МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ

Производственная практика   
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| старший преподаватель |  |  |  | С.А. Рогачев |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ПОЯСНИТЕЛЬНАЯ ЗАПИСКА |
|  |
| Алгоритм поворота изображения на произвольный угол |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4932 |  |  |  | А.И. Белов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2021

Оглавление

[Задание 3](#_Toc76937386)

[Теория 3](#_Toc76937387)

[Результаты работы 5](#_Toc76937388)

[Распознание арабских чисел. 5](#_Toc76937389)

[Распознание римских чисел. 6](#_Toc76937390)

[Листинг кода 7](#_Toc76937391)

[Скрипт Ms SQL 7](#_Toc76937392)

[Приложение API 10](#_Toc76937393)

[DataSet.cs 10](#_Toc76937394)

[Network.cs 10](#_Toc76937395)

[LiteSQLConnection.cs 11](#_Toc76937396)

[DataSetController.cs 11](#_Toc76937397)

[NetworkController.cs 12](#_Toc76937398)

[Program.cs 14](#_Toc76937399)

[Startup.cs 14](#_Toc76937400)

[Приложение NumberRecognition\_Practice2021 15](#_Toc76937401)

[Neuron.cs 15](#_Toc76937402)

[Layer.cs 16](#_Toc76937403)

[Perceptron.cs 16](#_Toc76937404)

[DataSet.cs 19](#_Toc76937405)

[Network.cs 19](#_Toc76937406)

[ArrayPoints.cs 20](#_Toc76937407)

[ImageProcessor.cs 20](#_Toc76937408)

[LiteWebClient.cs 23](#_Toc76937409)

[Program.cs 24](#_Toc76937410)

[NeuralNetworkControlForm.cs 24](#_Toc76937411)

[NeuralNetworkControlForm(разметка) 28](#_Toc76937412)

[Список литературы 40](#_Toc76937413)

# Задание

Произвести классификацию изображения с помощью нейронной сети.

# Теория

**Искусственная нейронная сеть (ИНС)** (англ. *Artificial neural network (ANN)*) — упрощенная модель биологической нейронной сети, представляющая собой совокупность искусственных нейронов, взаимодействующих между собой.

Для построения искусственной нейронной сети будем использовать ту же структуру. Как и биологическая нейронная сеть, искусственная состоит из нейронов, взаимодействующих между собой, однако представляет собой упрощенную модель. Так, например, искусственный нейрон, из которых состоит ИНС, имеет намного более простую структуру: у него есть несколько входов, на которых он принимает различные сигналы, преобразует их и передает другим нейронам. Другими словами, искусственный нейрон — это такая функция R*n*→R, которая преобразует несколько входных параметров в один выходной.



Как видно на рисунке, у нейрона есть *n* входов *xi*, у каждого из которого есть вес *wi*, на который умножается сигнал, проходящий по связи. После этого взвешенные сигналы *xi*⋅*wi* направляются в сумматор, который агрегирует все сигналы во взвешенную сумму. Эту сумму также называют *net*. Таким образом,

Просто так передавать взвешенную сумму *net* на выход достаточно бессмысленно нейрон должен ее как-то обработать и сформировать адекватный выходной сигнал. Для этих целей используют функцию активации, которая преобразует взвешенную сумму в какое-то число, которое и будет являться выходом нейрона. Функция активации обозначается *ϕ*(*net*). Таким образом, выходов искусственного нейрона является *ϕ*(*net*).

Для данной нейросети выбрана сигмоидальная функция активации

Так как поставленная задача достаточно простая, для ее реализации будет достаточно использовать перцептрон.

**Перцептрон** (англ. *Perceptron*) — простейший вид нейронных сетей. В основе лежит математическая модель восприятия информации мозгом, состоящая из сенсоров, ассоциативных и реагирующих элементов.

Задача обучения перцептрона — подобрать такие *w*0,*w*1,*w*2,…,*wn*, чтобы *sign*(*σ*(*w*0+*w*1⋅*x*1+*w*2⋅*x*2+…+*wn*⋅*xn*)) как можно чаще совпадал с *y*(*x*) — значением в обучающей выборке (здесь *σ* — функция активации). Для удобства, чтобы не тащить за собой свободный член *w*0, добавим в вектор *x* лишнюю «виртуальную размерность» и будем считать, что *x*=(1,*x*1,*x*2,…,*xn*). Тогда *w*0+*w*1⋅*x*1+*w*2⋅*x*2+…+*wn*⋅*xn* можно заменить на *wT*⋅*x*

Чтобы обучать эту функцию, сначала надо выбрать функцию ошибки, которую потом можно оптимизировать градиентным спуском. Число неверно классифицированных примеров не подходит на эту кандидатуру, потому что эта функция кусочно-гладкая, с массой разрывов: она будет принимать только целые значения и резко меняться при переходе от одного числа неверно классифицированных примеров к другому. Поэтому использовать будем другую функцию, так называемый критерий перцептрона: *EP*(*w*)=−∑*x*∈*My*(*x*)(*σ*(*wT*⋅*x*)), где *M* — множество примеров, которые перцептрон с весами *w* классифицирует неправильно.

Иначе говоря, мы минимизируем суммарное отклонение наших ответов от правильных, но только в неправильную сторону; верный ответ ничего не вносит в функцию ошибки. Умножение на *y*(*x*) здесь нужно для того, чтобы знак произведения всегда получался отрицательным: если правильный ответ *−1*, значит, перцептрон выдал положительное число (иначе бы ответ был верным), и наоборот. В результате у нас получилась кусочно-линейная функция, дифференцируемая почти везде, а этого вполне достаточно.

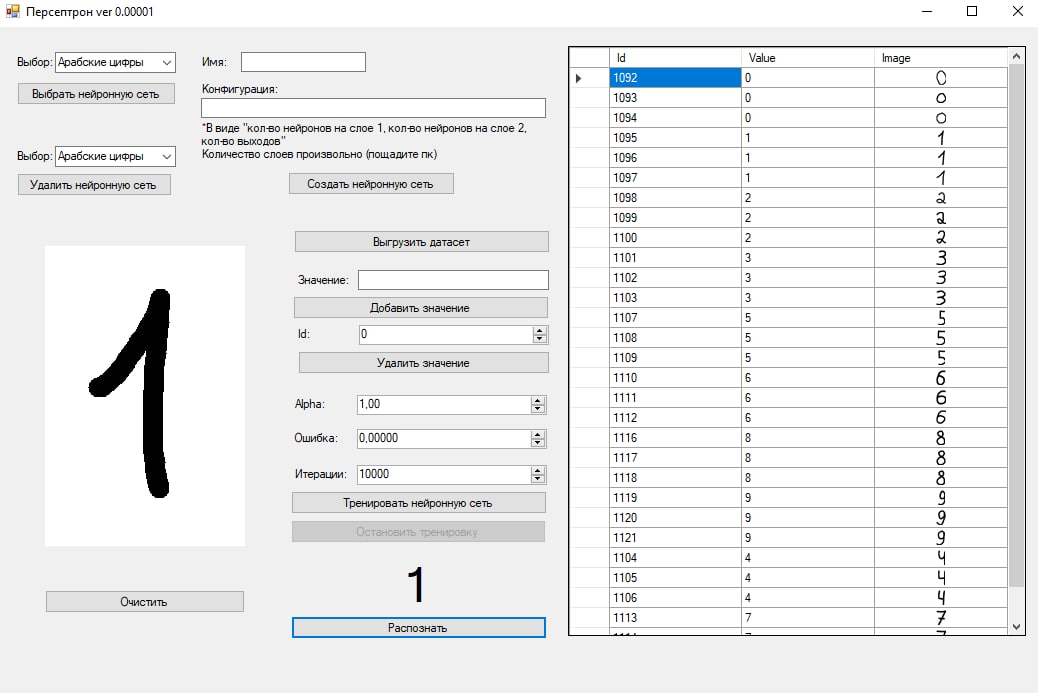
Теперь *EP*(*w*) можно оптимизировать градиентным спуском. На очередном шаге получаем: *w*(*τ*+1)=*w*(*τ*)−*η*▽*wEP*(*w*).

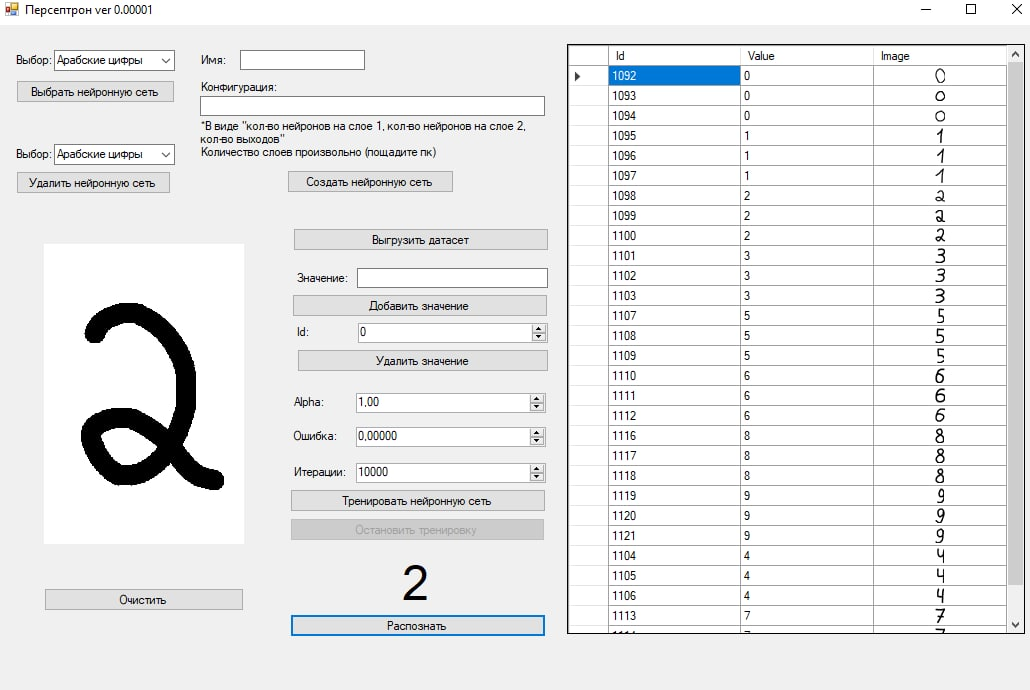
Алгоритм такой — мы последовательно проходим примеры *x*1,*x*2,…из обучающего множества, и для каждого *xn*

