

Sprawozdanie z laboratorium sztucznej inteligencji

Informatyka, sem. VI

Symulator sieci typu Echo State Network

I. Opis zadania

Tematem projektu było zaimplementowanie symulatora sieci neuronowej typu *Echo State Network*. W ramach tego zadania stworzono klasę stanowiącą implementację tej sieci neuronowej oraz aplikację okienkową stanowiącą interfejs do obsługi klasy sieci neuronowej.

System umożliwia poprzez obsługę aplikacji okienkowej tworzenie nowych sieci neuronowych typu *Echo State Network*, zapisywanie i wczytywanie tych sieci, obsługę procesu uczenia i zadawania pytań sieci.

Do danych wejściowych programu zaliczamy pliki tekstowe o określonym formacie danych, które są wykorzystywane do uczenia sieci, oraz pliki o rozszerzeniu .esn, które stanowią zapisane instancje klasy *EchoStateNetwork*.

Dane wyjściowe obejmują pliki o rozszerzeniu .esn, które reprezentują instancje klasy *EchoStateNetwork*. Zapisana w ten sposób sieć przechowuje swoje właściwości i można ją przenosić pomiędzy urządzeniami, lub wczytać po ponownym uruchomieniu aplikacji.

II. Założenia realizacyjne

1. Założenia dodatkowe.

W procesie implementacji podjęto następujące założenia dotyczące projektu:

- Program umożliwia tworzenie tylko sieci o określonym rozmiarze i stopniu wycieku. Rozmiar sieci jest określony na 1000 neuronów, a stopień wycieku ma wartość 0.3.
- Proces uczenia sieci następuje jedynie przy procesie wczytania danych wejściowych i wciśnięciu odpowiedniego przycisku w interfejsie aplikacji.
- Sieć można uczyć wielokrotnie.
- W celu odpytania sieci należy użyć aplikacji okienkowej, w której będzie aktualnie otwarta wyuczona sieć. Podczas zadawania pytania sieć nie podlega procesowi uczenia.
- Dane wejściowe do uczenia powinny być dostarczone w postaci pliku tekstowego o dowolnej nazwie i formacie rozszerzenia .txt. Treść pliku powinna wyglądać następująco:

$x_1 ; y_1$
$x_2 ; y_2$
...
$x_n ; y_n$

,gdzie x_n i y_n stanowią kolejne liczby zmiennoprzecinkowe ciągu elementów określonej funkcji ciągłej.

2. Metody, strategie oraz algorytmy.

- **Algorytm inicjalizacji sieci *Echo State Network***

Dane: n – liczba neuronów, a – stopień wycieku sieci neuronowej

Wynik: Macierz kwadratowa wielkości n reprezentująca wagi połączeń neuronów sieci

Algorytm postępowania:

1. Generacja wektora wejścia
2. Generowanie losowego rezerwuaru
3. Obliczanie parametru ρ
4. Szukanie największej wartości wśród bezwzględnych wartości własnych macierzy W
5. Normalizacja macierzy
6. Generowanie stanów przejściowych
7. Generacja i trenowanie wektora wyjścia
8. Nauczanie sieci ESN
9. Generowanie błędu średnio kwadratowego
10. Sygnał wyjściowy

3. Język programowania, narzędzia informatyczne i środowiska używane do implementacji systemu.

Zgodnie z założeniami przedstawionymi przy doborze tematu pracy projektu implementację można było wykonać w języku C# lub Java. Na podstawie umiejętności i doświadczenia zespołu podjęto decyzję o przeprowadzeniu implementacji sieci *Echo State Network* w języku C#. Wcześniejsze doświadczenia zespołu obejmują liczne projekty i zadania wykonywane samodzielnie i w zespołach. Dotyczą one, także zainteresowań każdego członka zespołu, który rozwijał się w kierunku programowania w języku C# oraz technologii związanych z tym językiem.

W celu implementacji projektu symulatora sieci *Echo State Network* zespół użył środowiska *Microsoft Visual Studio*, które jest zalecanym środowiskiem rozwoju oprogramowania w języku C#. Zespół wykorzystał program *Visual Studio* w wersji z roku 2015, a w implementacji interfejsu skorzystał z technologii *Windows Presentation Foundation* do utworzenia aplikacji okienkowej.

W celu wykonywania obliczeń na macierzach wykorzystana została biblioteka *Math.NET*, która zapewniła obsługę obliczeń algebry liniowej.

Do tworzenia korpusów danych wykorzystane został program własnego autorstwa zespołu, który obliczał wartości wskazanej funkcji w określonym przedziale z ustalonym krokiem. Ta aplikacja konsolowa zapisywała wyniki swojej pracy w pliku tekstowym dostosowanym w formacie do wczytania przez symulator sieci neuronowej zaimplementowanego w ramach projektu.

III. Podział prac

Autor	Podzadanie
Szymon Kaszuba	Aplikacja okienkowa, obsługa i funkcjonalności
Łukasz Knop	Generowanie korpusów danych, testowanie sieci
Adam Matuszak	Implementacja sieci neuronowej

Przedstawiony podział pracy pomiędzy członków zespołu stanowi pogląd generalny na przydzielone zadania. Poszczególne fragmenty były głównymi zadaniami pracy każdego członka zespołu, nie stanowiły jednak jego jedynych zadań, a jego praca nie ograniczała się jedynie do przedstawionych w tabeli podzadań.

IV. Opis implementacji

1. Struktury danych

1. Klasa EchoStateNetwork

Klasa reprezentująca obiekt sieci neuronowej typu Echo State Network. Zawiera między innymi takie struktury jak macierz wejścia, macierz rezerwuaru sieci neuronowej, macierz powłoki wyjścia, macierz oczekiwanych wartości i macierz wyjścia oraz korpus danych w strukturze o klasie Data.

2. Klasa Data

Klasa reprezentuje korpus danych wczytanych ze wskazanego przez użytkownika pliku. Dane przechowywane są w formie listy obiektów klasy Input.

3. Klasa Input

Klasa reprezentuje pojedynczy element danych z korpusu. Zawiera parę wartości 'x' i 'y', które stanowią kolejno wartość wejściową oraz wartość spodziewaną. Klasa posiada także funkcję zwrotu wartości x i y w formie jednowymiarowej tablicy liczb zmiennoprzecinkowych.

