Daniel Wysowski
Nr. albumu: 286136
Inżynieria Obliczeniowa
Wydział Inżynierii Metali i Informatyki Przemysłowej
Akademia Górniczo-Hutnicza w Krakowie

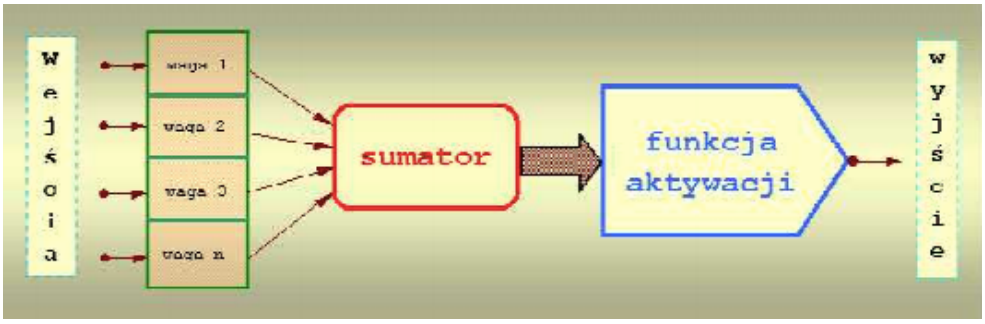
Sprawozdanie z przedmiotu: Podstawy sztucznej inteligencji

Scenariusz 3 - Budowa i działanie sieci wielowarstwowej.

Spis treści:

I.WSTĘP TEORETYCZNY

Neuron – jest to jeden z matematycznych modeli neuronu oraz podstawowy budulec wspomnianej wyżej sieci neuronowej (perceptron). Posiada wiele wejść gdzie każde ma swoją wagę (tzw. Wagę wejścia), oraz jedno wyjście którego waga jest obliczona dodatkowo za pomocą wzoru (funkcja aktywacji).



Sieć wielowarstwowa - aby zaprojektować sieć neuronową należy przeanalizować sformułowanie problemu, chodzi tu przede wszystkim o podanie specyfikacji wejścia sieci - ile oraz jakie typy danych chcemy podać a także jaką odpowiedź chcemy uzyskać na wyjściu. Pozostaje jeszcze kwestia określenia ilości warstw ukrytych oraz neuronów znajdujących się w tych warstwach - jest to wymagający moment podczas wykonywania tego projektu. Sieć jednowarstwowa składająca się z jednej warstwy ukrytej powinna nauczyć się rozwiązywać większą ilość postawionych przed nią problemów. Na chwilę obecną nie znane są problemy potrzebujące do swojego rozwiązania sieci z więcej niż trzema warstwami ukrytymi. Niestety na chwilę obecną nie znany jest sposób na obliczenie właściwej ilości neutronów w warstwie ukrytej. Przybliżoną ich ilość można próbować uzyskać za pomocą wzoru:

$$N_{wu} = \sqrt{N_{wwe} * N_{wwy}}$$

gdzie:
Nwu - ilość neuronów w warstwie ukrytej
Nwwe - ilość neuronów w warstwie wejściowej
Nwwy - ilość neuronów w warstwie wyjściowej

Działanie SSN polega na tym, że sygnały pobudzające (wektor wejściowy) podawane na wejścia sieci, przetwarzane są w poszczególnych neuronach. Po tej projekcji na wyjściach sieci otrzymuje się wartości liczbowe, które stanowią odpowiedź sieci na pobudzenie i stanowią rozwiązanie postawionego problemu. Jednak aby takie rozwiązanie uzyskać, należy przejść żmudną drogę uczeń.

ANALIZA FUNKCJI:

Funkcja tworzenia sieci NEWFF :

KOD:

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

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

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.

II. CEL ĆWICZENIA

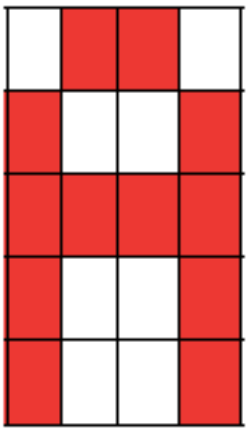
W tym scenariuszu moim celem było zapoznanie się z działaniem sieci neuronowej wielowarstwowej, przy pomocy jego implementacji w środowisku MatLab. Nasz program tworzy wielowarstwową sieć neuronową, która rozpoznaje litery alfabetu.

