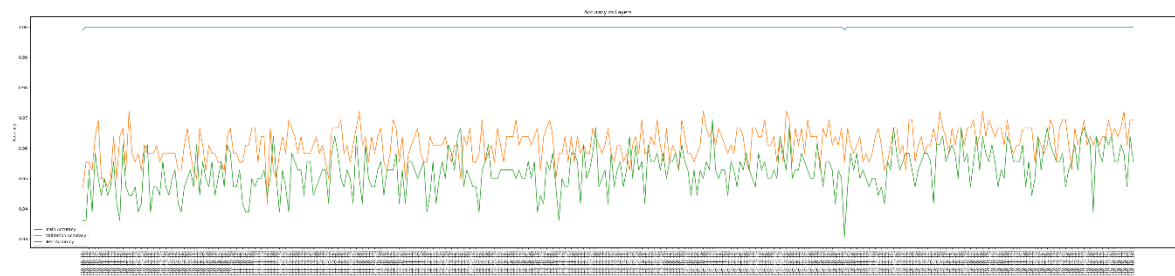
Obserwacje:

Wykres stał się bardziej „chaotyczny”, nie był generowany z wielu wykonania

## Trzywarstwowe sieci:

### Pierwszy test:

Wszystkie konfiguracje 3 warstwowych sieci z zakresem neuronów między 10-16

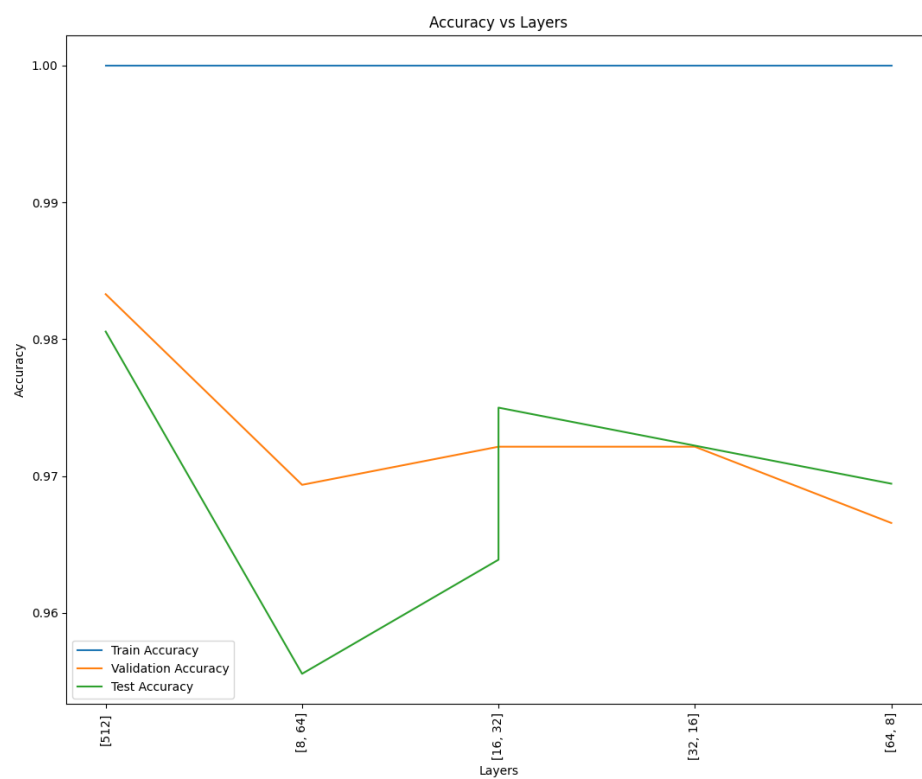
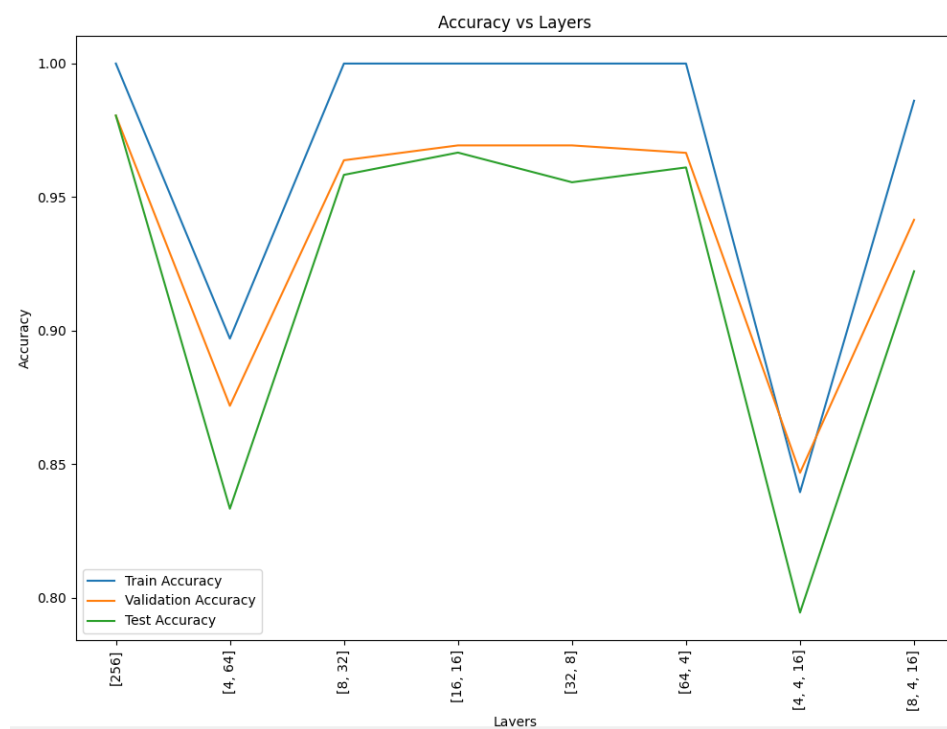


