

# Sprawozdanie:

Scenariusz 3 - Budowa i działanie sieci  
wielowarstwowej.

# 1. Cel projektu:

Celem scenariusza trzeciego do zrealizowania, z przedmiotu Podstawy Sztucznej Inteligencji jest zapoznanie się z działaniem sieci neuronowej wielowarstwowej, przy pomocy stworzenia programu w oprogramowaniu MatLab. Nasz program tworzy wielowarstwową sieć neuronową, która rozpoznaje litery alfabetu.

# 2. Informacje niezbędne do realizacji projektu:

## **Sieć neuronowa:**

Zbiór neuronów, realizujących różne cele. W przypadku sztucznych sieci neuronowych jest to sztuczna struktura, zaprojektowana i zbudowana w taki sposób, aby modelowała działanie naturalnego układu nerwowego, w szczególności mózgu. Cechą wspólną wszystkich sieci neuronowych jest to, że na ich strukturę składają się neurony połączone ze sobą synapsami. Z synapsami związane są wagi, czyli wartości liczbowe, których interpretacja zależy od modelu.

## **Sieć jednokierunkowa:**

Sieć neuronowa, składająca się z neuronów ułożonych w taki sposób, aby kierunek przepływu sygnałów był jeden.

Połączenie między-warstwowe w sieci jednokierunkowej występuje tylko między kolejnymi warstwami tej sieci.

## Sieć wielowarstwowa:

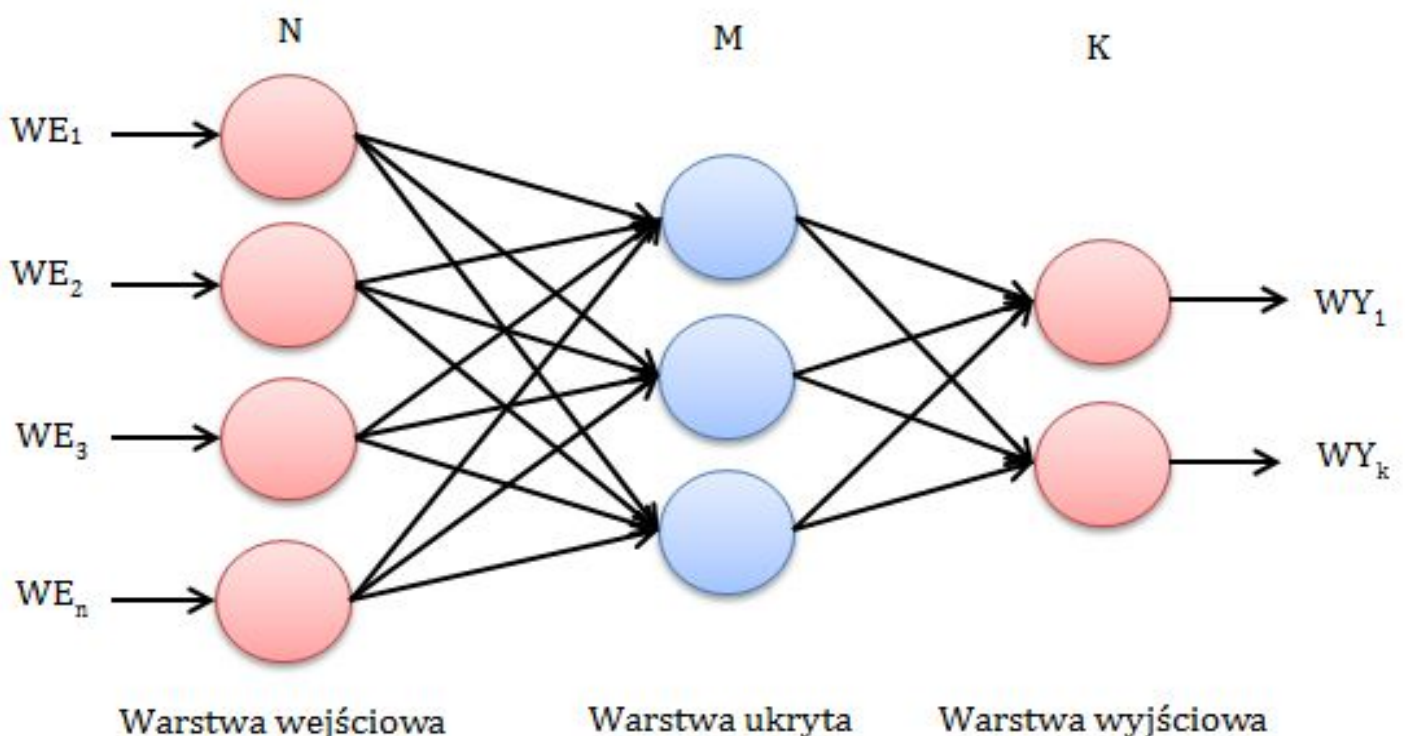
Sieć neuronowa, w której korzystamy z więcej niż dwóch warstw neuronów. Połączenie między-warstwowe w sieci jednokierunkowej występuje tylko między kolejnymi warstwami tej sieci.