2. Funkcje i procedury

1. EchoStateNetwork.InitializeParameters

Wejście int resSize, double alpha

Działanie procedura inicjuje powstanie rezerwuaru sieci neuronowej o wielkości resSize i stopniu wycieku wielkości alpha

2. EchoStateNetwork.NormalizeWeights

Działanie Procedura służy normalizacji wag neuronów w rezerwuarze sieci

3. EchoStateNetwork.Learn

Wejście string path, int ignoredInitialResults

Działanie Procedura inicjuje wczytanie korpusu danych z pliku w lokalizacji path i rozpoczęcie procesu nauczania z pominięciem ignoredInitialResults pierwszych wyników.

4. EchoStateNetwork.InputDataIntoReservoir

5. Wejście int ignoredDataCount

Działanie Procedura wprowadzenia danych korpusu do rezerwuaru sieci z pominięciem ignoredDataCount pierwszych wyników

6. EchoStateNetwork.LearnProcess

Wejście int initResult, int ignoredDataCount

Działanie Proces uczenia sieci na podstawie otrzymanych danych z pominięciem initResult wstępnych wyników i zadeklarowanej liczbie ignoreDataCount wyników. W ramach działania funkcji obliczany jest także błąd kwadratowy.

7. EchoStateNetwork.Ask

Wejście double inputData

Wyjście double y

Działanie Funkcja zwraca wartość otrzymaną w wyniku działania sieci dla danej wejściowej inputData.

8. EchoStateNetwork.JoinArrays

Wejście double[] a, double[] b

Wyjście double[] result

Działanie Funkcja dokonuje złączenia dwóch jednowymiarowych tablic.

9. Data.GetExpetedOutputArray

Wejście int start, int lenght

Wyjście double[] result

Działanie Funkcja zwraca zadany przedział spodziewanych wartości od pozycji początkowej start o wielkości lenght.

10. Input.ToArray

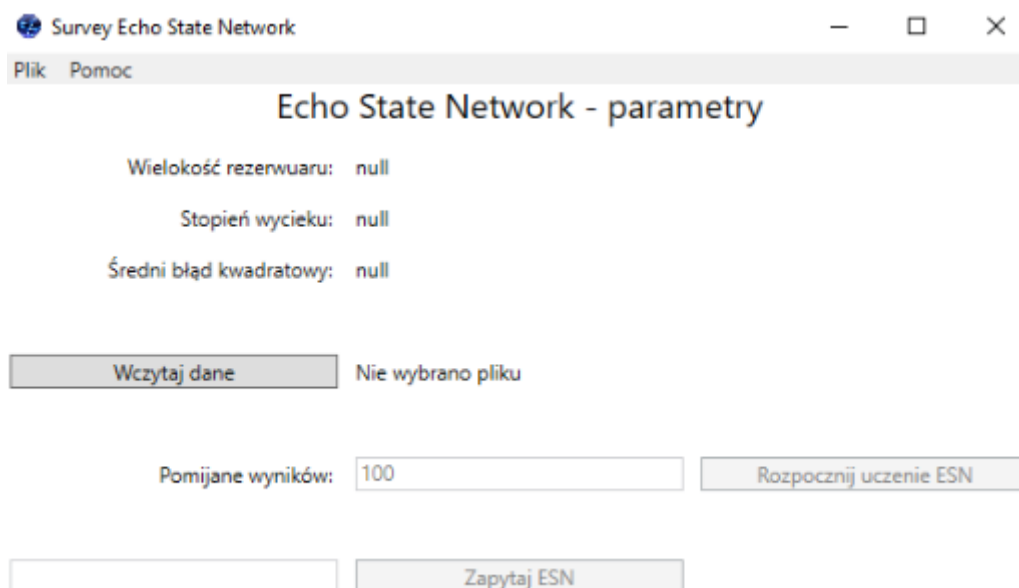
Wyjście double[] d

Działanie Funkcja zwraca wartość wejściową i spodziewana w formie jednowymiarowej tablicy liczb zmiennoprzecinkowych.

V. Użytkowanie i testowanie

1. Użytkowanie

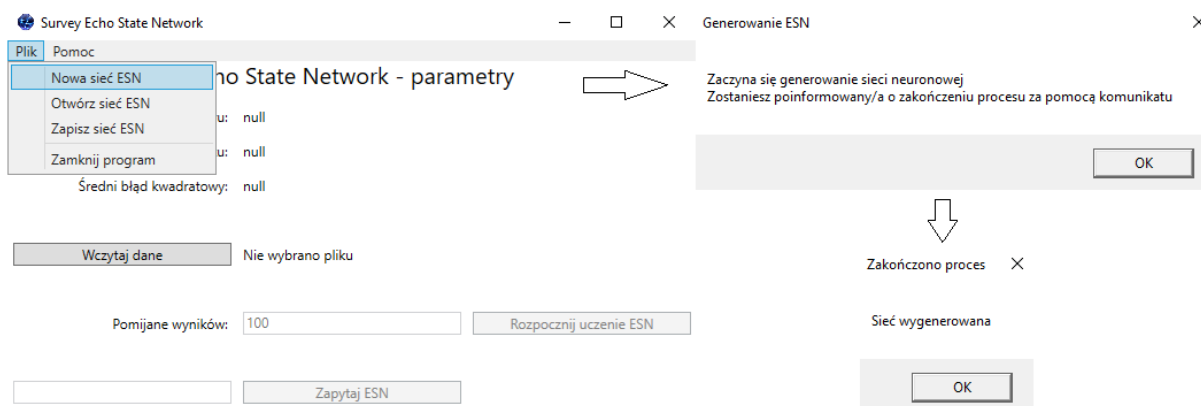
1. Interfejs programu



Rysunek 1 Okno symulatora sieci ESN

Zrzut ekranu przedstawia główne okno aplikacji symulatora sieci neuronowej typu *Echo State Network*. Aplikacja posiada pasek menu do zarządzania aplikacją. Panel główny okna zawiera kontrolki dotyczące zarządzania i modyfikacji obiektu reprezentującego sieć neuronową *Echo State network*.