III.Wykonanie zadania + opis działania

Jest to model wprowadzania danych wejściowych i wyjściowych, w formie macierzy Rx2, R to liczba wejść oraz wyjść. Wartości znajdujące się w pierwszej kolumnie określają minimalną wartość wektora zaś druga maksymalną wartość tego 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; ];
```

Poniżej znajduje się schemat macierzy w której mamy informacje na temat temat zbioru liter podanych na wejściu. Litery reprezentowane są w tym przypadku przez zbiór zer oraz jedynek.



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

Na schemacie znajdującym się po lewej stronie - czerwone kwadraciki oznaczają odpowiednio jedynki z macierzy znajdującej się po prawej stronie. Białe pola oznaczają zera w macierzy reprezentującej litery.

Macierz ta, to nasz dane wejściowe, poniżej kod programu:

```
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 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 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 0 1 1;
            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 1;
            1 1 0 1 0 0 0 1 0 1 0 0 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 0 0 1 0;
            1 1 0 0 1 1 0 1 0 0 1 0 0 0 1 1 1 1 0 1;
            1 1 0 0 1 1 1 1 0 0 0 0 1 0 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 0 0 1 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 1 0 1 1 0 1 1 1 0 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 0 0 1 0 0 0 0; ];
```

Poniżej znajduje się 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; ];
```

Aby przeprowadzić testy naszej sieci, musimy utworzyć zestaw danych wejściowych a zarazem testowych według schematu z pierwszej macierzy zawierających informacje na temat każdej z liter.

```
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; 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];
```

KOD:

```
Ilosc_neuronow_warstwy =[40 20 20];
```

Utworzenie wektora pozwala nam określić liczbę neuronów w sieci. W tym przypadku w pierwszej i ostatniej warstwie sieci, będzie 40 neuronów zaś w warstwie ukrytej 20. Dla naszych danych wejściowych można określić dwie wartości tj. 0 oraz 1. Od ilości liter zależna jest również niejako ilość neuronów. Tym samym, na wejściu posiadamy 20 liter dla każdej możemy przypisać dwie wartości otrzymując w sumie wartość 40 neuronów na wejściu.

KOD:

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

Tworzenie sieci neuronowej w której zostały określone wszystkie jej parametry. A także funkcja aktywacji neuronów oraz funkcja nauki.

KOD:

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

W tym miejscu określamy parametry uczące naszej sieci neuronowej. Maksymalna liczba epok - iteracji, wynosi 7000, po zbliżeniu do tej liczby zazwyczaj zachodzi przeuczenie. Przy wartościach 4500-5500 zarówno błąd średniokwadratowy jak i współczynnik uczenia wynoszą 0.001.

KOD:

```
net = train(net, Litery_wejscie, Litery_wyjscie);
```

Jak zawsze uczenie naszej sieci rozpoczynamy używając funkcji train. Argumentami funkcji są: nasza sieć neuronowa, macierz danych wejściowych oraz macierz danych wyjściowych.

KOD:

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

W tej linii kodu przeprowadzamy symulację sieci neuronowej.

IV. PODSUMOWANIE

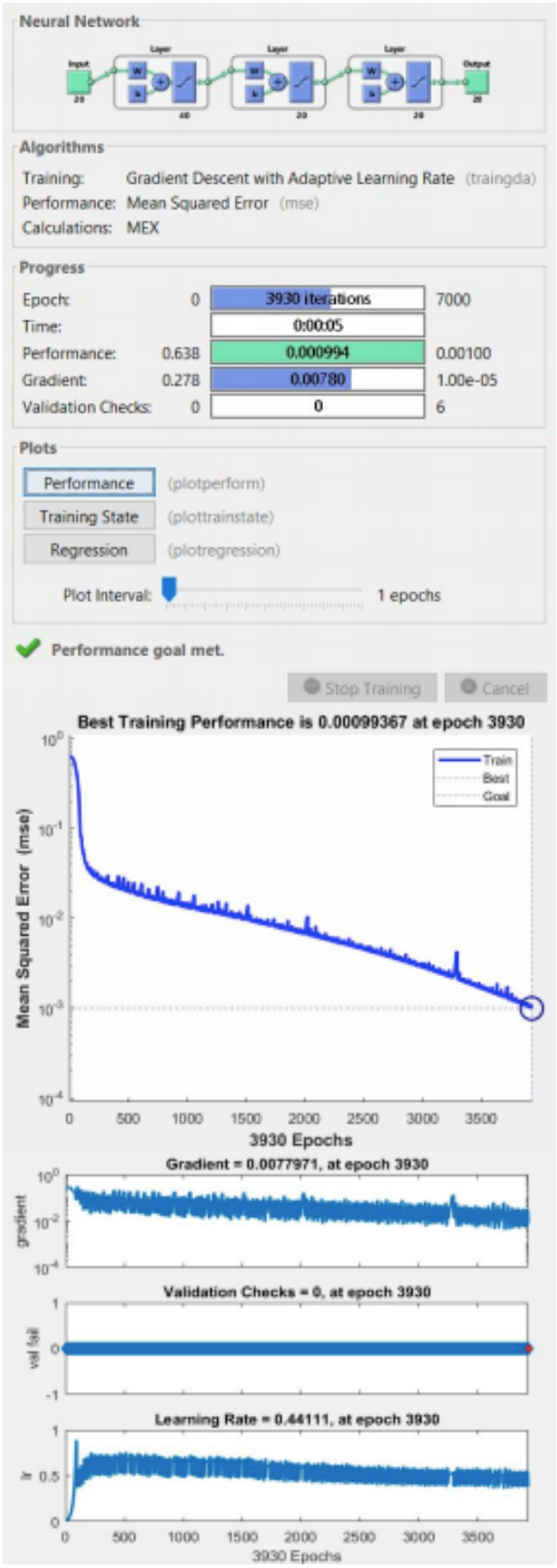
WYNIKI:

Na wyjściu naszego programu otrzymujemy wypisaną literę, którą wprowadziliśmy. Przykładem działania może być np. Jeżeli podamy naszej sieci literę „C” to nasza sieć powinna na wyjściu powiadomić nas komunikatem np. „ To litera C”.