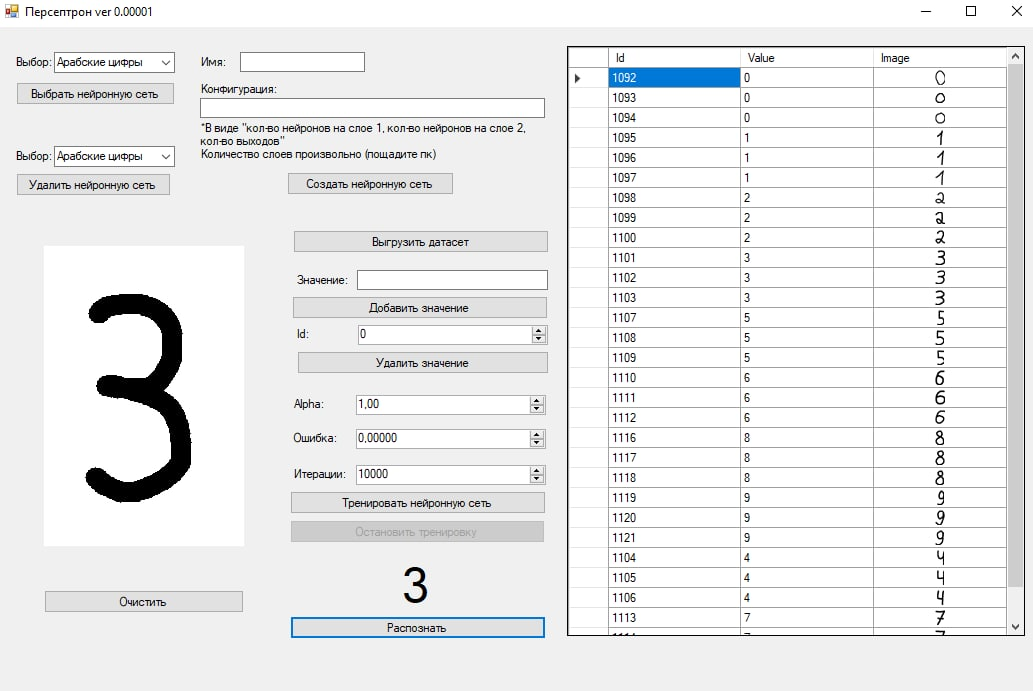
* если он классифицирован правильно, не меняем ничего;
* а если неправильно, прибавляем *η*▽*wEP*(*w*)

# Результаты работыЫ

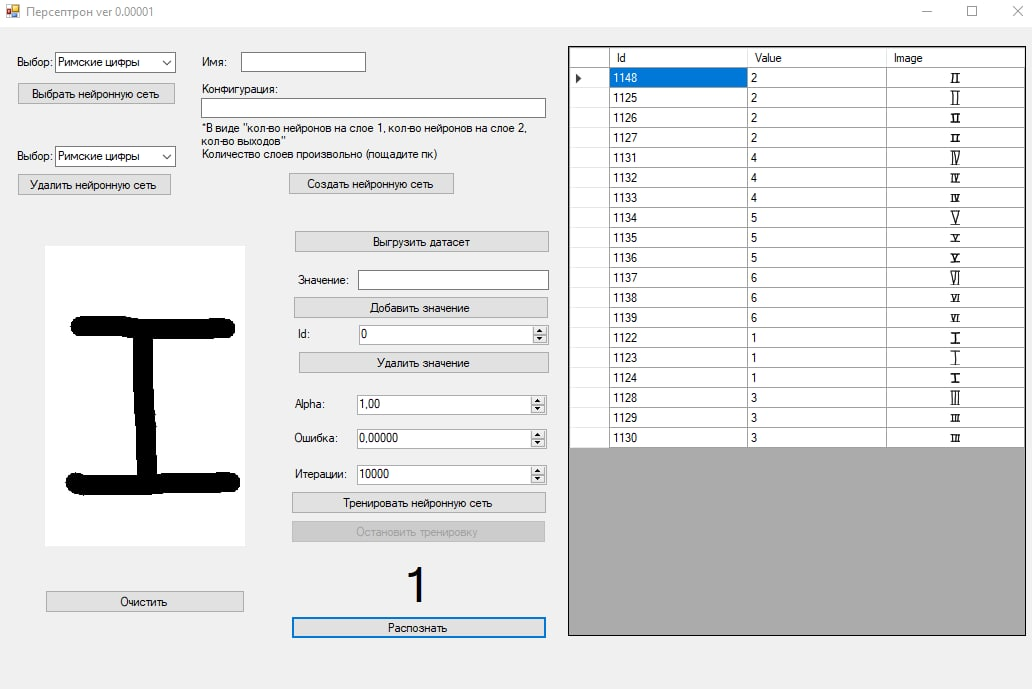
## Распознание арабских чисел.

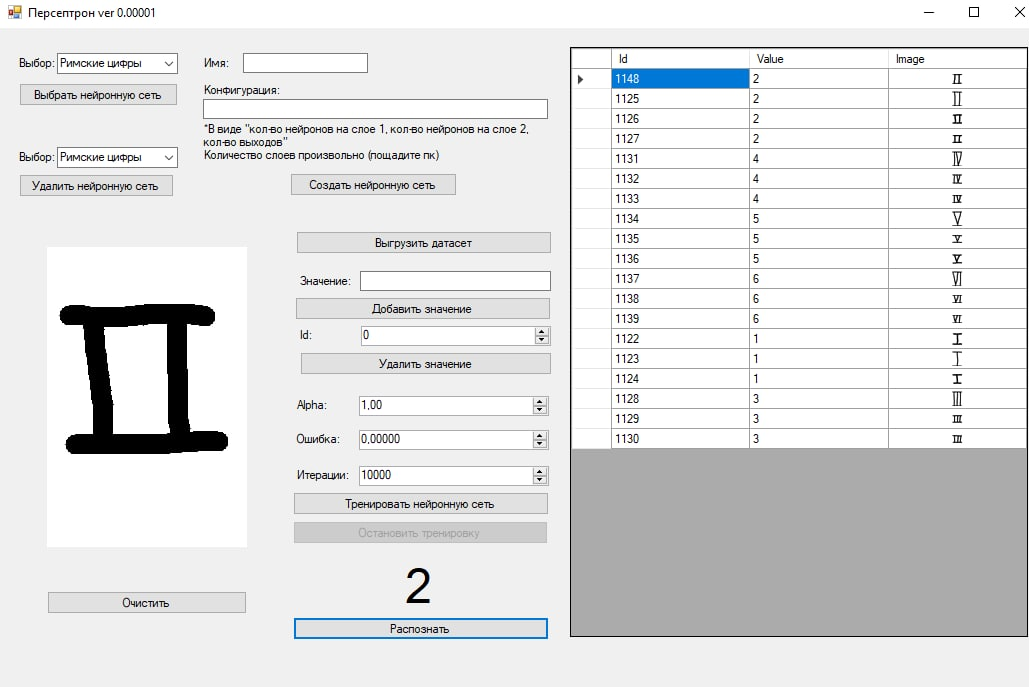






## Распознание римских чисел.





# Листинг кода

Репозиторий проекта: <https://github.com/Lichuhah/NumberRecognition_Practice2021>

## Скрипт Ms SQL

USE [master]

GO

/\*\*\*\*\*\* Object: Database [BDNeuralNetworks] Script Date: 01.07.2021 21:46:37 \*\*\*\*\*\*/

CREATE DATABASE [BDNeuralNetworks]

CONTAINMENT = NONE

ON PRIMARY

( NAME = N'BDNeuralNetworks', FILENAME = N'D:\Programms\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\DATA\BDNeuralNetworks.mdf' , SIZE = 8192KB , MAXSIZE = UNLIMITED, FILEGROWTH = 65536KB )

LOG ON

( NAME = N'BDNeuralNetworks\_log', FILENAME = N'D:\Programms\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\DATA\BDNeuralNetworks\_log.ldf' , SIZE = 8192KB , MAXSIZE = 2048GB , FILEGROWTH = 65536KB )