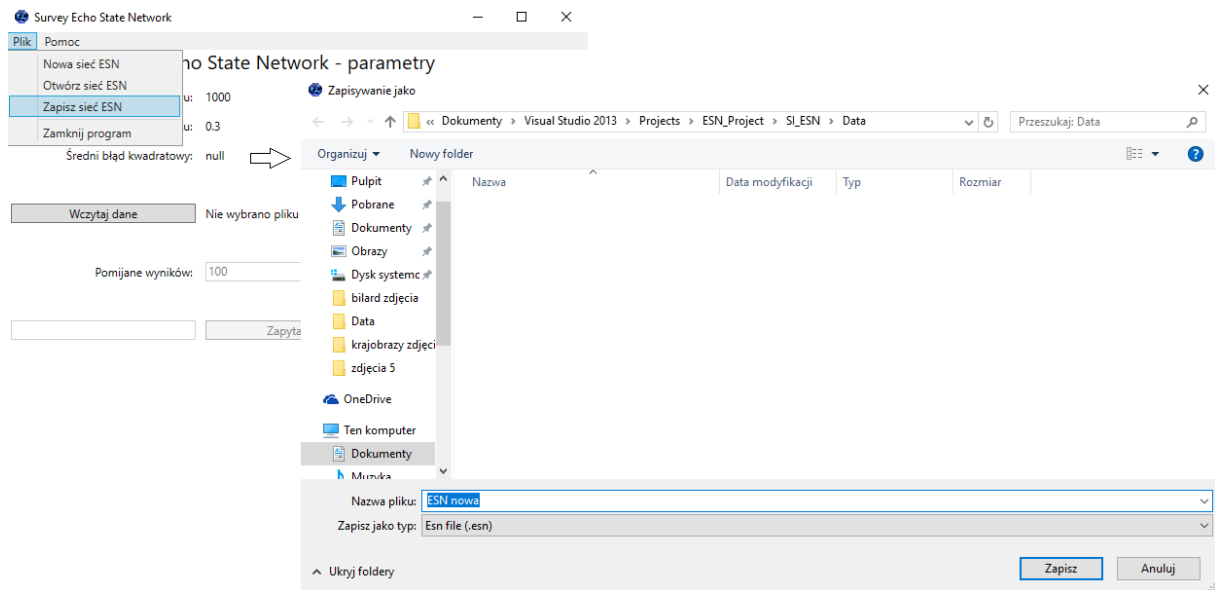
2. Tworzenie nowej sieci



Rysunek 2 Proces tworzenia nowej sieci

W celu utworzenia nowego obiektu sieci neuronowej z zakładki Plik wybieramy opcję Nowa sieć ESN. Po kliknięciu zostaniemy poinformowani, że rozpoczyna się generowanie sieci zakończenie procesu zostanie użytkownikowi zasygnalizowane. Po pomyślnym utworzeniu ESN zatwierdzamy komunikat, aby przejść do dalszego użytkowania.

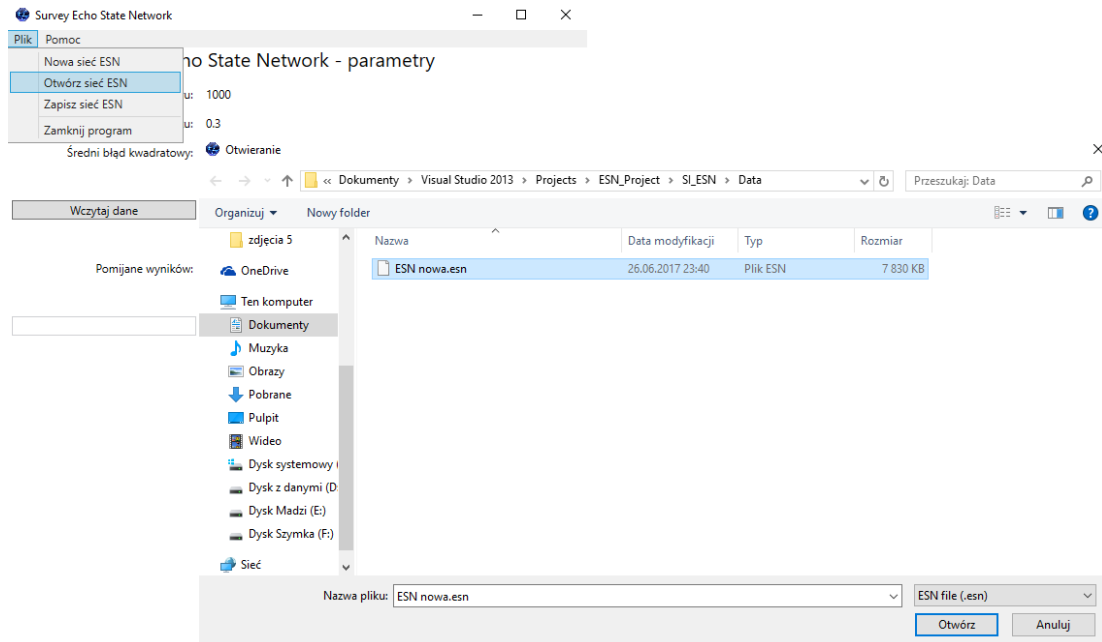
3. Zapisywanie sieci do pliku



Rysunek 3 Zapisywanie obiektu sieci do pliku

Aby zapisać obiekt sieci neuronowej do pliku w celu przeniesienia jej lub wczytania w innym momencie z zakładki Plik w menu głównym wybieramy opcję Zapisz sieć ESN. Następnie używając standardowego okna Windows do zapisu plików umieszczany jest nowy plik o rozszerzeniu .esn w określonej przez użytkownika lokalizacji.