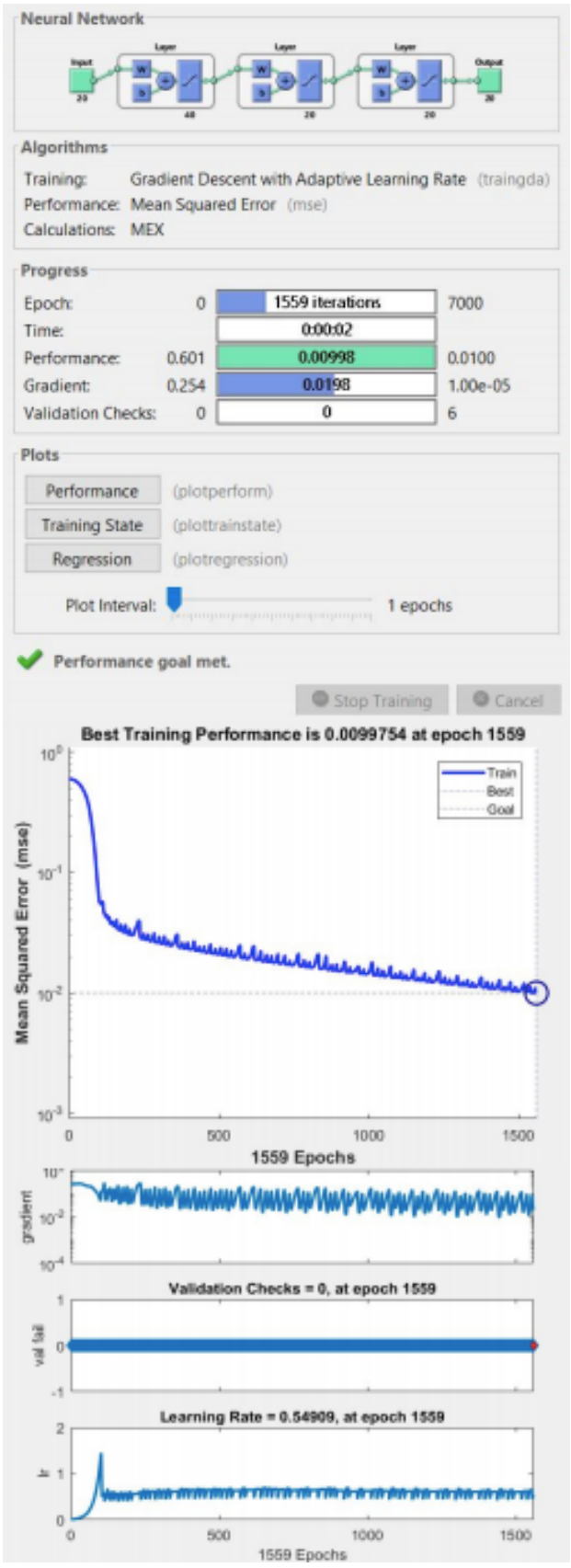
WNIOSKI:

Po przetestowaniu różnych wariantów liter można łatwo dostrzec że im więcej jedynek w macierzy tym więcej iteracji potrzeba podczas przeprowadzenia symulacji sieci. Przykładowo podczas symulacji sieci dla litery A potrzeba aż 1500 iteracji więcej niż dla litery I. W moim programie test pokazał że podczas zmian wartości błędu średnio-kwadratowego oraz współczynnika uczenia ma bardzo duży wpływ na szybkość uczenia się - nawet 2000 iteracji. Jeżeli spada ilość iteracji spada również jakość nauki.

Screeny dla parametru: 0.001



Screeny dla parametru: 0.01:



V.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; 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 0 0 1 0 0 0 1 1 1 1 1 0 0;
 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 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 0 1 1;
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 1;
 1 1 0 1 0 0 0 1 0 1 0 0 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 0 0 1 0;
1 1 0 0 1 1 0 1 0 0 1 0 0 0 1 1 1 1 0 1;
1 1 0 0 1 1 1 1 0 0 0 0 1 0 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 0 0 1 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 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 1 0 1 1 0 1 1 1 0 1 1 1 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 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 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; ];
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; 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];

Ilosc_neuronow_warstwy =[40 20 20];
net = newff(MIN_MAX,Ilosc_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_wyjscie); symulacja =sim(net, R);
max=1;
for i=1:1:20 if (symulacja(max)
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(symulacjat(18))
case 19
disp('Litera : U')
disp(symulacja(19))
case 20
disp('Litera : Y')
disp(symulacja(20))
otherw
ise
disp('error error') end
```