WITH CATALOG\_COLLATION = DATABASE\_DEFAULT

GO

ALTER DATABASE [BDNeuralNetworks] SET COMPATIBILITY\_LEVEL = 150

GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))

begin

EXEC [BDNeuralNetworks].[dbo].[sp\_fulltext\_database] @action = 'enable'

end

GO

ALTER DATABASE [BDNeuralNetworks] SET ANSI\_NULL\_DEFAULT OFF

GO

ALTER DATABASE [BDNeuralNetworks] SET ANSI\_NULLS OFF

GO

ALTER DATABASE [BDNeuralNetworks] SET ANSI\_PADDING OFF

GO

ALTER DATABASE [BDNeuralNetworks] SET ANSI\_WARNINGS OFF

GO

ALTER DATABASE [BDNeuralNetworks] SET ARITHABORT OFF

GO

ALTER DATABASE [BDNeuralNetworks] SET AUTO\_CLOSE OFF

GO

ALTER DATABASE [BDNeuralNetworks] SET AUTO\_SHRINK OFF

GO

ALTER DATABASE [BDNeuralNetworks] SET AUTO\_UPDATE\_STATISTICS ON

GO

ALTER DATABASE [BDNeuralNetworks] SET CURSOR\_CLOSE\_ON\_COMMIT OFF

GO

ALTER DATABASE [BDNeuralNetworks] SET CURSOR\_DEFAULT GLOBAL

GO

ALTER DATABASE [BDNeuralNetworks] SET CONCAT\_NULL\_YIELDS\_NULL OFF

GO

ALTER DATABASE [BDNeuralNetworks] SET NUMERIC\_ROUNDABORT OFF

GO

ALTER DATABASE [BDNeuralNetworks] SET QUOTED\_IDENTIFIER OFF

GO

ALTER DATABASE [BDNeuralNetworks] SET RECURSIVE\_TRIGGERS OFF

GO

ALTER DATABASE [BDNeuralNetworks] SET DISABLE\_BROKER

GO

ALTER DATABASE [BDNeuralNetworks] SET AUTO\_UPDATE\_STATISTICS\_ASYNC OFF

GO

ALTER DATABASE [BDNeuralNetworks] SET DATE\_CORRELATION\_OPTIMIZATION OFF

GO

ALTER DATABASE [BDNeuralNetworks] SET TRUSTWORTHY OFF

GO

ALTER DATABASE [BDNeuralNetworks] SET ALLOW\_SNAPSHOT\_ISOLATION OFF

GO

ALTER DATABASE [BDNeuralNetworks] SET PARAMETERIZATION SIMPLE

GO

ALTER DATABASE [BDNeuralNetworks] SET READ\_COMMITTED\_SNAPSHOT OFF

GO

ALTER DATABASE [BDNeuralNetworks] SET HONOR\_BROKER\_PRIORITY OFF

GO

ALTER DATABASE [BDNeuralNetworks] SET RECOVERY SIMPLE

GO

ALTER DATABASE [BDNeuralNetworks] SET MULTI\_USER

GO

ALTER DATABASE [BDNeuralNetworks] SET PAGE\_VERIFY CHECKSUM

GO

ALTER DATABASE [BDNeuralNetworks] SET DB\_CHAINING OFF

GO

ALTER DATABASE [BDNeuralNetworks] SET FILESTREAM( NON\_TRANSACTED\_ACCESS = OFF )

GO

ALTER DATABASE [BDNeuralNetworks] SET TARGET\_RECOVERY\_TIME = 60 SECONDS

GO

ALTER DATABASE [BDNeuralNetworks] SET DELAYED\_DURABILITY = DISABLED

GO

ALTER DATABASE [BDNeuralNetworks] SET ACCELERATED\_DATABASE\_RECOVERY = OFF

GO

ALTER DATABASE [BDNeuralNetworks] SET QUERY\_STORE = OFF

GO

USE [BDNeuralNetworks]

GO

/\*\*\*\*\*\* Object: Table [dbo].[DataSet] Script Date: 01.07.2021 21:46:38 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[DataSet](

[Id\_Network] [int] NOT NULL,

[Id\_Picture] [int] NOT NULL

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Network] Script Date: 01.07.2021 21:46:38 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Network](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](50) NOT NULL,

[Data] [varbinary](max) NOT NULL,

CONSTRAINT [PK\_Network] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Picture] Script Date: 01.07.2021 21:46:38 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Picture](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Value] [int] NOT NULL,

[Image] [varbinary](max) NOT NULL,

CONSTRAINT [PK\_Picture] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

ALTER TABLE [dbo].[DataSet] WITH CHECK ADD CONSTRAINT [FK\_DataSet\_Network] FOREIGN KEY([Id\_Network])

REFERENCES [dbo].[Network] ([Id])

GO

ALTER TABLE [dbo].[DataSet] CHECK CONSTRAINT [FK\_DataSet\_Network]

GO

ALTER TABLE [dbo].[DataSet] WITH CHECK ADD CONSTRAINT [FK\_DataSet\_Picture] FOREIGN KEY([Id\_Picture])

REFERENCES [dbo].[Picture] ([Id])

GO

ALTER TABLE [dbo].[DataSet] CHECK CONSTRAINT [FK\_DataSet\_Picture]

GO

USE [master]

GO

ALTER DATABASE [BDNeuralNetworks] SET READ\_WRITE

GO

## Приложение API

### DataSet.cs

namespace API.Models

{

public class Picture

{

public int Id { get; set; }

public int Value { get; set; }

public byte[] Image { get; set; }

}

public class DataSet

{

public int Id { get; set; }

public List<Picture> Pictures { get; set; }

public DataSet()

{

Pictures = new List<Picture>();

}

}

}

### Network.cs

namespace API.Models

{

public class Network

{

public int Id { get; set; }

public string Name { get; set; }

public byte[] Data { get; set; }

}

}

### LiteSQLConnection.cs

namespace API

{

static public class LiteSQLConnection

{

static public SqlConnection GetSQLConnection()

{

SqlConnection sqlConnection = new SqlConnection(@"Data Source=LAPTOP-R75QLBQV\SQL;Initial Catalog=BDNeuralNetworks;Integrated Security=True");

sqlConnection.Open();

return sqlConnection;

}

}

}

### DataSetController.cs

namespace API.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class DataSetController : ControllerBase

{

// GET api/<DataSetController>/5

[HttpGet("{id}")]

public DataSet Get(int id)

{

DataSet ds = new DataSet();

ds.Id = id;

SqlConnection sqlConnection = LiteSQLConnection.GetSQLConnection();

SqlCommand sqlCommand = new SqlCommand("SELECT \* FROM [dbo].[Picture] JOIN [dbo].[DataSet] ON Picture.Id = DataSet.Id\_Picture WHERE DataSet.Id\_Network=@id", sqlConnection);

sqlCommand.Parameters.AddWithValue("id", id);

SqlDataReader sqlDataReader = sqlCommand.ExecuteReader();

while (sqlDataReader.Read())

{

Picture pic = new Picture();

pic.Id = (int)sqlDataReader[0];

pic.Value = (int)sqlDataReader[1];

pic.Image = (byte[])sqlDataReader[2];

ds.Pictures.Add(pic);

}

sqlConnection.Close();

return ds;

}

// POST api/<DataSetController>/5

[HttpPost("{id}")]

public void Post(int id, [FromBody] Picture picture)

{

SqlConnection sqlConnection = LiteSQLConnection.GetSQLConnection();

SqlCommand sqlCommand = new SqlCommand("INSERT INTO [dbo].[Picture] VALUES (@Value, @Image)", sqlConnection);

sqlCommand.Parameters.AddWithValue("Value", picture.Value);

sqlCommand.Parameters.AddWithValue("Image", picture.Image);

sqlCommand.ExecuteNonQuery();

sqlCommand.CommandText = "SELECT @@IDENTITY";

int lastId = Convert.ToInt32(sqlCommand.ExecuteScalar());

sqlCommand = new SqlCommand("INSERT INTO [dbo].[DataSet] VALUES (@idNet, @idImg)", sqlConnection);

sqlCommand.Parameters.AddWithValue("idNet", id);

sqlCommand.Parameters.AddWithValue("idImg", lastId);

sqlCommand.ExecuteNonQuery();

sqlConnection.Close();

}

// DELETE api/<DataSetController>/5

[HttpDelete("{id}")]

public void Delete(int id)

{

SqlConnection sqlConnection = LiteSQLConnection.GetSQLConnection();

//удаляем датасет

SqlCommand sqlCommand = new SqlCommand("DELETE FROM DataSet WHERE Id\_Picture=@id", sqlConnection);

sqlCommand.Parameters.AddWithValue("id", id);

sqlCommand.ExecuteNonQuery();

//удаляем изображения из датасета

sqlCommand.CommandText = "DELETE FROM Picture WHERE Id=@id";

sqlCommand.ExecuteNonQuery();

sqlConnection.Close();

}

}

}

### NetworkController.cs

namespace API.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class NetworkController : ControllerBase

{

// GET: api/<NetworkController>

[HttpGet]

public List<Network> GetNetworks()

{

List<Network> networks = new List<Network>();

SqlConnection sqlConnection = LiteSQLConnection.GetSQLConnection();

SqlCommand sqlCommand = new SqlCommand("SELECT \* FROM [dbo].[Network]", sqlConnection);

SqlDataReader sqlDataReader = sqlCommand.ExecuteReader();

while (sqlDataReader.Read())

{

Network network = new Network();

network.Id = (int)sqlDataReader[0];

network.Name = sqlDataReader[1].ToString();

network.Data = (byte[])sqlDataReader[2];

networks.Add(network);

}

sqlConnection.Close();

return networks;

}

// GET api/<NetworkController>/GetNames

[HttpGet]

[Route("GetNames")]

public List<Network> GetNames()

{

List<Network> networksNames = new List<Network>();

SqlConnection sqlConnection = LiteSQLConnection.GetSQLConnection();

SqlCommand sqlCommand = new SqlCommand("SELECT Id,Name FROM [dbo].[Network]", sqlConnection);

SqlDataReader sqlDataReader = sqlCommand.ExecuteReader();

while (sqlDataReader.Read())

{

Network n = new Network();

n.Id = (int)sqlDataReader[0];

n.Name = sqlDataReader[1].ToString();

n.Data = null;

networksNames.Add(n);

}

sqlConnection.Close();

return networksNames;

}

// GET api/<NetworkController>/5

[HttpGet("{id}")]

public Network Get(int id)

{

SqlConnection sqlConnection = LiteSQLConnection.GetSQLConnection();

SqlCommand sqlCommand = new SqlCommand("SELECT \* FROM [dbo].[Network] WHERE Id=@id", sqlConnection);

sqlCommand.Parameters.AddWithValue("id", id);

SqlDataReader sqlDataReader = sqlCommand.ExecuteReader();

sqlDataReader.Read();

Network network = new Network();

network.Id = (int)sqlDataReader[0];

network.Name = sqlDataReader[1].ToString();

network.Data = (byte[])sqlDataReader[2];

sqlConnection.Close();

return network;