Sieć jednokierunkowa posiada warstwy:

- wejściową
- wyjściową
- warstwy ukryte

Układ sieci jednokierunkowej możemy traktować, jako układ aproksymacji funkcji nieliniowej wielu zmiennych ( $y = f[u]$ ).

Schemat opisujący budowę sieci i jej warstwy:



Pierwsza warstwa zawsze jest warstwą wejściową, natomiast ostatnia wyjściową. Wszystkie warstwy pomiędzy warstwami wejściową i wyjściową to warstwy ukryte.

### **Funkcja tworzenia sieci *NEWFF*:**

Funkcja NEWFF tworzy sieć neuronową, w której każda warstwa składa się z zadanej liczby neuronów o nieliniowych funkcjach aktywacji.

Funkcja NEWFF - wywołanie w oprogramowaniu MatLab:

```
net = newff (macierz_w, [L1, L2, ... , LN], {FA1, FA2, ... , FAN}, fun_t, fun_k, fun_j );
```

#### ***macierz\_w*:**

Macierz zawierająca liczbę wejść sieci (współrzędnych wektorów wejściowych):

- pierwsza kolumna zawiera minimalne wartości kolejnych współrzędnych wektorów wejściowych.
- druga kolumna - maksymalne wartości tych współrzędnych.

#### ***[L1, L2, ... LN]*:**

Liczba neuronów kolejno w pierwszej, drugiej, ... ,N-tej warstwie sieci.

#### ***{FA1, FA2, ... , FAN}*:**

Nazwa funkcji aktywacji neuronów kolejno w pierwszej, drugiej, ..., N-tej warstwie sieci. Dopuszczalne wartości parametru TF to: 'tansig' (domyślnie) i 'logsig' i 'purelin'.

Tansig - Tangens Hiperboliczny.

*fun\_t:*

Nazwa funkcji, wykorzystywanej do treningu sieci.

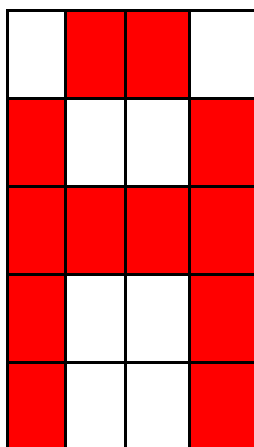
Użyta przeze mnie “traingda” to funkcja szkolenia sieci neuronowej, która aktualizuje wartości wag i odchylenia zgodnie ze spadkiem gradientu z szybkością uczenia się sieci.