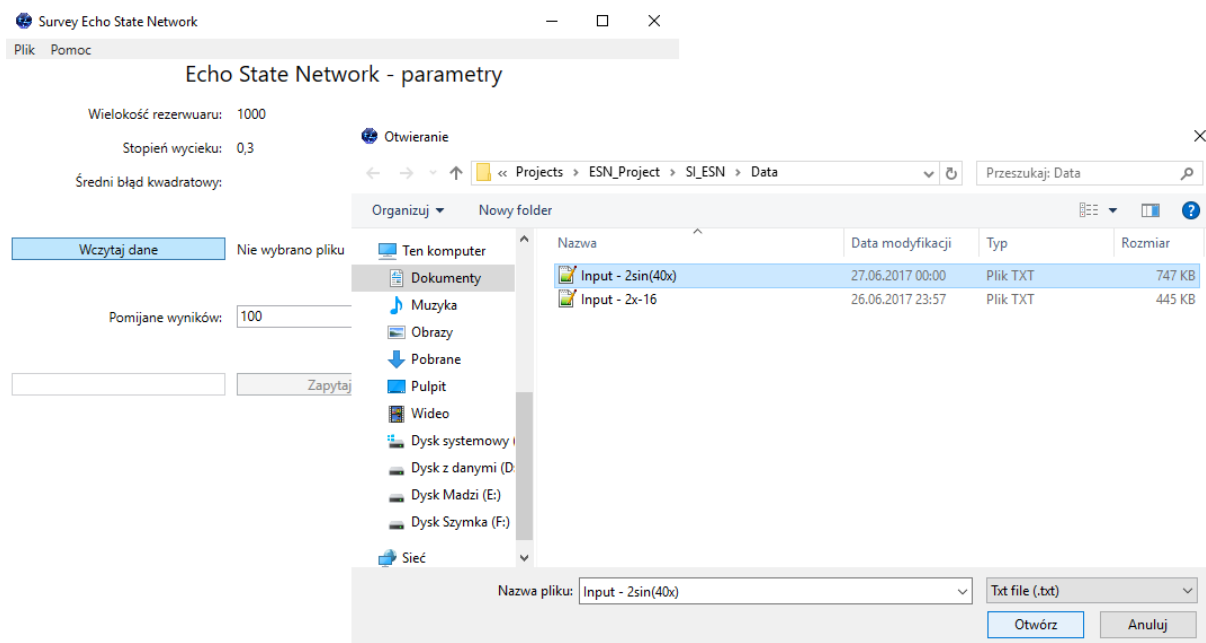
4. Wczytywanie sieci z pliku



Rysunek 4 Wczytywanie pliku

Program umożliwia wczytanie wcześniej zapisanego obiektu sieci neuronowej, która na przykład mogła być uczona wcześniej i zapisana w celu umożliwiania dalszych testów lub odpytywania sieci. Aby wczytać plik do aplikacji należy z zakładki Plik wybieramy opcję Otwórz sieć ESN. Po kliknięciu zostaniemy poproszeni o wybranie pliku jaki chcemy wczytać. Filtr gwarantuje wczytanie pliku z odpowiednim rozszerzeniem.

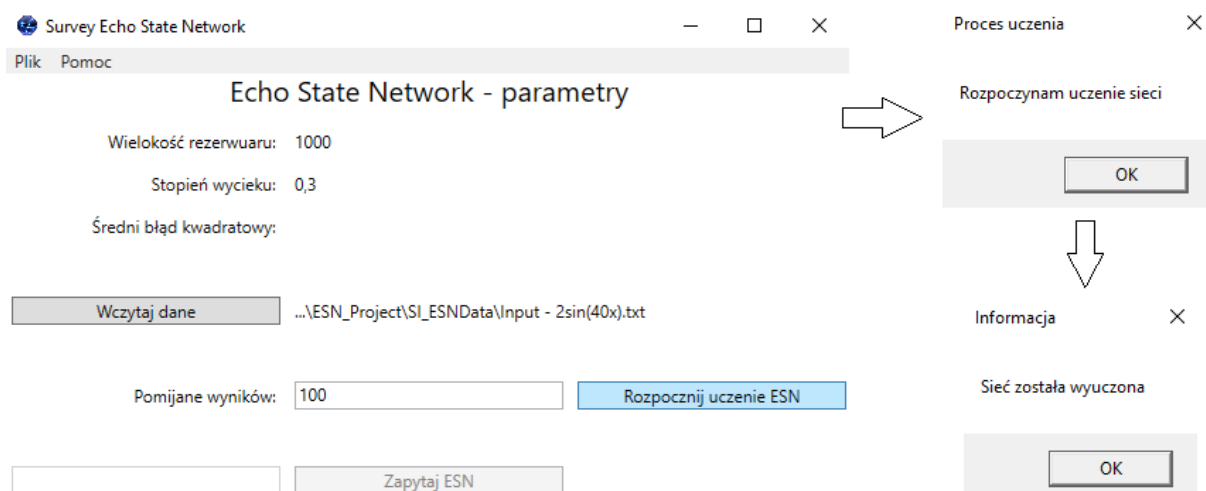
5. Wczytywanie danych wejściowych



Rysunek 5 Wczytywanie korpusu danych

W celu rozpoczęcia nauczania wygenerowanej sieci należy wczytać korpus danych. Aby to zrobić należy wybrać przycisk Wczytaj dane i wskazać plik tekstowy, którego format jest zgodny w przedstawionych założeniach projektu.

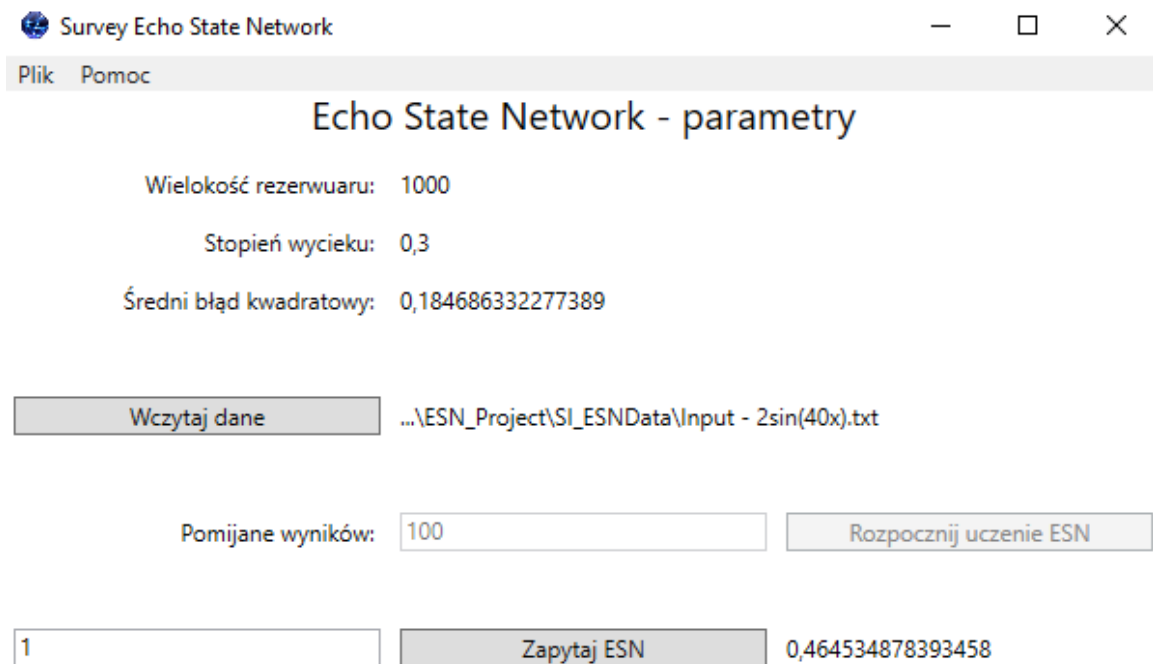
6. Uczenie sieci



Rysunek 6 Nauczanie sieci

Po wygenerowaniu sieci neuronowej i wskazaniu lokalizacji korpusu danych odblokowana zostaje możliwość określenia ilości pomijanych wstępnych wyników, która domyślnie jest ustalona na 100 oraz przycisk Rozpoczęcie uczenia ESN. Po wciśnięciu tego przycisku nastąpi rozpoczęcie procesu nauczania obiektu sieci neuronowej ESN. Po zakończeniu procesu nauczania sieć jest gotowa do odpytania.