Obserwacje:

Mało czytelny wykres. Wszystkie wartości osiągają podobne wyniki

### Drugi test:

Porównanie wybranych wartości layers, które mają równą wartość mnożenia liczby neuronów przez warstwy

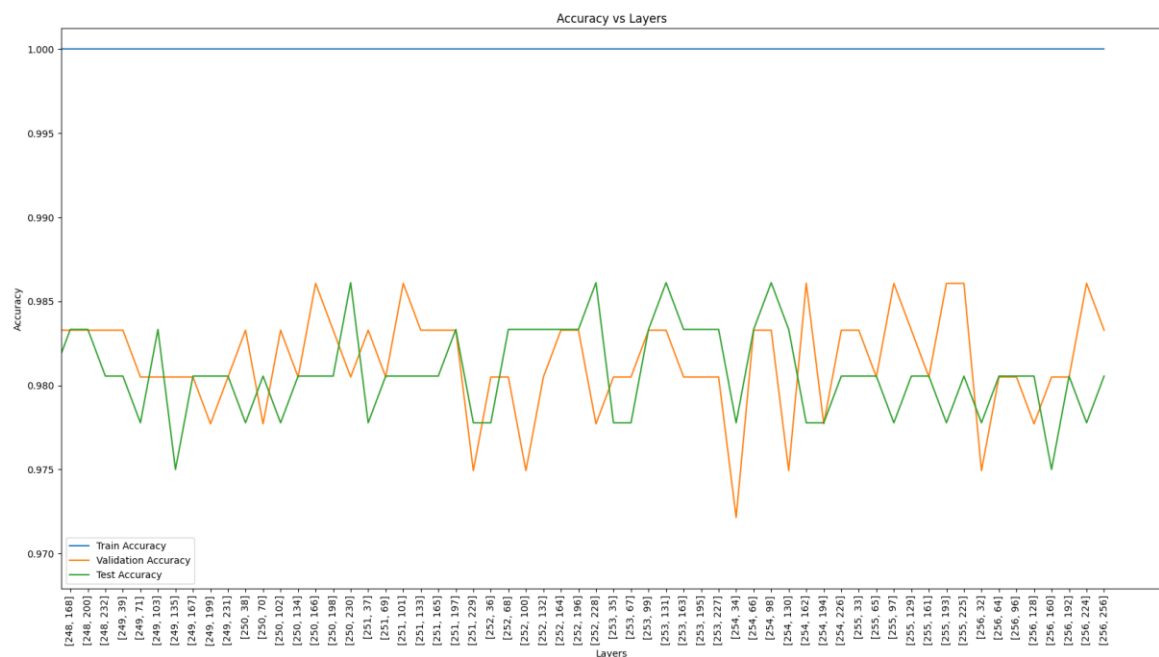
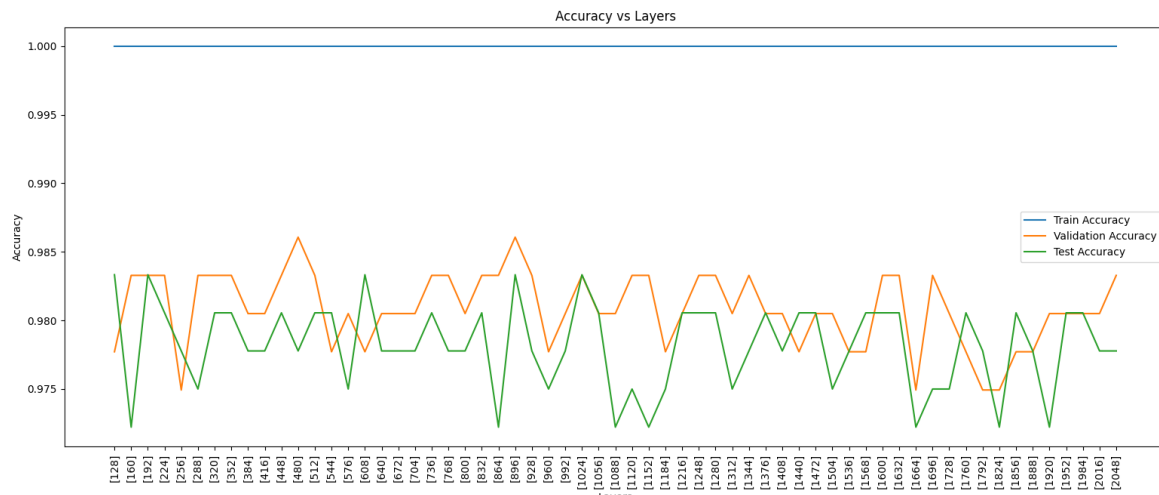