### 3. Kod programu i użyte funkcje w oprogramowaniu MatLab:

→ Tworzenie wektora danych wejściowych i wyjściowych, w formie macierzy  $R \times 2$ , gdzie  $R$  to właśnie liczba wejść/wyjść sieci. Pierwsza wartość z dwóch kolumn określa minimalną wartość wektora wejściowego, a druga kolumna maksymalną wartość tego właśnie wektora.

```
MIN_MAX = [ 0 1; 0 1; 0 1; 0 1;  
            0 1; 0 1; 0 1; 0 1;  
            0 1; 0 1; 0 1; 0 1;  
            0 1; 0 1; 0 1; 0 1;  
            0 1; 0 1; 0 1; 0 1; ];
```

→ Tworzenie macierzy naszych liter. Litery przedstawione są w następujący sposób. Litery są reprezentowane przez zbiór zer i jedynek, zgodnie ze schematem ( przykład dla litery A ):



0	1	1	1	0
1	0	0	0	1
1	1	1	1	1
1	0	0	0	1
1	0	0	0	1

Kolor czerwony na rysunku pierwszym oznacza jedynki, a biały zera w naszej macierzy reprezentującej litery. Ta macierz to nasze dane wejściowe.

```

Litery_wejscie =[
0101111011111101101111;
1111111100100011111100;
111111110010001111101;
00101111011010001010;
11111111101111111011;
00000000000010000100;
000000000001000000001;
11010001010011110010;
111111111101111110010;
11001101001000111101;
11001111000010111000;
10010011010011000010;
111111111101111110010;
000000000000000000101;
000000000001000010000;
11010011010011001010;
11111101101110111000;
01111010000101001111;
01111010000101001010;
10101011011110010000;
];

```

→ Tworzenie macierzy danych wejściowych naszych liter, w postaci jedynek postawionych na kolejnych miejscach naszej macierzy. Jest to macierz jednostkowa. Np. dla litery A liczba 1 jest usytuowana na pierwszym miejscu naszej macierzy, liczba B na drugim itp. (macierz zer i jedynek).

```
Litery_wyjscie =[
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0;
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0;
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0;
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0;
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0;
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0;
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1;
];
```

→ Tworzenie danych testowych, a więc naszych liter, dla których przeprowadzać będziemy testy naszej sieci. Są to macierze poszczególnych liter, stworzone analogicznie do macierzy danych wejściowych.

```
A = [0; 1; 1; 0; 1; 0; 0; 1; 1; 1; 1; 1; 1; 0; 0; 1; 1; 0; 0; 1];
B = [1; 1; 1; 0; 1; 0; 0; 1; 1; 1; 1; 0; 1; 0; 0; 1; 1; 1; 1; 0];
C = [0; 1; 1; 1; 1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 0; 1; 1; 1];
D = [1; 1; 1; 0; 1; 0; 0; 1; 1; 0; 0; 1; 1; 0; 0; 1; 1; 1; 1; 0];
E = [1; 1; 1; 1; 1; 0; 0; 0; 1; 1; 1; 0; 1; 0; 0; 0; 1; 1; 1; 1];
F = [1; 1; 1; 1; 1; 0; 0; 0; 1; 1; 1; 0; 1; 0; 0; 0; 1; 0; 0; 0];
G = [0; 1; 1; 1; 1; 0; 0; 0; 1; 0; 1; 1; 1; 0; 0; 1; 0; 1; 1; 1];
H = [1; 0; 0; 1; 1; 0; 0; 1; 1; 1; 1; 1; 1; 0; 0; 1; 1; 0; 0; 1];
I = [1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0];
J = [1; 1; 1; 1; 0; 0; 0; 1; 0; 0; 0; 1; 1; 0; 0; 1; 0; 1; 1; 1];
```

```

K = [1; 0; 0; 1; 1; 0; 1; 0; 1; 1; 0; 0; 1; 0; 1; 0; 0; 1];
L = [1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 1; 1; 1; 1];
N = [1; 0; 0; 1; 1; 1; 0; 1; 1; 0; 1; 1; 1; 0; 0; 1; 1; 0; 0; 1];
O = [0; 1; 1; 0; 1; 0; 0; 1; 1; 0; 0; 1; 1; 0; 0; 1; 0; 1; 1; 0];
P = [1; 1; 1; 0; 1; 0; 0; 1; 1; 1; 1; 0; 1; 0; 0; 0; 1; 0; 0; 0];
R = [1; 1; 1; 0; 1; 0; 0; 1; 1; 1; 1; 0; 1; 0; 1; 0; 1; 0; 0; 1];
S = [0; 1; 1; 1; 1; 0; 0; 0; 0; 1; 1; 0; 0; 0; 0; 1; 1; 1; 1; 0];
T = [1; 1; 1; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0];
U = [1; 0; 0; 1; 1; 0; 0; 1; 1; 0; 0; 1; 1; 0; 0; 1; 0; 1; 1; 0];
Y = [1; 0; 1; 0; 1; 0; 1; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0];