}

// POST api/<NetworkController>

[HttpPost]

public void Post([FromBody] Network network)

{

SqlConnection sqlConnection = LiteSQLConnection.GetSQLConnection();

SqlCommand sqlCommand = new SqlCommand("INSERT INTO [dbo].[Network] VALUES (@name, @data)", sqlConnection);

sqlCommand.Parameters.AddWithValue("name", network.Name);

sqlCommand.Parameters.AddWithValue("data", network.Data);

sqlCommand.ExecuteNonQuery();

sqlConnection.Close();

}

// PUT api/<NetworkController>/5

[HttpPut("{id}")]

public void Put(int id, [FromBody] Network network)

{

SqlConnection sqlConnection = LiteSQLConnection.GetSQLConnection();

SqlCommand sqlCommand = new SqlCommand("UPDATE [dbo].[Network] SET Data=@data WHERE Id=@id", sqlConnection);

sqlCommand.Parameters.AddWithValue("id", network.Id);

sqlCommand.Parameters.AddWithValue("data", network.Data);

sqlCommand.ExecuteNonQuery();

sqlConnection.Close();

}

// DELETE api/<NetworkController>/5

[HttpDelete("{id}")]

public void Delete(int id)

{

SqlConnection sqlConnection = LiteSQLConnection.GetSQLConnection();

//считываем айди изображений связанных с данной сетью

List<int> idPictures = new List<int>();

SqlCommand sqlCommand = new SqlCommand("SELECT Id\_Picture FROM DataSet WHERE Id\_Network=@id",sqlConnection);

sqlCommand.Parameters.AddWithValue("id", id);

SqlDataReader sqlDataReader = sqlCommand.ExecuteReader();

while (sqlDataReader.Read())

{

idPictures.Add((int)sqlDataReader[0]);

}

sqlDataReader.Close();

//удаляем датасет

sqlCommand.CommandText = "DELETE FROM DataSet WHERE Id\_Network=@id";

sqlCommand.ExecuteNonQuery();

//удаляем сеть

sqlCommand.CommandText = "DELETE FROM Network WHERE Id=@id";

sqlCommand.ExecuteNonQuery();

//удаляем изображения из датасета

sqlCommand.CommandText = "DELETE FROM Picture WHERE Id=@id";

for(int i=0; i<idPictures.Count(); i++)

{

sqlCommand.Parameters.Clear();

sqlCommand.Parameters.AddWithValue("id", idPictures[i]);

sqlCommand.ExecuteNonQuery();

}

sqlConnection.Close();

}

}

}

### Program.cs

namespace API

{

public class Program

{

public static void Main(string[] args)

{

CreateHostBuilder(args).Build().Run();

}

public static IHostBuilder CreateHostBuilder(string[] args) =>

Host.CreateDefaultBuilder(args)

.ConfigureWebHostDefaults(webBuilder =>

{

webBuilder.UseStartup<Startup>();

});

}

}

### Startup.cs

namespace API

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

services.AddControllers();

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.UseHttpsRedirection();

app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

});

}

}

}

## Приложение NumberRecognition\_Practice2021

### Neuron.cs

namespace NumberRecognition\_Practice2021

{

[Serializable]

class Neuron

{

public double[] weights; //веса

public double lastActivation; //последняя активация

public double bias; //сдвиг

public static double Sigmoid(double input)//сигмоида

{

return 1 / (1 + Math.Exp(-input));

}

public static double SigmoidDerivated(double input)//производная от сигмоиды

{

double y = Sigmoid(input);

return y \* (1 - y);

}

public Neuron(int numberOfInputs, Random r) //конструктор (кол-во входов, рандом)

{

bias = 5 \* r.NextDouble() - 2.5;

weights = new double[numberOfInputs];

for (int i = 0; i < numberOfInputs; i++)

{

weights[i] = 5\*r.NextDouble()-2.5;

}

}

public double Activate(double[] inputs) //активация массив входов

{

double activation = bias;

for (int i = 0; i < weights.Length; i++)

{

activation += weights[i] \* inputs[i];

}

lastActivation = activation;

return Sigmoid(activation);

}

}

}

### Layer.cs

namespace NumberRecognition\_Practice2021

{

[Serializable]

class Layer

{

public List<Neuron> neurons;

public int numberOfNeurons;

public double[] output;

public Layer(int \_numberOfNeurons, int numberOfInputs, Random r)

{

numberOfNeurons = \_numberOfNeurons;

neurons = new List<Neuron>();

for (int i = 0; i < numberOfNeurons; i++)

{

neurons.Add(new Neuron(numberOfInputs, r));

}

}

public double[] Activate(double[] inputs)

{

List<double> outputs = new List<double>();

for (int i = 0; i < numberOfNeurons; i++)

{

outputs.Add(neurons[i].Activate(inputs));

}

output = outputs.ToArray();

return outputs.ToArray();

}

}

}

### Perceptron.cs

namespace NumberRecognition\_Practice2021

{

[Serializable]

class Perceptron

{

List<Layer> layers;

public Perceptron(int[] neuronsPerlayer)

{

layers = new List<Layer>();

Random r = new Random();

for (int i = 0; i < neuronsPerlayer.Length; i++)

{

layers.Add(new Layer(neuronsPerlayer[i], i == 0 ? neuronsPerlayer[i] : neuronsPerlayer[i - 1], r));

}

}

public double[] Activate(double[] inputs)

{

double[] outputs = new double[0];

for (int i = 1; i < layers.Count; i++)

{

outputs = layers[i].Activate(inputs);

inputs = outputs;

}

return outputs;

}

double IndividualError(double[] realOutput, double[] desiredOutput)

{

double err = 0;

for (int i = 0; i < realOutput.Length; i++)

{

err += Math.Pow(realOutput[i] - desiredOutput[i], 2);

}

return err;

}

double GeneralError(List<double[]> input, List<double[]> desiredOutput)

{

double err = 0;

for (int i = 0; i < input.Count; i++)

{

err += IndividualError(Activate(input[i]), desiredOutput[i]);

}

return err;

}

List<double[]> sigmas;

List<double[,]> deltas;

void SetSigmas(double[] desiredOutput)

{

sigmas = new List<double[]>();

for (int i = 0; i < layers.Count; i++)

{

sigmas.Add(new double[layers[i].numberOfNeurons]);

}

for (int i = layers.Count - 1; i >= 0; i--)

{

for (int j = 0; j < layers[i].numberOfNeurons; j++)

{

if (i == layers.Count - 1)

{

double y = layers[i].neurons[j].lastActivation;

sigmas[i][j] = (Neuron.Sigmoid(y) - desiredOutput[j]) \* Neuron.SigmoidDerivated(y);

}

else

{

double sum = 0;

for (int k = 0; k < layers[i + 1].numberOfNeurons; k++)

{

sum += layers[i + 1].neurons[k].weights[j] \* sigmas[i + 1][k];

}

sigmas[i][j] = Neuron.SigmoidDerivated(layers[i].neurons[j].lastActivation) \* sum;

}

}

}

}

void SetDeltas()

{

deltas = new List<double[,]>();

for (int i = 0; i < layers.Count; i++)

{

deltas.Add(new double[layers[i].numberOfNeurons, layers[i].neurons[0].weights.Length]);

}

}

void AddDelta()

{

for (int i = 1; i < layers.Count; i++)

{

for (int j = 0; j < layers[i].numberOfNeurons; j++)

{

for (int k = 0; k < layers[i].neurons[j].weights.Length; k++)

{

deltas[i][j, k] += sigmas[i][j] \* Neuron.Sigmoid(layers[i - 1].neurons[k].lastActivation);

}

}

}

}

void UpdateBias(double alpha)

{

for (int i = 0; i < layers.Count; i++)

{

for (int j = 0; j < layers[i].numberOfNeurons; j++)

{

layers[i].neurons[j].bias -= alpha \* sigmas[i][j];

}

}

}

void UpdateWeights(double alpha)

{

for (int i = 0; i < layers.Count; i++)

{

for (int j = 0; j < layers[i].numberOfNeurons; j++)

{

for (int k = 0; k < layers[i].neurons[j].weights.Length; k++)

{

layers[i].neurons[j].weights[k] -= alpha \* deltas[i][j, k];

}

}

}

}

void ApplyBackPropagation(List<double[]> input, List<double[]> desiredOutput, double alpha)

{

SetDeltas();

for (int i = 0; i < input.Count; i++)

{

Activate(input[i]);

SetSigmas(desiredOutput[i]);

UpdateBias(alpha);

AddDelta();

}

UpdateWeights(alpha);

}

public bool Learn(ref object isNotStop, List<double[]> input, List<double[]> desiredOutput, double alpha, double maxError, int maxIterations, int iter\_save = 1)

{

alpha = 0.3;

double err = 99999;

int it = maxIterations;

while ((bool)isNotStop && err > maxError)

{

ApplyBackPropagation(input, desiredOutput, alpha);

err = GeneralError(input, desiredOutput);

if ((it - maxIterations) % 200 == 0)

{

Console.WriteLine(err + " iterations: " + (it - maxIterations));

}

maxIterations--;

if (maxIterations <= 0)

{

return false;

}

}

Console.WriteLine("complete");

return true;

}

public static Perceptron GetPerceptronFromByte (byte[] data)

{

Perceptron p;

using (MemoryStream ms = new MemoryStream(data))

{

var formatter = new BinaryFormatter();

ms.Seek(0, SeekOrigin.Begin);

p = (Perceptron)formatter.Deserialize(ms);

}

return p;

}

public static byte[] GetByteFromPerceptron(Perceptron p)

{

byte[] data;

BinaryFormatter formatter = new BinaryFormatter();

using (MemoryStream ms = new MemoryStream())

{

formatter.Serialize(ms, p);

data = ms.ToArray();