Obserwacje:

Warstwy z małą liczbą neuronów, obniżają accuracy

**Trzeci test:**

Porównanie wartości accuracy dla sieci z dużą liczbą neuronów



## Wyniki i wnioski

Z analizy wpływu parametrów na accuracy wynika, że bardzo małe wartości learning rate, małe wartości epochs, oraz mała liczba neuronów mają negatywny wpływ na accuracy. Dodatkowo za duże wartości learning rate  $>0.07$  powodują bardzo słabe wyniki accuracy.

Duże wartości dla batch size powodują małą utratę accuracy, natomiast bardzo małe wartości powodują dłuższe wykonywanie algorytmu jednak z minimalnym pozytywnym wpływem na accuracy, co dowodzi poprawności działania algorytmu optymalizującego.

Wartości epochs dla dobranych wartości negatywnie wpływają tylko dla małych wartości, po osiągnięciu wartości koło 20 wykres zaczyna się wypłaszczać.

Największy wpływ oraz ciekawe wartości generuje dobieranie wartości layers. Dla sieci z jedną wewnętrzną warstwą im więcej neuronów tym wyższa dokładność, która wypłaszcza po osiągnięciu 70 neuronów.

Dla sieci dwu warstwowej można zauważyć negatywny wpływ warstw o małej liczbie neuronów w szczególności dla jednego neurona, można zauważyć, że pomimo dużej liczby neuronów na poprzedniej warstwie, bardzo on

Mateusz Daszewski i Miłosz Andryszczuk

negatywnie wpływa na dokładność. Również wartości mniejsze niż 8 neuronów na warstwę negatywnie wpływają na accuracy, jednak na nie częściowo negatywnie wpływa również za duża wartość batch size jak ukazuje wykres w czwartym teście dla sieci dwuwarstwowych.

Sieci powyżej dwóch warstw nie poprawiają znacząco accuracy, jednak znacząco zwiększają czas wykonywania się programu oraz negatywny na nie wpływ mają również warstwy o za małej liczbie neuronów.