7. Odpytywanie sieci



The screenshot shows a software window titled "Survey Echo State Network". The main title bar includes a globe icon, the text "Survey Echo State Network", and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with "Plik" and "Pomoc". The main content area is titled "Echo State Network - parametry". It displays three parameters: "Wielokość rezerwuaru: 1000", "Stopień wycieku: 0,3", and "Średni błąd kwadratowy: 0,184686332277389". There is a button labeled "Wczytaj dane" followed by a text field containing the file path "...\\ESN_Project\\SI_ESNData\\Input - 2sin(40x).txt". Below this is a label "Pomijane wyników:" followed by a text field with the value "100" and a button labeled "Rozpocznij uczenie ESN". At the bottom, there is a text field with the value "1", a button labeled "Zapytaj ESN", and a text field displaying the output value "0,464534878393458".

Rysunek 7 Odpytywanie sieci

Po ukończeniu procesu nauczanie odblokowane zostanie przycisk Zapytaj ESN oraz pole do wprowadzania wartości wejściowej. Po wciśnięciu przycisku zapytaj ESN wartość zostanie wprowadzona do sieci neuronowej, a otrzymany wynik wyświetlony obok przycisku Zapytaj ESN. Wprowadzane w ten sposób dane nie uczestniczą w procesie nauczania sieci.

2. Testy

1. Nauczanie funkcji kwadratowej

1. Dane wejściowe

Funkcja	$y = 2x^2 - 16$
Zakres	$\langle -15, 15 \rangle$
Krok	0,001
Rozmiar danych	30001
Ilość pominiętych wyników	100

2. Błąd kwadratowy 0,9652807446806

3. Porównanie wybranych wyników

Porównanie wyników		
Wartość wejściowa	Wartość spodziewana	Wartość właściwa
-15	434	335,011649489774
-6,37	65,1538	153,039192126762
0,151	-15,954398	15,53717654724
4,6	26,32	-78,2745804424437
14,124	382,974752	-279,097950538079

4. Wnioski

W porównaniu z innymi testami funkcja kwadratowa uzyskała najmniejszy błąd kwadratowy, a dla niektórych wartości wejściowych wartości wyjścia była stosunkowo zbliżona i porównywalna. Możliwe, że dla większego korpusu danych dla nauczania można by znacznie poprawić otrzymywane wartości.

2. Nauczanie funkcji sinusoidalnej

1. Dane wejściowe

Funkcja	$y = 2\sin(40x)$
Zakres	$\langle -15, 15 \rangle$
Krok	0,001
Rozmiar danych	30001
Ilość pominiętych wyników	100

2. Błąd kwadratowy 1,00681586772796

3. Porównanie wyników

Porównanie wyników		
Wartość wejściowa	Wartość spodziewana	Wartość właściwa
-14,987	-1,06947459987825	-1642,07941740433
-5,369	-1,81027203005697	-909,83917413703
0	0	-501,0849772003975
11,75	-1,89085102453865	393,469281310351
14,999	0,168194781682115	640,823050324228

4. Wnioski

Pomimo niskiego błędu kwadratowego wyniki znacznie odbiegają od spodziewanych wartości. Może być to spowodowane stosunkowo małym korpusem danych. Sieć ta może także znaczenie lepiej radzić sobie z odzwierciedlaniem funkcji innego typu.

VI. Tekst programu

1. Klasa EchoStateNetwork

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using LibraryESN;
using MathNet.Numerics.LinearAlgebra;

namespace SurveyESN
{
    [Serializable]
    public class EchoStateNetwork
    {
        // Status
        public bool isTeached; // flaga, która mówi o tym czy
        sieć jest już nauczona

        // Parametry sieci
        public double a; // Stopień wycieku
        public int size; // Wielkość rezerwuaru

        // Zmienne pośrednie i pomocnicze
        public Matrix<double> Win; // Warstwa wejścia
        public Matrix<double> W; // Warstwa rezerwuaru
        public Matrix<double> Wout; // Powłoka wyjścia
        public Matrix<double> X; // Macierz zerowa
        public Matrix<double> x;
        public Matrix<double> Yt; // Oczekiwane wartości
        public Matrix<double> Y; // Macierz wyjścia
        public double rhoW; //
        public Data data; //dane wejściowe
        public double mse; //błąd średniokwadratowy

        //Konstruktor
        public EchoStateNetwork(int reservoirSize, double leak-
        ingRate)
        {
            isTeached = false;
            InitializeParameters(reservoirSize, leakingRate); //
            <- a
            NormalizeWeights(); // Zapewnij rozrzut wag
        }

        // Przypisanie parametrów startowych
        private void InitializeParameters(int resSize, double al-
        pha)
        {
            size = resSize;
            a = alpha;
        }
    }
}
```

```

// Normalizowanie macierzy wag neuronów
private void NormalizeWeights()
{
    Win = Matrix<double>.Build.Random(size, 2) - 0.5;
    W = Matrix<double>.Build.Random(size, size) - 0.5;

    rhoW = W.Evd().EigenValues.AbsoluteMaximum().Real;
//rhoW = max(abs(eigenvalues(W)))

    W *= 1.25 / rhoW; // Normalizacja
}

// Ładowanie danych
public void Learn(string path, int ignoredInitialResults)
{
    // Inicjacja zmiennych
    data = new Data(path);
    X = Matrix<double>.Build.Random(data.DataLenght - ignoredInitialResults, 2 + size) * 0; // 2 <- ilość parametrów wejścia i wyjścia 'x' i 'y'

    Yt = Matrix<double>.Build.Random(1, data.DataLenght - ignoredInitialResults); // data.DataLenght - 1 - ignoredInitialResults, 1 Macierz wartości spodziewanych
    Yt.SetRow(0, data.GetExpetedOutputArray(ignoredInitialResults, data.DataLenght-ignoredInitialResults)); //Ustawienie pomijanych wyników