}

return data;

}

}

}

### DataSet.cs

public class Picture

{

public int Id { get; set; }

public int Value { get; set; }

public byte[] Image { get; set; }

}

public class DataSet

{

public int Id { get; set; }

public List<Picture> Pictures { get; set; }

}

### Network.cs

public class Network

{

public int Id { get; set; }

public string Name { get; set; }

public byte[] Data { get; set; }

}

### ArrayPoints.cs

namespace NumberRecognition\_Practice2021

{

class ArrayPoints

{

private int index = 0;

private Point[] points;

public ArrayPoints(int size)

{

if (size <= 0) { size = 2; };

points = new Point[size];

}

public void SetPoint(int x, int y)

{

if (index >= points.Length)

{

index = 0;

}

points[index] = new Point(x, y);

index++;

}

public void ResetPoints()

{

index = 0;

}

public int GetCountPoints()

{

return index;

}

public Point[] GetPoints()

{

return points;

}

}

}

### ImageProcessor.cs

namespace NumberRecognition\_Practice2021

{

static class ImageProcessor

{

static public Image ScaleImage(Image source, int width, int height)

{

Point[] bgges = FindImageBordes(source);

source = CropImage(source, bgges);

Image dest = new Bitmap(width, height);

using (Graphics gr = Graphics.FromImage(dest))

{

gr.FillRectangle(Brushes.White, 0, 0, width, height); // Очищаем экран

gr.InterpolationMode = System.Drawing.Drawing2D.InterpolationMode.HighQualityBicubic;

float srcwidth = source.Width;

float srcheight = source.Height;

float dstwidth = width;

float dstheight = height;

if (srcwidth <= dstwidth && srcheight <= dstheight) // Исходное изображение меньше целевого

{

int left = (width - source.Width) / 2;

int top = (height - source.Height) / 2;

gr.DrawImage(source, left, top, source.Width, source.Height);

}

else if (srcwidth / srcheight > dstwidth / dstheight) // Пропорции исходного изображения более широкие

{

float cy = srcheight / srcwidth \* dstwidth;

float top = ((float)dstheight - cy) / 2.0f;

if (top < 1.0f) top = 0;

gr.DrawImage(source, 0, top, dstwidth, cy);

}

else // Пропорции исходного изображения более узкие

{

float cx = srcwidth / srcheight \* dstheight;

float left = ((float)dstwidth - cx) / 2.0f;

if (left < 1.0f) left = 0;

gr.DrawImage(source, left, 0, cx, dstheight);

}

return dest;

}

}

static public Image CropImage(Image img, Point[] points)

{

Bitmap map = new Bitmap(img);

int x1 = points[0].X;

int y1 = points[0].Y;

int x2 = points[1].X;

int y2 = points[1].Y;

int width = x2 - x1 + 1;

int height = y2 - y1 + 1;

var result = new Bitmap(width, height);

for (int i = x1; i <= x2; i++)

for (int j = y1; j <= y2; j++)

result.SetPixel(i - x1, j - y1, map.GetPixel(i, j));

result.Save(@"C:\Users\belov\Desktop\NumberRecognition\_Practice2021\testCROP.jpg");

return result;

}

static public double GetDoubleFromColor(Color color)

{

double a = Convert.ToDouble(color.R + color.G + color.B) / 765;

return a;

}

static public double[] FromImageToInputs(Image pic)

{

pic.Save(@"C:\Users\belov\Desktop\NumberRecognition\_Practice2021\test.jpg");

Image outimg = pic;

if (pic.Width != 10 && pic.Height != 15)

{

outimg = ImageProcessor.ScaleImage(pic, 10, 15);

}

outimg.Save(@"C:\Users\belov\Desktop\NumberRecognition\_Practice2021\test2.jpg");

Bitmap outmap = new Bitmap(outimg);

double[] inputs = new double[150];

for (int i = 0; i < outmap.Height; i++)

{

for (int j = 0; j < outmap.Width; j++)

{

double a = GetDoubleFromColor(outmap.GetPixel(j, i));

inputs[i \* 10 + j] = a;

}

}

return inputs;

}

static public Point[] FindImageBordes(Image pic)

{

Bitmap img = new Bitmap(pic);

Point[] borders = new Point[2];

borders[0] = new Point(0, 0);

borders[1] = new Point(0, 0);

for (int i = 0; i < pic.Height; i++)

{

for (int j = 0; j < pic.Width; j++)

{

double temp = GetDoubleFromColor(img.GetPixel(j, i));

if (temp < 0.8)

{

borders[0].Y = i;

i = pic.Height;

j = pic.Width;

}

}

}

for (int i = pic.Height - 1; i > 0; i--)

{

for (int j = pic.Width - 1; j > 0; j--)

{

double temp =GetDoubleFromColor(img.GetPixel(j, i));

if (temp < 0.8)

{

borders[1].Y = i;

i = 0;

j = 0;

}

}

}

for (int i = 0; i < pic.Width; i++)

{

for (int j = 0; j < pic.Height; j++)

{

double temp = GetDoubleFromColor(img.GetPixel(i, j));

if (temp < 0.8)

{

borders[0].X = i;

i = pic.Width;

j = pic.Height;

}

}

}

for (int i = pic.Width - 1; i > 0; i--)

{

for (int j = pic.Height - 1; j > 0; j--)

{

double temp = GetDoubleFromColor(img.GetPixel(i, j));

if (temp < 0.8)

{

borders[1].X = i;

i = 0;

j = 0;

}

}

}

return borders;

}

static public byte[] GetByteFromImage(Image pic)

{

using (var ms = new MemoryStream())

{

pic.Save(ms, System.Drawing.Imaging.ImageFormat.Png);

return ms.ToArray();

}

}

static public Image GetImageFromByte(byte[] data)

{

using (var ms = new MemoryStream(data))

{

return Image.FromStream(ms);

}

}

}

}

### LiteWebClient.cs

namespace NumberRecognition\_Practice2021

{

static public class LiteWebClient

{

static public WebClient GetWebClientForJson()

{

var WebClient = new WebClient();

WebClient.Encoding = Encoding.UTF8;

WebClient.Headers[HttpRequestHeader.ContentType] = "application/json";

return WebClient;

}

static public void PostImage(int value, Image image, int networkId)

{

Picture picture = new Picture();

picture.Value = value;

picture.Image = ImageProcessor.GetByteFromImage(image);

string json = JsonConvert.SerializeObject(picture);

var WebClient = LiteWebClient.GetWebClientForJson();

string response = WebClient.UploadString("https://localhost:44387/api/DataSet/" + networkId, json);

}

static public void PostNetwork(string name, byte[] p)

{

Network network = new Network();

network.Name = name;

network.Data = p;

string json = JsonConvert.SerializeObject(network);

var WebClient = LiteWebClient.GetWebClientForJson();

string response = WebClient.UploadString("https://localhost:44387/api/Network", json);

}

static public Network GetPerceptron(Network net)

{

var WebClient = LiteWebClient.GetWebClientForJson();

string url = "https://localhost:44387/api/Network/" + net.Id;

string json = WebClient.DownloadString(url);

return JsonConvert.DeserializeObject<Network>(json);

}

}

}

### Program.cs

namespace NumberRecognition\_Practice2021

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

### NeuralNetworkControlForm.cs

namespace NumberRecognition\_Practice2021

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

SetSize();

int inputCount = 150;

int outputCount = 10;

int[] net\_def = new int[] { inputCount,24 ,outputCount };

p = new Perceptron(net\_def);

}

//рисование

private Bitmap map = new Bitmap(100, 100);

private Graphics graphics;

private Pen pen = new Pen(Color.Black, 20f);

private bool isMouse = false;

private ArrayPoints arrayPoints = new ArrayPoints(2);

private void SetSize()

{

Rectangle rectangle = Screen.PrimaryScreen.Bounds;

map = new Bitmap(pictureBox.Width, pictureBox.Height);

graphics = Graphics.FromImage(map);

graphics.Clear(pictureBox.BackColor);

pictureBox.Image = map;

pen.StartCap = System.Drawing.Drawing2D.LineCap.Round;

pen.EndCap = System.Drawing.Drawing2D.LineCap.Round;

}

private void pictureBox\_MouseDown(object sender, MouseEventArgs e)

{

isMouse = true;

if (e.Button == MouseButtons.Left)

{

pen.Color = Color.Black;

}

else

{

pen.Color = Color.White;

}

}

private void pictureBox\_MouseUp(object sender, MouseEventArgs e)

{

isMouse = false;

arrayPoints.ResetPoints();

}

private void pictureBox\_MouseMove(object sender, MouseEventArgs e)

{

if (!isMouse)

{

return;

}

arrayPoints.SetPoint(e.X, e.Y);

if (arrayPoints.GetCountPoints() >= 2)

{

graphics.DrawLines(pen, arrayPoints.GetPoints());

pictureBox.Image = map;

arrayPoints.SetPoint(e.X, e.Y);

}

}

private void btnClearPictureBox\_Click(object sender, EventArgs e)

{

graphics.Clear(pictureBox.BackColor);

pictureBox.Image = map;

}

private void btnAddData\_Click(object sender, EventArgs e)

{

double[] inputs = ImageProcessor.FromImageToInputs(pictureBox.Image);

double[] outputs = new double[] { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };

outputs[Convert.ToInt32(txtAddData.Text)] = 1;

input.Add(inputs);

output.Add(outputs);

Image img = ImageProcessor.ScaleImage(pictureBox.Image, 10, 15);

LiteWebClient.PostImage(Convert.ToInt32(txtAddData.Text), img, network.Id);

img.Save(@"C:\Users\belov\Desktop\NumberRecognition\_Practice2021\test3.jpg");

}

//работа с сетями

private Network network;

private Perceptron p;

private List<double[]> input = new List<double[]>();

private List<double[]> output = new List<double[]>();

private void btnCreateNewNetwork\_Click(object sender, EventArgs e)