```

→ Tworzenie wektora, który określa liczbę neuronów w każdej sieci. Dla naszej sieci w pierwszej i ostatniej warstwie sieci będzie 40 neuronów, natomiast w warstwie ukrytej 20. Danymi wejściowymi naszej sieci są litery ( macierz ), liczba wszystkich liter to 20 oraz dla każdej litery możemy określić 0 lub 1 ( macierz MAX\_MIN ), a więc dwie wartości dla każdej z 40 liter daje nam wartość neuronów na wejściu 40. Analogicznie jest z warstwą wyjściową. W warstwie ukrytej znajduje się 20 neuronów, ponieważ nasze dane testujące sieć neuronową to 20 liter.

```

Ilosc_neuronow_warstwy =[40 20 20];

```

→ Tworzenie naszej sieci neuronowej, w której określaliśmy wektor danych wejściowych i wyjściowych (MIN\_MAX), ilość neuronów w każdej z warstw naszej sieci (Ilosc\_neuronow\_warstwy), a także funkcję aktywacji neuronów (wyżej omawiany *tansig*) i funkcję nauki (*traingda*).

```

net = newff(MIN_MAX,Ilosc_neuronow_warstwy,{'tansig','tansig','tansig'},'traingda');

```



→ Określamy parametry uczenia się naszej sieci neuronowej. Maksymalna liczba epok ( iteracji ), wynosi 7000 ponieważ zazwyczaj “przeuczenie” sieci następuje już po ok. 4500 - 5500 powtórzeniu. Zarówno błąd średniokwadratowy oraz współczynnik uczenia wynoszą 0.001.

```
net.trainParam.epochs = 7000;  
net.trainParam.mu = 0.001;  
net.trainParam.goal = 0.001;
```

→ Wywołujemy uczenie się naszej sieci funkcją *train*. Argumentami tej funkcji są: nasza sieć neuronowa -stworzona wyżej, macierz danych wejściowych (*littery\_wejscie*) oraz macierz danych wyjściowych (*littery\_wyjście*).

```
net = train(net, Littery_wejscie, Littery_wyjście);
```

→ Przeprowadzamy symulację naszej sieci dla wybranej litery.

```
symulacja =sim(net, I);
```

## 4. Wyjście programu:

Program jako wyjście wypisuje nam literę, którą wprowadziliśmy ( o ile sieć działa poprawnie ). Jeżeli testujemy program np. dla litery A, to nasza sieć rozpozna tą literę i program wypisze komunikat:

*Litera: A.*

## 5. Wnioski:

Podczas testowania różnych liter łatwo jest zauważyć, że im więcej jedynek w macierzy określającej daną literę tym więcej iteracji potrzeba podczas przeprowadzenia symulacji sieci. Przykładowo dla litery A podczas symulacji potrzeba aż o 1500 iteracji więcej niż dla litery I ( litera A to macierz gdzie jest aż 12 jedynek, natomiast macierz litery I składa się tylko z 4 jedynek ).