    InputDataIntoReservoir(ignoredInitialResults); //
    Przejsćie pierszych pomijanych wyników

    double reg = (double)1; // Doładność double

    Matrix<double> X_T = X.Transpose();

    Wout = Yt * X * Matrix<double>.Build.DenseOfMatrix(X_T * X + reg * Matrix<double>.Build.DenseDiagonal(2 + size, 2 + size, 1)).Inverse();

    //Uczenie
    LearnProcess(ignoredInitialResults, ignoredInitialResults);
    isTeached = true;
}

public void InputDataIntoReservoir(int ignoredDataCount)
{
    x = Matrix<double>.Build.Random(size, 1) * 0;

    for (int t = 0; t < data.InputData.Count; t++)
    {
        double[] u = data.InputData[t].ToArray();
    }
}

```

```

        x = (1 - a) * x + a * Matrix<double>.Tanh(Win *
Matrix<double>.Build.Dense(2, 1, u) + W * x);

        if (t >= ignoredDataCount)
        {
            X.SetRow(t-ignoredDataCount, Vector<double>.Build.DenseOfArray(JoinArrays(u,x.Column(0).ToArray())));
        }
    }

    private void LearnProcess(int initResults,int ignoredDataCount)
    {
        int testLenght = data.DataLenght - initResults;
        Y = Matrix<double>.Build.Dense(testLenght, 1);
        int count = data.InputData.Count;

        for (int t = initResults; t < data.DataLenght; t++)
        {
            double[] u = data.InputData[t].ToArray();
            x = (1 - a) * x + a * Matrix<double>.Tanh(Win *
Matrix<double>.Build.Dense(2, 1, u) + W * x);

            Vector<double> temp;

            if (t >= ignoredDataCount)
            {
                temp = Vector<double>.Build.DenseOfArray(JoinArrays(u, x.Column(0).ToArray()));

                Matrix<double> temMat = Matrix<double>.Build.Dense(1, temp.Count);
                temMat.SetRow(0, temp);

                double y = (Wout * temMat.Transpose())[0, 0];
                Y.SetRow(t-ignoredDataCount, Vector<double>.Build.Dense(1, y));
            }

        }

        mse = 0; // Minimal square error
        for (int i = 0; i < testLenght; i++)
        {
            double temp = Math.Abs(Y[i, 0]) - Math.Abs(Yt[0, i]);

            mse += Math.Sqrt(Math.Abs(temp));
        }
        mse /= testLenght;
    }
}

```

```

        public double Ask(double inputData)
        {
            double[] u = new double[2] { inputData, 0 };
            Matrix<double> x_local = (1 - a) * x + a * Matrix<double>.Tanh(Win * Matrix<double>.Build.Dense(2, 1, u) + W
            * x);

            Vector<double> temp = Vector<double>.Build.DenseOfArray(JoinArrays(u, x.Column(0).ToArray()));

            Matrix<double> temMat = Matrix<double>.Build.Dense(1, temp.Count);
            temMat.SetRow(0, temp);

            double y = (Wout * temMat.Transpose())[0, 0];

            return y;
        }

        public double[] JoinArrays(double[] a, double[] b)
        {
            double[] result = new double[a.Length + b.Length];

            for(int i = 0; i < a.Length; i++)
            {
                result[i] = a[i];
            }

            for (int i = 0; i < b.Length; i++)
            {
                result[a.Length + i] = b[i];
            }

            return result;
        }
    }
}

```

2. Klasa Data i Input

```

using System;
using MathNet.Numerics.LinearAlgebra;
using System.Collections.Generic;
using System.IO;

namespace LibraryESN
{
    /// <summary>
    /// Klasa zawiera listę danych wejściowych potrzebnych do
    /// procesu uczenia

```

```

    /// </summary>
    [Serializable]
    public class Data
    {
        public List<Input> InputData; //lista danych wejściowych
        public int DataLenght; //wielkość korpusu

        public Data(string filePath) //konstruktor przyjmuje w
        argumencie ścieżkę do pliku(korpusu)
        {
            InputData = new List<Input>();

            using (StreamReader sr = File.OpenText(filePath))
            {
                string s;
                string[] record;
                while ((s = sr.ReadLine()) != null) //odczytywa-
                nie danych z pliku
                {
                    record = s.Split(';');
                    InputData.Add(new Input(double.Parse(rec-
                    ord[0]),double.Parse(record[1]))); //dodawanie danych do listy
                }
                sr.Close();
                DataLenght = InputData.Count;
            }
        }

        public double[] GetExpetedOutputArray(int start, int
        lenght)
        {
            double[] result = new double[lenght];

            for(int i = start; i < lenght; i++)
            {
                result[i] = InputData[i].y;
            }

            return result;
        }
    }
    /// <summary>
    /// Klasa reprezentuje pojedynczy element danych z korpusu.
    Zawiera wartość x i y funkcji.
    /// </summary>
    [Serializable]
    public class Input
    {
        public double x;
        public double y;

        public Input(double _x, double _y)
        {

```



```

        x = _x;
        y = _y;
    }

    public double[] ToArray()
    {
        double[] d = new double[2];
        d[0] = x;
        d[1] = y;
        return d;
    }
}

```

3. Klasa SurveyESN – Aplikacja okeinkowa

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using LibraryESN;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;
using System.Threading;
using System.Globalization;

namespace SurveyESN
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public EchoStateNetwork esn;

        // Status
        public bool fileSelected;
        public String filePath;

        public MainWindow()
        {

```

```

        fileSelected = false;
        InitializeComponent();
        CheckStatus();
    }

    public void CheckStatus()
    {
        // Check if data file is selected
        if (fileSelected || esn != null)
        {
            teach.IsEnabled = true;
            initValue.IsEnabled = true;
        }
        else
        {
            teach.IsEnabled = false;
            initValue.IsEnabled = false;
        }

        // Check if ESN was teached
        try
        {
            if (esn.isTeached == false)
            {
                askBox.IsEnabled = false;
                askButton.IsEnabled = false;
            }
            else
            {
                askBox.IsEnabled = true;
                askButton.IsEnabled = true;
            }
        }
        catch (Exception e)
        {
            askBox.IsEnabled = false;
            askButton.IsEnabled = false;
        }
    }

    #region Buttons

    // Utworzenie nowej sieci (okno z parametrami)
    private void File_New_Click(object sender, RoutedEventArgs e)
    {
        showMessageBox("Now will generate ESN" + Environment.NewLine + "You will be inform about end of process", "Start the process");
        File_New.IsEnabled = false;
        Thread th = new Thread(() => generateNewESN());
        th.Start();
    }