{

try

{

String[] ss = textBox1.Text.Split(',');

int n = ss.Count();

int[] net\_def = new int[n + 1];

net\_def[0] = 150;

for (int i = 1; i < n + 1; i++)

{

net\_def[i] = int.Parse(ss[i - 1]);

}

p = new Perceptron(net\_def);

LiteWebClient.PostNetwork(txtNewNetworkName.Text, Perceptron.GetByteFromPerceptron(p));

}

catch

{

MessageBox.Show("Некорректная запись");

}

}

private void cmbNetworkSelection\_Click(object sender, EventArgs e)

{

var WebClient = LiteWebClient.GetWebClientForJson();

string json = WebClient.DownloadString("https://localhost:44387/api/Network/GetNames");

List<Network> things = JsonConvert.DeserializeObject<List<Network>>(json);

cmbNetworkSelection.DataSource = things;

cmbNetworkSelection.DisplayMember = "Name";

cmbNetworkDelete.DataSource = things;

cmbNetworkDelete.DisplayMember = "Name";

btnNetworkSelection.Enabled = true;

btnDeleteNetwork.Enabled = true;

}

private void btnNetworkSelection\_Click(object sender, EventArgs e)

{

network = LiteWebClient.GetPerceptron((Network)cmbNetworkSelection.SelectedItem);

p = Perceptron.GetPerceptronFromByte(network.Data);

btnCheckNumber.Enabled = true;

btnLoadDataSet.Enabled = true;

}

private void btnDeleteNetwork\_Click(object sender, EventArgs e)

{

var WebClient = LiteWebClient.GetWebClientForJson();

string json = WebClient.UploadString("https://localhost:44387/api/Network/" + ((Network)cmbNetworkDelete.SelectedItem).Id, "DELETE", "");

}

//работа с датасетами

private void btnLoadDataSet\_Click(object sender, EventArgs e)

{

var WebClient = LiteWebClient.GetWebClientForJson();

string json = WebClient.DownloadString("https://localhost:44387/api/DataSet/" + network.Id);

DataSet dataSets = JsonConvert.DeserializeObject<DataSet>(json);

dataGridView.DataSource = dataSets.Pictures;

input.Clear();

output.Clear();

for (int i = 0; i < dataSets.Pictures.Count; i++)

{

Image img = ImageProcessor.GetImageFromByte(dataSets.Pictures[i].Image);

double[] inputs = ImageProcessor.FromImageToInputs(img);

double[] outputs = new double[] { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };

outputs[dataSets.Pictures[i].Value] = 1;

input.Add(inputs);

output.Add(outputs);

}

btnAddData.Enabled = true;

btnDeleteData.Enabled = true;

btnTrainNetwork.Enabled = true;

}

private void btnDeleteData\_Click(object sender, EventArgs e)

{

var WebClient = LiteWebClient.GetWebClientForJson();

string json = WebClient.UploadString("https://localhost:44387/api/Dataset/" + numIdDeletePic.Value, "DELETE", "");

Console.WriteLine(json);

}

//тренировка и распознание чисел

private BackgroundWorker bw;

object isNotStop;

private void btnCheckNumber\_Click(object sender, EventArgs e)

{

double[] inputs = ImageProcessor.FromImageToInputs(pictureBox.Image);

double[] sal = p.Activate(inputs);

lblResult.Text = "";

int max = 0;

for (int i = 0; i < 10; i++)

{

if (sal[max] < sal[i])

{

max = i;

}

}

lblResult.Text = max.ToString();

}

private void btnTrainNetwork\_Click(object sender, EventArgs e)

{

btnTrainNetwork.Enabled = false;

btnStopTraining.Enabled = true;

int inputCount = 150;

int outputCount = 10;

int[] net\_def = new int[] { inputCount ,30,outputCount };

bw = new BackgroundWorker { WorkerSupportsCancellation = true };

bw.DoWork += (obj, args) =>

{

BackgroundWorker lbw = (BackgroundWorker)obj;

isNotStop = true;

while (!p.Learn(ref isNotStop, input, output, (double)numAlpha.Value, (double)numError.Value, (int)numIterations.Value, 10000))

{

p = new Perceptron(net\_def);

}

};

bw.RunWorkerAsync();

network.Data = Perceptron.GetByteFromPerceptron(p);

string json = JsonConvert.SerializeObject(network);

var WebClient = LiteWebClient.GetWebClientForJson();

string response = WebClient.UploadString("https://localhost:44387/api/Network/" + ((Network)cmbNetworkDelete.SelectedItem).Id, "PUT", json);

}

private void btnStopTraining\_Click(object sender, EventArgs e)

{

if (bw != null && bw.IsBusy && !bw.CancellationPending)

{

bw.CancelAsync();

isNotStop = false;

btnStopTraining.Enabled = false;

btnTrainNetwork.Enabled = true;

}

}

}

}

### NeuralNetworkControlForm(разметка)

namespace NumberRecognition\_Practice2021

{

partial class Form1

{

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

/// <param name="disposing">истинно, если управляемый ресурс должен быть удален; иначе ложно.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Код, автоматически созданный конструктором форм Windows

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

private void InitializeComponent()

{

this.pictureBox = new System.Windows.Forms.PictureBox();

this.btnClearPictureBox = new System.Windows.Forms.Button();

this.panel1 = new System.Windows.Forms.Panel();

this.panel2 = new System.Windows.Forms.Panel();

this.btnLoadDataSet = new System.Windows.Forms.Button();

this.btnDeleteData = new System.Windows.Forms.Button();

this.numIdDeletePic = new System.Windows.Forms.NumericUpDown();

this.label6 = new System.Windows.Forms.Label();

this.btnAddData = new System.Windows.Forms.Button();

this.txtAddData = new System.Windows.Forms.TextBox();

this.label2 = new System.Windows.Forms.Label();

this.panel3 = new System.Windows.Forms.Panel();

this.btnNetworkSelection = new System.Windows.Forms.Button();

this.cmbNetworkSelection = new System.Windows.Forms.ComboBox();

this.label3 = new System.Windows.Forms.Label();

this.panel4 = new System.Windows.Forms.Panel();

this.dataGridView = new System.Windows.Forms.DataGridView();

this.panel5 = new System.Windows.Forms.Panel();

this.label11 = new System.Windows.Forms.Label();

this.textBox1 = new System.Windows.Forms.TextBox();

this.label1 = new System.Windows.Forms.Label();

this.btnCreateNewNetwork = new System.Windows.Forms.Button();

this.txtNewNetworkName = new System.Windows.Forms.TextBox();

this.label4 = new System.Windows.Forms.Label();

this.panel6 = new System.Windows.Forms.Panel();

this.cmbNetworkDelete = new System.Windows.Forms.ComboBox();

this.btnDeleteNetwork = new System.Windows.Forms.Button();

this.label5 = new System.Windows.Forms.Label();

this.btnTrainNetwork = new System.Windows.Forms.Button();

this.label7 = new System.Windows.Forms.Label();

this.label8 = new System.Windows.Forms.Label();

this.label9 = new System.Windows.Forms.Label();

this.numAlpha = new System.Windows.Forms.NumericUpDown();

this.numError = new System.Windows.Forms.NumericUpDown();

this.numIterations = new System.Windows.Forms.NumericUpDown();

this.panel7 = new System.Windows.Forms.Panel();

this.btnStopTraining = new System.Windows.Forms.Button();

this.panel8 = new System.Windows.Forms.Panel();

this.lblResult = new System.Windows.Forms.Label();

this.btnCheckNumber = new System.Windows.Forms.Button();

this.button1 = new System.Windows.Forms.Button();

((System.ComponentModel.ISupportInitialize)(this.pictureBox)).BeginInit();

this.panel1.SuspendLayout();

this.panel2.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.numIdDeletePic)).BeginInit();

this.panel3.SuspendLayout();

this.panel4.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.dataGridView)).BeginInit();

this.panel5.SuspendLayout();

this.panel6.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.numAlpha)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.numError)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.numIterations)).BeginInit();

this.panel7.SuspendLayout();

this.panel8.SuspendLayout();

this.SuspendLayout();

//

// pictureBox

//

this.pictureBox.BackColor = System.Drawing.Color.White;

this.pictureBox.Location = new System.Drawing.Point(35, 35);

this.pictureBox.Name = "pictureBox";

this.pictureBox.Size = new System.Drawing.Size(200, 300);

this.pictureBox.TabIndex = 0;

this.pictureBox.TabStop = false;

this.pictureBox.MouseDown += new System.Windows.Forms.MouseEventHandler(this.pictureBox\_MouseDown);

this.pictureBox.MouseMove += new System.Windows.Forms.MouseEventHandler(this.pictureBox\_MouseMove);

this.pictureBox.MouseUp += new System.Windows.Forms.MouseEventHandler(this.pictureBox\_MouseUp);

//

// btnClearPictureBox

//

this.btnClearPictureBox.Location = new System.Drawing.Point(35, 379);

this.btnClearPictureBox.Name = "btnClearPictureBox";

this.btnClearPictureBox.Size = new System.Drawing.Size(200, 23);

this.btnClearPictureBox.TabIndex = 3;

this.btnClearPictureBox.Text = "Очистить";

this.btnClearPictureBox.UseVisualStyleBackColor = true;

this.btnClearPictureBox.Click += new System.EventHandler(this.btnClearPictureBox\_Click);

//

// panel1

//

this.panel1.Controls.Add(this.pictureBox);

this.panel1.Controls.Add(this.btnClearPictureBox);

this.panel1.Location = new System.Drawing.Point(13, 184);

this.panel1.Name = "panel1";

this.panel1.Size = new System.Drawing.Size(272, 428);

this.panel1.TabIndex = 4;