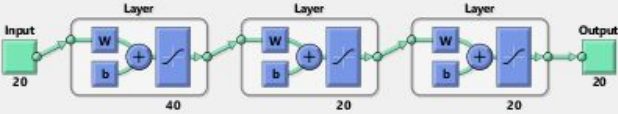
Test mojego programu pokazał, że podczas zmian wartości błędu średniokwadratowego oraz współczynnika uczenia ma bardzo duży wpływ na szybkość uczenia się ( nawet o 2000 iteracji mniej dla zwiększeniu tych współczynników dziesięciokrotnie ). Wraz ze spadkiem liczby iteracji spada niestety też jakość nauki.

Porównanie nauki dla wartości błędu średniokwadratowego oraz współczynnika uczenia równych 0.001 i 0.01.

parametry = 0.001 :

Neural Network Training (nntraintool)

Neural Network



Algorithms

Training:

Gradient Descent with Adaptive Learning Rate (traingda)

Performance:

Mean Squared Error (mse)

Calculations:

MEX

Progress

Epoch:

0

3930 iterations

7000

Time:

0:00:05

Performance:

0.638

0.000994

0.00100

Gradient:

0.278

0.00780

1.00e-05

Validation Checks:

0

0

6

Plots

Performance

(plotperform)

Training State

(plottrainstate)

Regression

(plotregression)

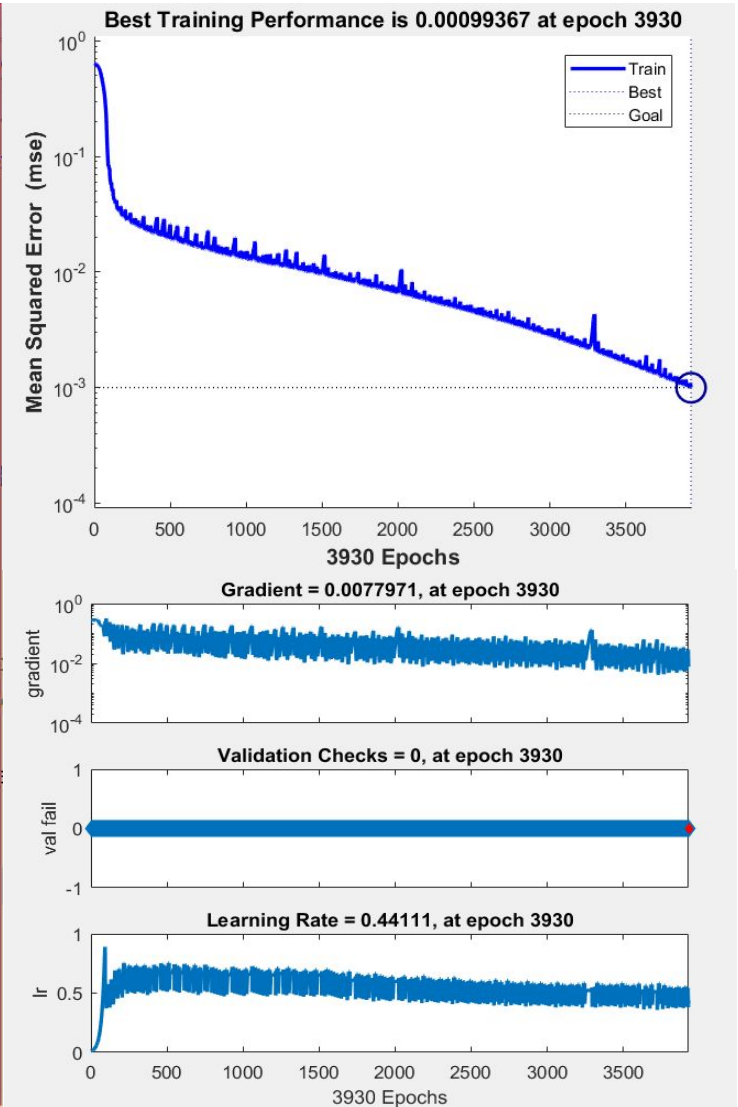
Plot Interval:

1 epochs

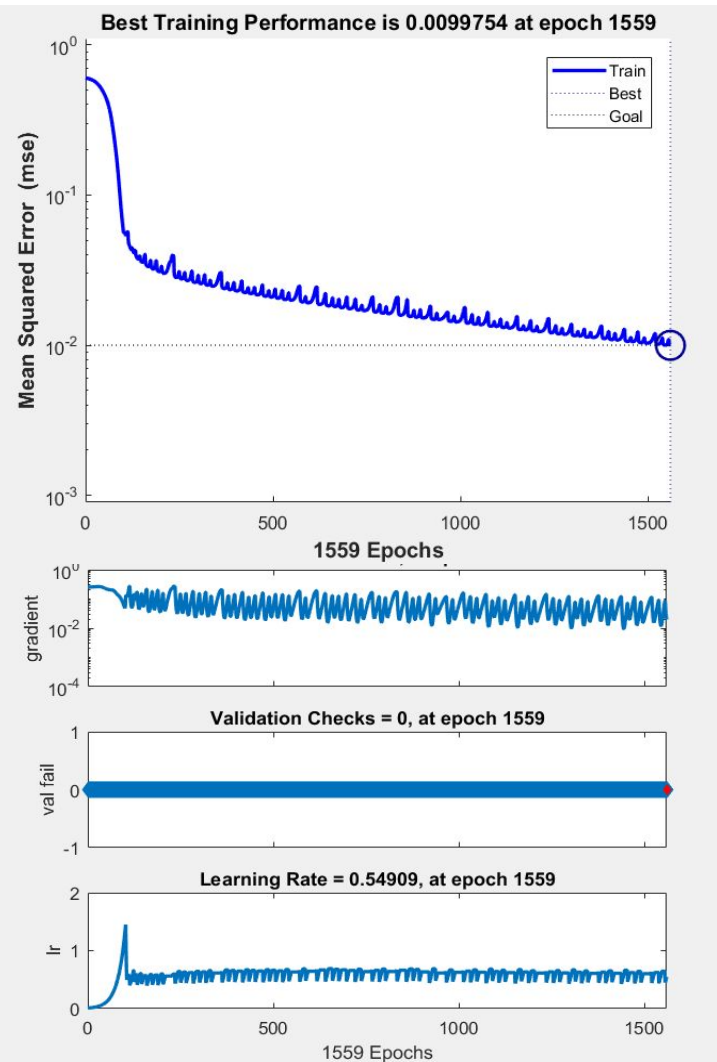
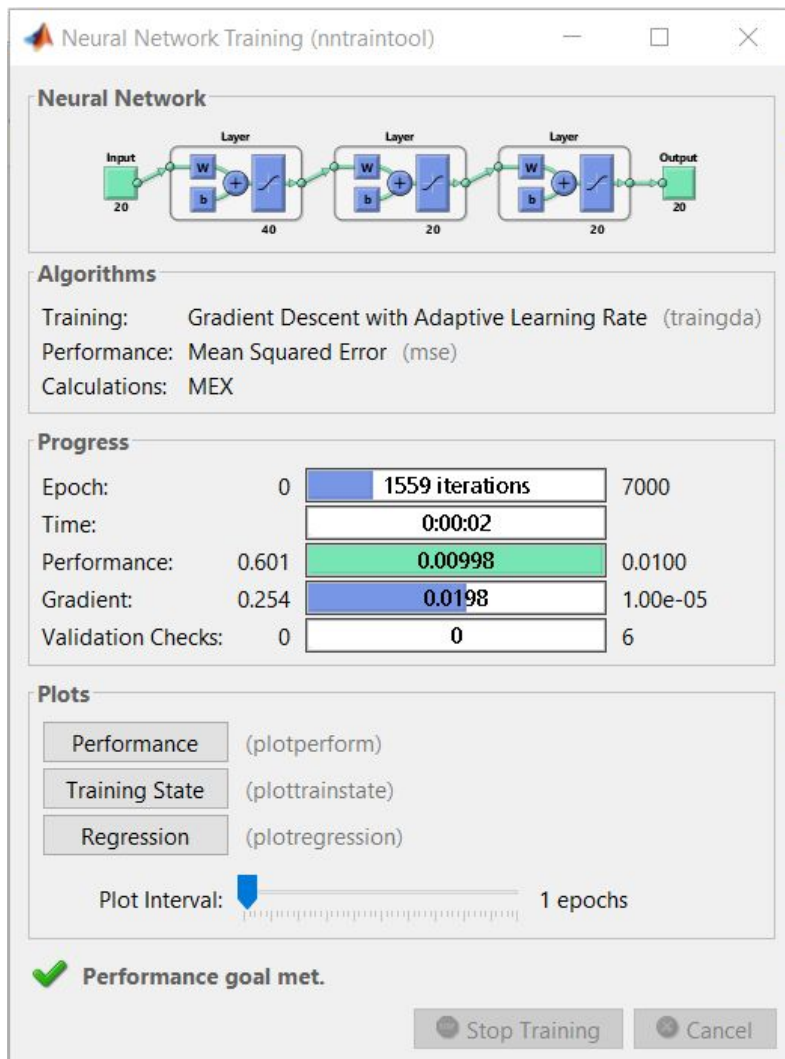
Performance goal met.