```

```

        // Otwarcie zapisanej wcześniej sieci
        private void File_Open_Click(object sender, RoutedEventArgs e)
        {
            Microsoft.Win32.OpenFileDialog dlg = new Microsoft.Win32.OpenFileDialog();
            dlg.FileName = "File"; // Default file name
            dlg.DefaultExt = ".esn"; // Default file extension
            dlg.Filter = "ESN file (.esn)|*.esn"; // Filter files by extension

            // Show open file dialog box
            Nullable<bool> result = dlg.ShowDialog();
            FileStream fs;
            // Process open file dialog box results
            if (result == true)
            {
                // Open document
                string filename = dlg.FileName;
                fs = new FileStream(filename, FileMode.Open);
                try
                {
                    BinaryFormatter formatter = new BinaryFormatter();

                    //Deserialize the hashtable from the file and
                    //assign the reference to the local variable.
                    esn = (EchoStateNetwork)formatter.Deserialize(fs);

                    mseValue.Text = esn.mse.ToString();
                    reservoirValue.Text = esn.size.ToString();
                    if (esn.mse == 0) { mseValue.Text = ""; }
                }
                else { mseValue.Text = esn.mse.ToString(); }
                leakValue.Text = esn.a.ToString();
                if (esn.isTeached == true)
                {
                    askBox.IsEnabled = true;
                    askButton.IsEnabled = true;
                    teach.IsEnabled = false;
                    initValue.IsEnabled = false;
                }
                else
                {
                    askBox.IsEnabled = false;
                    askButton.IsEnabled = false;
                    teach.IsEnabled = true;
                    initValue.IsEnabled = true;
                    loadData.IsEnabled = true;
                }
                if (loadDataPath.Text.Equals("Nie wybrano pliku"))
                {
                    teach.IsEnabled = false;
                    initValue.IsEnabled = false;
                }
            }
        }
    }
}

```

```

        }
    }
    catch (Exception ex)
    {
        showMessageBox("Failed to deserialize. Reason: " + ex.Message, "Error");
        throw;
    }
    fs.Close();
}

// Zapisanie obiektu sieci
private void File_Save_Click(object sender, RoutedEventArgs e)
{
    // Configure save file dialog box
    Microsoft.Win32.SaveFileDialog dlg = new Microsoft.Win32.SaveFileDialog();
    dlg.FileName = "ESN " + DateTime.Today.ToShortDateString(); // Default file name
    dlg.DefaultExt = ".esn"; // Default file extension
    dlg.Filter = "Esn file (.esn)|*.esn"; // Filter files by extension

    // Show save file dialog box
    Nullable<bool> result = dlg.ShowDialog();
    FileStream fs;
    // Process save file dialog box results
    if (result == true)
    {
        // Save document
        string filename = dlg.FileName;
        fs = new FileStream(filename, FileMode.Create);

        // Construct a BinaryFormatter and use it to serialize the data to the stream.
        BinaryFormatter formatter = new BinaryFormatter();
        try
        {
            EchoStateNetwork temp = esn;
            temp.data = null;
            temp.Y = null;
            temp.Yt = null;
            temp.X = null;
            formatter.Serialize(fs, temp);
        }
        catch (Exception ex)
        {
            showMessageBox("Failed to serialize. Reason: " + ex.Message, "Error");
            throw;
        }
    }
}

```

```

        fs.Close();
    }
}

// Zamknięcie programu
private void File_Exit_Click(object sender, RoutedEventArgs e)
{
    this.Close();
}

// Okno z nami
private void Authors_Click(object sender, RoutedEventArgs e)
{
    showMessageBox("Adam Matuszak" + Environment.NewLine
+ "Łukasz Knop" + Environment.NewLine + "Szymon Kaszuba", "Au-
tors");
}

// Okienko z linkami do dokumentacji/git
private void Document_Click(object sender, RoutedEventArgs e)
{
    System.Diagnostics.Process.Start("https://gi-
thub.com/Tetrach121/SI_ESN");
}

// Wczytanie danych do nauki sieci
private void loadData_Click(object sender, RoutedEventArgs e)
{
    // Wczytaj dane

    Microsoft.Win32.OpenFileDialog dlg = new Mi-
crosoft.Win32.OpenFileDialog();
    dlg.FileName = "File"; // Default file name
    dlg.DefaultExt = ".txt"; // Default file extension
    dlg.Filter = "Txt file (.txt)|*.txt"; // Filter files
by extension

    // Show open file dialog box
    Nullable<bool> result = dlg.ShowDialog();

    // Process open file dialog box results
    if (result == true)
    {
        // Open document
        filePath = dlg.FileName;
        loadDataPath.Text = FileNameShow(dlg.FileName); //
Wypisz ścieżkę w label loadDataPath

```

```

        fileSelected = true; // Jeśli nie ma wyjątków to
fileSelected = true
    }

    CheckStatus();
}

private string FileNameShow(string s)
{
    string[] sSplit = s.Split('\\');
    if(sSplit.Length > 5)
        return "...\\\" + sSplit[sSplit.Length - 4] + '\\\"
+ sSplit[sSplit.Length - 3] + sSplit[sSplit.Length - 2] + '\\\" +
sSplit[sSplit.Length - 1];
    if (sSplit.Length > 3)
        return "...\\\" + sSplit[sSplit.Length - 2] + '\\\"
+ sSplit[sSplit.Length - 1];
    return s;
}

// Wprowadź zapytanie
private void askButton_Click(object sender, RoutedEventArgs e)
{
    double input;

    //Try parsing in the current culture
    if (!double.TryParse(askBox.Text, System.Globaliza-
tion.NumberStyles.Any, CultureInfo.CurrentCulture, out input) &&
        //Then try in US english
        !double.TryParse(askBox.Text, System.Globaliza-
tion.NumberStyles.Any, CultureInfo.GetCultureInfo("en-US"), out
input) &&
        //Then in neutral language
        !double.TryParse(askBox.Text, System.Globaliza-
tion.NumberStyles.Any, CultureInfo.InvariantCulture, out input))
    {
        input = 0;
    }
    double ans = esn.Ask(input);
    answer.Text = ans.ToString();
}

// Nauczaj się wczytanymi danymi
private void teach_Click(object sender, RoutedEventArgs
e)
{
    esn.Learn(filePath, int.Parse(initValue.Text));
    mseValue.Text = esn.mse.ToString();
    MessageBox.Show(esn.mse.ToString());
    CheckStatus();
}
#endregion