//

// panel2

//

this.panel2.Controls.Add(this.btnLoadDataSet);

this.panel2.Controls.Add(this.btnDeleteData);

this.panel2.Controls.Add(this.numIdDeletePic);

this.panel2.Controls.Add(this.label6);

this.panel2.Controls.Add(this.btnAddData);

this.panel2.Controls.Add(this.txtAddData);

this.panel2.Controls.Add(this.label2);

this.panel2.Location = new System.Drawing.Point(294, 187);

this.panel2.Name = "panel2";

this.panel2.Size = new System.Drawing.Size(267, 163);

this.panel2.TabIndex = 5;

//

// btnLoadDataSet

//

this.btnLoadDataSet.Enabled = false;

this.btnLoadDataSet.Location = new System.Drawing.Point(3, 16);

this.btnLoadDataSet.Name = "btnLoadDataSet";

this.btnLoadDataSet.Size = new System.Drawing.Size(256, 23);

this.btnLoadDataSet.TabIndex = 20;

this.btnLoadDataSet.Text = "Выгрузить датасет";

this.btnLoadDataSet.UseVisualStyleBackColor = true;

this.btnLoadDataSet.Click += new System.EventHandler(this.btnLoadDataSet\_Click);

//

// btnDeleteData

//

this.btnDeleteData.Enabled = false;

this.btnDeleteData.Location = new System.Drawing.Point(7, 137);

this.btnDeleteData.Name = "btnDeleteData";

this.btnDeleteData.Size = new System.Drawing.Size(252, 23);

this.btnDeleteData.TabIndex = 5;

this.btnDeleteData.Text = "Удалить значение";

this.btnDeleteData.UseVisualStyleBackColor = true;

this.btnDeleteData.Click += new System.EventHandler(this.btnDeleteData\_Click);

//

// numIdDeletePic

//

this.numIdDeletePic.Location = new System.Drawing.Point(68, 111);

this.numIdDeletePic.Maximum = new decimal(new int[] {

100000,

0,

0,

0});

this.numIdDeletePic.Name = "numIdDeletePic";

this.numIdDeletePic.Size = new System.Drawing.Size(190, 20);

this.numIdDeletePic.TabIndex = 4;

//

// label6

//

this.label6.AutoSize = true;

this.label6.Location = new System.Drawing.Point(4, 113);

this.label6.Name = "label6";

this.label6.Size = new System.Drawing.Size(19, 13);

this.label6.TabIndex = 3;

this.label6.Text = "Id:";

//

// btnAddData

//

this.btnAddData.Enabled = false;

this.btnAddData.Location = new System.Drawing.Point(2, 82);

this.btnAddData.Name = "btnAddData";

this.btnAddData.Size = new System.Drawing.Size(256, 23);

this.btnAddData.TabIndex = 2;

this.btnAddData.Text = "Добавить значение";

this.btnAddData.UseVisualStyleBackColor = true;

this.btnAddData.Click += new System.EventHandler(this.btnAddData\_Click);

//

// txtAddData

//

this.txtAddData.Location = new System.Drawing.Point(67, 56);

this.txtAddData.Name = "txtAddData";

this.txtAddData.Size = new System.Drawing.Size(191, 20);

this.txtAddData.TabIndex = 1;

//

// label2

//

this.label2.AutoSize = true;

this.label2.Location = new System.Drawing.Point(4, 59);

this.label2.Name = "label2";

this.label2.Size = new System.Drawing.Size(58, 13);

this.label2.TabIndex = 0;

this.label2.Text = "Значение:";

//

// panel3

//

this.panel3.Controls.Add(this.btnNetworkSelection);

this.panel3.Controls.Add(this.cmbNetworkSelection);

this.panel3.Controls.Add(this.label3);

this.panel3.Location = new System.Drawing.Point(13, 15);

this.panel3.Name = "panel3";

this.panel3.Size = new System.Drawing.Size(178, 85);

this.panel3.TabIndex = 6;

//

// btnNetworkSelection

//

this.btnNetworkSelection.Enabled = false;

this.btnNetworkSelection.Location = new System.Drawing.Point(7, 40);

this.btnNetworkSelection.Name = "btnNetworkSelection";

this.btnNetworkSelection.Size = new System.Drawing.Size(159, 23);

this.btnNetworkSelection.TabIndex = 2;

this.btnNetworkSelection.Text = "Выбрать нейронную сеть";

this.btnNetworkSelection.UseVisualStyleBackColor = true;

this.btnNetworkSelection.Click += new System.EventHandler(this.btnNetworkSelection\_Click);

//

// cmbNetworkSelection

//

this.cmbNetworkSelection.FormattingEnabled = true;

this.cmbNetworkSelection.Location = new System.Drawing.Point(45, 10);

this.cmbNetworkSelection.Name = "cmbNetworkSelection";

this.cmbNetworkSelection.Size = new System.Drawing.Size(121, 21);

this.cmbNetworkSelection.TabIndex = 1;

this.cmbNetworkSelection.Click += new System.EventHandler(this.cmbNetworkSelection\_Click);

//

// label3

//

this.label3.AutoSize = true;

this.label3.Location = new System.Drawing.Point(4, 13);

this.label3.Name = "label3";

this.label3.Size = new System.Drawing.Size(43, 13);

this.label3.TabIndex = 0;

this.label3.Text = "Выбор:";

//

// panel4

//

this.panel4.Controls.Add(this.dataGridView);

this.panel4.Location = new System.Drawing.Point(564, 15);

this.panel4.Name = "panel4";

this.panel4.Size = new System.Drawing.Size(468, 597);

this.panel4.TabIndex = 7;

//

// dataGridView

//

this.dataGridView.AutoSizeColumnsMode = System.Windows.Forms.DataGridViewAutoSizeColumnsMode.Fill;

this.dataGridView.AutoSizeRowsMode = System.Windows.Forms.DataGridViewAutoSizeRowsMode.AllCells;

this.dataGridView.ColumnHeadersHeightSizeMode = System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize;

this.dataGridView.Location = new System.Drawing.Point(7, 4);

this.dataGridView.Name = "dataGridView";

this.dataGridView.Size = new System.Drawing.Size(458, 590);

this.dataGridView.TabIndex = 0;

//

// panel5

//

this.panel5.Controls.Add(this.label11);

this.panel5.Controls.Add(this.textBox1);

this.panel5.Controls.Add(this.label1);

this.panel5.Controls.Add(this.btnCreateNewNetwork);

this.panel5.Controls.Add(this.txtNewNetworkName);

this.panel5.Controls.Add(this.label4);

this.panel5.Location = new System.Drawing.Point(198, 15);

this.panel5.Name = "panel5";

this.panel5.Size = new System.Drawing.Size(360, 163);

this.panel5.TabIndex = 8;

//

// label11

//

this.label11.AutoSize = true;

this.label11.Location = new System.Drawing.Point(4, 79);

this.label11.Name = "label11";

this.label11.Size = new System.Drawing.Size(335, 39);

this.label11.TabIndex = 5;

this.label11.Text = "\*В виде \"кол-во нейронов на слое 1, кол-во нейронов на слое 2, \r\nкол-во выходов\"\r" +

"\nКоличество слоев произвольно (пощадите пк)";

//

// textBox1

//

this.textBox1.Location = new System.Drawing.Point(6, 56);

this.textBox1.Name = "textBox1";

this.textBox1.Size = new System.Drawing.Size(345, 20);

this.textBox1.TabIndex = 4;

//

// label1

//

this.label1.AutoSize = true;

this.label1.Location = new System.Drawing.Point(4, 40);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(83, 13);

this.label1.TabIndex = 3;

this.label1.Text = "Конфигурация:";

//

// btnCreateNewNetwork

//

this.btnCreateNewNetwork.Location = new System.Drawing.Point(93, 130);

this.btnCreateNewNetwork.Name = "btnCreateNewNetwork";

this.btnCreateNewNetwork.Size = new System.Drawing.Size(167, 23);

this.btnCreateNewNetwork.TabIndex = 2;

this.btnCreateNewNetwork.Text = "Создать нейронную сеть";

this.btnCreateNewNetwork.UseVisualStyleBackColor = true;

this.btnCreateNewNetwork.Click += new System.EventHandler(this.btnCreateNewNetwork\_Click);

//

// txtNewNetworkName

//

this.txtNewNetworkName.Location = new System.Drawing.Point(46, 10);

this.txtNewNetworkName.Name = "txtNewNetworkName";

this.txtNewNetworkName.Size = new System.Drawing.Size(125, 20);

this.txtNewNetworkName.TabIndex = 1;

//

// label4

//

this.label4.AutoSize = true;

this.label4.Location = new System.Drawing.Point(4, 13);

this.label4.Name = "label4";

this.label4.Size = new System.Drawing.Size(32, 13);

this.label4.TabIndex = 0;

this.label4.Text = "Имя:";

//

// panel6

//

this.panel6.Controls.Add(this.cmbNetworkDelete);

this.panel6.Controls.Add(this.btnDeleteNetwork);

this.panel6.Controls.Add(this.label5);

this.panel6.Location = new System.Drawing.Point(13, 94);

this.panel6.Name = "panel6";

this.panel6.Size = new System.Drawing.Size(178, 84);

this.panel6.TabIndex = 9;

//

// cmbNetworkDelete

//

this.cmbNetworkDelete.FormattingEnabled = true;

this.cmbNetworkDelete.Location = new System.Drawing.Point(45, 25);

this.cmbNetworkDelete.Name = "cmbNetworkDelete";

this.cmbNetworkDelete.Size = new System.Drawing.Size(121, 21);

this.cmbNetworkDelete.TabIndex = 3;

this.cmbNetworkDelete.Click += new System.EventHandler(this.cmbNetworkSelection\_Click);