Stop Training

Cancel



parametry = 0.01



## 6. Listing kodu programu:

```
close all; clear all; clc;
MIN_MAX=[0 1; 0 1; 0 1; 0 1;
0 1; 0 1; 0 1; 0 1;
0 1; 0 1; 0 1; 0 1;
0 1; 0 1; 0 1; 0 1;];
Litery_wejscie =[
0 1 0 1 1 1 0 1 1 1 1 1 1 0 1 1 0 1 1 1 1;
1 1 1 1 1 1 1 1 0 0 1 0 0 0 1 1 1 1 1 0 0;
1 1 1 1 1 1 1 1 0 0 1 0 0 0 1 1 1 1 1 0 1;
0 0 1 0 1 1 1 1 0 1 1 1 0 1 0 0 0 1 0 1 0;
1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1;
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0;
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1;
1 1 0 1 0 0 0 1 0 1 0 0 1 1 1 1 1 0 0 1 0;
1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 1 0;
1 1 0 0 1 1 0 1 0 0 1 0 0 0 1 1 1 1 1 0 1;
1 1 0 0 1 1 1 1 1 0 0 0 0 1 0 1 1 1 1 0 0 0;
1 0 0 1 0 0 1 1 0 1 0 0 1 1 0 0 0 0 1 0;
1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 1 0;
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1;
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0;
1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1 0 1 0;
1 1 1 1 1 1 0 1 1 0 1 1 1 1 0 1 1 1 1 0 0 0;
0 1 1 1 1 0 1 0 0 0 0 1 0 1 0 0 1 1 1 1;
0 1 1 1 1 0 1 0 0 0 0 1 0 1 0 0 1 0 1 0;
1 0 1 0 1 0 1 1 0 1 1 1 1 1 0 0 1 0 0 0 0;
];
Litery_wyjscie =[
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0;
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0;
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0;
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0;
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0;
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0;
```

```

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0;
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1;
];
A = [0; 1; 1; 0; 1; 0; 0; 1; 1; 1; 1; 1; 1; 0; 0; 1; 1; 0; 0; 1];
B = [1; 1; 1; 0; 1; 0; 0; 1; 1; 1; 1; 0; 1; 0; 0; 1; 1; 1; 1; 0];
C = [0; 1; 1; 1; 1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 1; 1; 1];
D = [1; 1; 1; 0; 1; 0; 0; 1; 1; 0; 0; 1; 1; 0; 0; 1; 1; 1; 0];
E = [1; 1; 1; 1; 1; 0; 0; 0; 1; 1; 1; 0; 1; 0; 0; 0; 1; 1; 1];
F = [1; 1; 1; 1; 1; 0; 0; 0; 1; 1; 1; 0; 1; 0; 0; 0; 1; 0; 0];
G = [0; 1; 1; 1; 1; 0; 0; 0; 1; 0; 1; 1; 1; 0; 0; 1; 0; 1; 1];