```

```

        #region Logic

        public void generateNewESN()
        {
            esn = new EchoStateNetwork(1000, 0.3);

            reservoirValue.Dispatcher.Invoke(() => { reservoir-
Value.Text = "1000"; });
            leakValue.Dispatcher.Invoke(() => { leakValue.Text =
"0.3"; });
            File_New.Dispatcher.Invoke(() => { File_New.IsEnabled
= true; });

            showMessageBox("Complete generate ESN", "Process com-
pleted");
        }

        static public void showMessageBox(String msg, String ti-
tle = "Message Box")
        {
            MessageBox.Show(msg, title, MessageBoxButton.OK);
        }
        #endregion
    }
}

```

4. Plik interfejsu graficznego

```

<Window x:Class="SurveyESN.MainWindow"
        xmlns="http://schemas.mi-
crosoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expres-
sion/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-com-
patibility/2006"
        xmlns:local="clr-namespace:SurveyESN"
        mc:Ignorable="d"
        Title="Survey Echo State Network" Height="420"
Width="800"
        MinHeight="420" MinWidth="800" MaxHeight="420"
MaxWidth="800">
    <Grid x:Name="grid_Main">
        <Grid.RowDefinitions>
            <RowDefinition Height="20" MinHeight="20" MaxHe-
ight="20"/>
            <RowDefinition Height="*" />
            <RowDefinition Height="30" MinHeight="30" MaxHe-
ight="30"/>
        </Grid.RowDefinitions>
    </Grid>

```

```

        <Menu Grid.Row="0" x:Name="Menu" HorizontalAlign-
ment="Left" VerticalAlignment="Top" IsMainMenu="True"
Width="{Binding ElementName=grid_Main,Path=ActualWidth}" Dock-
Panel.Dock="Top">
            <MenuItem x:Name="File" Header="Plik">
                <MenuItem x:Name="File_New" Header="Nowa sieć
ESN" Click="File_New_Click"/>
                <MenuItem x:Name="File_Open" Header="Otwórz sieć
ESN" Click="File_Open_Click"/>
                <MenuItem x:Name="File_Save" Header="Zapisz sieć
ESN" Click="File_Save_Click"/>
                <Separator/>
                <MenuItem x:Name="File_Exit" Header="Zamknij pro-
gram" Click="File_Exit_Click"/>
            </MenuItem>
            <MenuItem x:Name="Help" Header="Pomoc">
                <MenuItem x:Name="Autors" Header="Autorzy"
Click="Autors_Click"/>
                <MenuItem x:Name="Document" Header="Dokumentacja"
Click="Document_Click"/>
            </MenuItem>
        </Menu>
        <Grid Grid.Row="1" x:Name="ESN" VerticalAlignment="Bot-
tom" >
            <Grid.RowDefinitions>
                <RowDefinition Height="40"/>
                <RowDefinition Height="30"/>
                <RowDefinition Height="30"/>
                <RowDefinition Height="30"/>
                <RowDefinition Height="30"/>
                <RowDefinition Height="30"/>
                <RowDefinition Height="30"/>
                <RowDefinition Height="30"/>
                <RowDefinition Height="30"/>
                <RowDefinition Height="30"/>
                <RowDefinition Height="30"/>
            </Grid.RowDefinitions>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="200"/>
                <ColumnDefinition Width="200"/>
                <ColumnDefinition Width="200"/>
                <ColumnDefinition Width="200"/>
            </Grid.ColumnDefinitions>
            <TextBlock Grid.Row="0" Grid.ColumnSpan="2" Verti-
calAlignment="Center" TextAlignment="Center" HorizontalAlign-
ment="Center" Text="Echo State Network - parametry" Width="300"
Height="30" FontSize="20" Grid.Column="1" Grid.RowSpan="1"/>

            <TextBlock VerticalAlignment="Center" Grid.Row="1"
Grid.Column="0" x:Name="reservoirLable" Text="Wielokość
rezerwuaru: " Height="16" Margin="5" HorizontalAlignment="Right"
/>

```



```

        <TextBlock VerticalAlignment="Center" Grid.Row="1"
Grid.Column="1" x:Name="reservoirValue" Grid.ColumnSpan="3"
Text="null" Height="16" Margin="5"/>

        <TextBlock VerticalAlignment="Center" Grid.Row="2"
Grid.Column="0" x:Name="leakLable" Text="Stopień wycieku: " Mar-
gin="5" HorizontalAlignment="Right"/>
        <TextBlock VerticalAlignment="Center" Grid.Row="2"
Grid.Column="1" x:Name="leakValue" Grid.ColumnSpan="3"
Text="null" Margin="5"/>

        <TextBlock VerticalAlignment="Center" Grid.Row="3"
Grid.Column="0" x:Name="mseLable" Text="Średni błąd kwadratowy:
" Margin="5" HorizontalAlignment="Right"/>
        <TextBlock VerticalAlignment="Center" Grid.Row="3"
Grid.Column="1" Grid.ColumnSpan="3" x:Name="mseValue"
Text="null" Margin="5"/>

        <Button Grid.Row="5" Grid.Column="0" Margin="5"
x:Name="loadData" Content="Wczytaj dane" Click="load-
Data_Click"/>
        <TextBlock Margin="5" Grid.Row="5" Grid.Column="1"
Grid.ColumnSpan="3" x:Name="loadDataPath" VerticalAlign-
ment="Center" Text="Nie wybrano pliku"/>

        <TextBlock Grid.Row="7" Grid.Column="0" Margin="5"
VerticalAlignment="Center" x:Name="initLable" Text="Pomijane
wyników: " HorizontalAlignment="Right"/>
        <TextBox Grid.Row="7" Grid.Column="1" Margin="5"
x:Name="initValue" Text="100" />
        <Button Grid.Row="7" Grid.Column="2" Margin="5"
x:Name="teach" Content="Rozpocznij uczenie ESN"
Click="teach_Click"/>

        <TextBox Grid.Row="9" Grid.Column="0" Margin="5"
x:Name="askBox" Text="" />
        <Button Grid.Row="9" Grid.Column="1" Margin="5"
x:Name="askButton" Content="Zapytaj ESN" Click="askBut-
ton_Click"/>
        <TextBlock Grid.Row="9" Grid.Column="2" Grid.Col-
umnSpan="2" Margin="5" VerticalAlignment="Center" x:Name="an-
swer" />

    </Grid>
    <Grid Grid.Row="3">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="300"/>
            <ColumnDefinition Width="490"/>
        </Grid.ColumnDefinitions>
    </Grid>
</Grid>
</Window>

```