//

// btnDeleteNetwork

//

this.btnDeleteNetwork.Enabled = false;

this.btnDeleteNetwork.Location = new System.Drawing.Point(7, 52);

this.btnDeleteNetwork.Name = "btnDeleteNetwork";

this.btnDeleteNetwork.Size = new System.Drawing.Size(155, 23);

this.btnDeleteNetwork.TabIndex = 2;

this.btnDeleteNetwork.Text = "Удалить нейронную сеть";

this.btnDeleteNetwork.UseVisualStyleBackColor = true;

this.btnDeleteNetwork.Click += new System.EventHandler(this.btnDeleteNetwork\_Click);

//

// label5

//

this.label5.AutoSize = true;

this.label5.Location = new System.Drawing.Point(4, 28);

this.label5.Name = "label5";

this.label5.Size = new System.Drawing.Size(43, 13);

this.label5.TabIndex = 0;

this.label5.Text = "Выбор:";

//

// btnTrainNetwork

//

this.btnTrainNetwork.Enabled = false;

this.btnTrainNetwork.Location = new System.Drawing.Point(3, 106);

this.btnTrainNetwork.Name = "btnTrainNetwork";

this.btnTrainNetwork.Size = new System.Drawing.Size(256, 23);

this.btnTrainNetwork.TabIndex = 6;

this.btnTrainNetwork.Text = "Тренировать нейронную сеть";

this.btnTrainNetwork.UseVisualStyleBackColor = true;

this.btnTrainNetwork.Click += new System.EventHandler(this.btnTrainNetwork\_Click);

//

// label7

//

this.label7.AutoSize = true;

this.label7.Location = new System.Drawing.Point(4, 12);

this.label7.Name = "label7";

this.label7.Size = new System.Drawing.Size(37, 13);

this.label7.TabIndex = 10;

this.label7.Text = "Alpha:";

//

// label8

//

this.label8.AutoSize = true;

this.label8.Location = new System.Drawing.Point(4, 46);

this.label8.Name = "label8";

this.label8.Size = new System.Drawing.Size(50, 13);

this.label8.TabIndex = 11;

this.label8.Text = "Ошибка:";

//

// label9

//

this.label9.AutoSize = true;

this.label9.Location = new System.Drawing.Point(4, 82);

this.label9.Name = "label9";

this.label9.Size = new System.Drawing.Size(59, 13);

this.label9.TabIndex = 12;

this.label9.Text = "Итерации:";

//

// numAlpha

//

this.numAlpha.DecimalPlaces = 2;

this.numAlpha.Increment = new decimal(new int[] {

100,

0,

0,

0});

this.numAlpha.Location = new System.Drawing.Point(69, 10);

this.numAlpha.Maximum = new decimal(new int[] {

1,

0,

0,

0});

this.numAlpha.Name = "numAlpha";

this.numAlpha.Size = new System.Drawing.Size(190, 20);

this.numAlpha.TabIndex = 15;

this.numAlpha.Value = new decimal(new int[] {

1,

0,

0,

0});

//

// numError

//

this.numError.DecimalPlaces = 5;

this.numError.Location = new System.Drawing.Point(69, 44);

this.numError.Name = "numError";

this.numError.Size = new System.Drawing.Size(190, 20);

this.numError.TabIndex = 16;

//

// numIterations

//

this.numIterations.Location = new System.Drawing.Point(69, 80);

this.numIterations.Maximum = new decimal(new int[] {

100000,

0,

0,

0});

this.numIterations.Name = "numIterations";

this.numIterations.Size = new System.Drawing.Size(190, 20);

this.numIterations.TabIndex = 17;

this.numIterations.Value = new decimal(new int[] {

10000,

0,

0,

0});

//

// panel7

//

this.panel7.Controls.Add(this.btnStopTraining);

this.panel7.Controls.Add(this.label7);

this.panel7.Controls.Add(this.numIterations);

this.panel7.Controls.Add(this.btnTrainNetwork);

this.panel7.Controls.Add(this.numAlpha);

this.panel7.Controls.Add(this.numError);

this.panel7.Controls.Add(this.label8);

this.panel7.Controls.Add(this.label9);

this.panel7.Location = new System.Drawing.Point(291, 358);

this.panel7.Name = "panel7";

this.panel7.Size = new System.Drawing.Size(267, 161);

this.panel7.TabIndex = 18;

//

// btnStopTraining

//

this.btnStopTraining.Enabled = false;

this.btnStopTraining.Location = new System.Drawing.Point(3, 135);

this.btnStopTraining.Name = "btnStopTraining";

this.btnStopTraining.Size = new System.Drawing.Size(255, 23);

this.btnStopTraining.TabIndex = 20;

this.btnStopTraining.Text = "Остановить тренировку";

this.btnStopTraining.UseVisualStyleBackColor = true;

this.btnStopTraining.Click += new System.EventHandler(this.btnStopTraining\_Click);

//

// panel8

//

this.panel8.Controls.Add(this.lblResult);

this.panel8.Controls.Add(this.btnCheckNumber);

this.panel8.Location = new System.Drawing.Point(291, 525);

this.panel8.Name = "panel8";

this.panel8.Size = new System.Drawing.Size(267, 87);

this.panel8.TabIndex = 19;

//

// lblResult

//

this.lblResult.AutoSize = true;

this.lblResult.Font = new System.Drawing.Font("Microsoft Sans Serif", 36F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.lblResult.Location = new System.Drawing.Point(105, 6);

this.lblResult.Name = "lblResult";

this.lblResult.Size = new System.Drawing.Size(51, 55);

this.lblResult.TabIndex = 2;

this.lblResult.Text = "0";

//

// btnCheckNumber

//

this.btnCheckNumber.Enabled = false;

this.btnCheckNumber.Location = new System.Drawing.Point(3, 64);

this.btnCheckNumber.Name = "btnCheckNumber";

this.btnCheckNumber.Size = new System.Drawing.Size(256, 23);

this.btnCheckNumber.TabIndex = 1;

this.btnCheckNumber.Text = "Распознать";

this.btnCheckNumber.UseVisualStyleBackColor = true;

this.btnCheckNumber.Click += new System.EventHandler(this.btnCheckNumber\_Click);

//

// Form1

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(1044, 669);

this.Controls.Add(this.button1);

this.Controls.Add(this.panel8);

this.Controls.Add(this.panel7);

this.Controls.Add(this.panel6);

this.Controls.Add(this.panel5);

this.Controls.Add(this.panel4);

this.Controls.Add(this.panel3);

this.Controls.Add(this.panel2);

this.Controls.Add(this.panel1);

this.Name = "Form1";

this.Text = "Персептрон ver 0.00001";

((System.ComponentModel.ISupportInitialize)(this.pictureBox)).EndInit();

this.panel1.ResumeLayout(false);

this.panel2.ResumeLayout(false);

this.panel2.PerformLayout();

((System.ComponentModel.ISupportInitialize)(this.numIdDeletePic)).EndInit();

this.panel3.ResumeLayout(false);

this.panel3.PerformLayout();

this.panel4.ResumeLayout(false);

((System.ComponentModel.ISupportInitialize)(this.dataGridView)).EndInit();

this.panel5.ResumeLayout(false);

this.panel5.PerformLayout();

this.panel6.ResumeLayout(false);

this.panel6.PerformLayout();

((System.ComponentModel.ISupportInitialize)(this.numAlpha)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.numError)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.numIterations)).EndInit();

this.panel7.ResumeLayout(false);

this.panel7.PerformLayout();

this.panel8.ResumeLayout(false);

this.panel8.PerformLayout();

this.ResumeLayout(false);

}

#endregion

private System.Windows.Forms.PictureBox pictureBox;

private System.Windows.Forms.Button btnClearPictureBox;

private System.Windows.Forms.Panel panel1;

private System.Windows.Forms.Panel panel2;

private System.Windows.Forms.Button btnAddData;

private System.Windows.Forms.TextBox txtAddData;

private System.Windows.Forms.Label label2;

private System.Windows.Forms.Panel panel3;

private System.Windows.Forms.Button btnNetworkSelection;

private System.Windows.Forms.ComboBox cmbNetworkSelection;

private System.Windows.Forms.Label label3;

private System.Windows.Forms.Panel panel4;

private System.Windows.Forms.DataGridView dataGridView;

private System.Windows.Forms.Panel panel5;

private System.Windows.Forms.Button btnCreateNewNetwork;

private System.Windows.Forms.TextBox txtNewNetworkName;

private System.Windows.Forms.Label label4;

private System.Windows.Forms.Button btnDeleteData;

private System.Windows.Forms.NumericUpDown numIdDeletePic;

private System.Windows.Forms.Label label6;

private System.Windows.Forms.Panel panel6;

private System.Windows.Forms.Button btnDeleteNetwork;

private System.Windows.Forms.Label label5;

private System.Windows.Forms.Button btnTrainNetwork;

private System.Windows.Forms.Label label7;

private System.Windows.Forms.Label label8;

private System.Windows.Forms.Label label9;

private System.Windows.Forms.NumericUpDown numAlpha;

private System.Windows.Forms.NumericUpDown numError;

private System.Windows.Forms.NumericUpDown numIterations;

private System.Windows.Forms.Panel panel7;

private System.Windows.Forms.Panel panel8;

private System.Windows.Forms.Label lblResult;

private System.Windows.Forms.Button btnCheckNumber;

private System.Windows.Forms.ComboBox cmbNetworkDelete;

private System.Windows.Forms.Button btnLoadDataSet;

private System.Windows.Forms.Label label11;

private System.Windows.Forms.TextBox textBox1;

private System.Windows.Forms.Label label1;

private System.Windows.Forms.Button btnStopTraining;

private System.Windows.Forms.Button button1;

}

}

# Список литературы

«Нейронные сети. Полный курс» - Саймон Хайкин

**«Глубокое обучение» - Гудфеллоу Я., Бенджио И., Курвилль А..**

«Грокаем глубокое обучение» - Траск Эндрю

Документация .Net <https://docs.microsoft.com/ru-ru/dotnet/> - Microsoft