H = [1; 0; 0; 1; 1; 0; 0; 1; 1; 1; 1; 1; 1; 0; 0; 1; 1; 0; 0];
I = [1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0];
J = [1; 1; 1; 1; 0; 0; 0; 1; 0; 0; 0; 1; 1; 0; 0; 1; 0; 1; 1];
K = [1; 0; 0; 1; 1; 0; 1; 0; 1; 1; 0; 0; 1; 0; 1; 0; 1; 0; 0];
L = [1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 1; 1; 1];
N = [1; 0; 0; 1; 1; 1; 0; 1; 1; 0; 1; 1; 1; 0; 0; 1; 1; 0; 0];
O = [0; 1; 1; 0; 1; 0; 0; 1; 1; 0; 0; 1; 1; 0; 0; 1; 0; 1; 1];
P = [1; 1; 1; 0; 1; 0; 0; 1; 1; 1; 1; 0; 1; 0; 0; 0; 1; 0; 0];
R = [1; 1; 1; 0; 1; 0; 0; 1; 1; 1; 1; 0; 1; 0; 1; 0; 1; 0; 0];
S = [0; 1; 1; 1; 1; 0; 0; 0; 1; 1; 0; 0; 0; 0; 1; 1; 1; 1; 0];
T = [1; 1; 1; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 1; 0];
U = [1; 0; 0; 1; 1; 0; 0; 1; 1; 0; 0; 1; 1; 0; 0; 1; 0; 1; 1];
Y = [1; 0; 1; 0; 1; 0; 1; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 1; 0];
llosc_neuronow_warstwy =[40 20 20];
net = newff(MIN_MAX,llosc_neuronow_warstwy,
{'tansig','tansig','tansig'},'traingda');
net.trainParam.epochs = 7000;
net.trainParam.mu = 0.01;
net.trainParam.goal = 0.01;
net = train(net, Litery_wejscie, Litery_wyjście);
symulacja =sim(net, R);
max=1;
for i=1:1:20
if (symulacja(max)<symulacja(i))
max=i;
end;
end
switch max
case 1
disp('Litera : A')
disp(symulacja(1))
case 2
disp('Litera : B')
disp(symulacja(2))
case 3
disp('Litera : C')
disp(symulacja(3))
case 4
disp('Litera : D')
disp(symulacja(4))
case 5

```

```
disp('Litera : E')
disp(symulacja(5))
case 6
disp('Litera : F')
disp(symulacja(6))
case 7
disp('Litera : G')
disp(symulacja(7))
case 8
disp('Litera : H')
disp(symulacja(8))
case 9
disp('Litera : I')
disp(symulacja(9))
case 10
disp('Litera : J')
disp(symulacja(10))
case 11
disp('Litera : K')
disp(symulacja(11))
case 12
disp('Litera : L')
disp(symulacja(12))
case 13
disp('Litera : N')
disp(symulacja(13))
case 14
disp('Litera : O')
disp(symulacja(14))
case 15
disp('Litera : P')
disp(symulacja(15))
case 16
disp('Litera : R')
disp(symulacja(16))
case 17
disp('Litera : S')
disp(symulacja(17))
case 18
disp('Litera : T')
disp(symulacja(18))
case 19
disp('Litera : U')
disp(symulacja(19))
case 20
disp('Litera : Y')
disp(symulacja(20))
otherw
ise
disp('error error')
end
```