Отчет по лабораторной работе №3 по предмету «Нейронные сети в задачах технического зрения и управления»

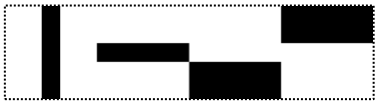
Тема: Применение свёрточных нейронных сетей.

Цель работы: Разработать алгоритм классификации стандартизованных изображений с помощью свёрточной нейронной сети.

Задачи:

- Подготовить приложение для генерации обучающей выборки из исходного набора изображений;
- Разработать функцию расчета свёрточной карты для выбранного ядра;
- Разработать класс, реализующий функциональность свёрточной сети для классификации изображений;
- Разработать функцию обучения с учителем для нейтронной сети по подготовленной ранее обучающей выборке.

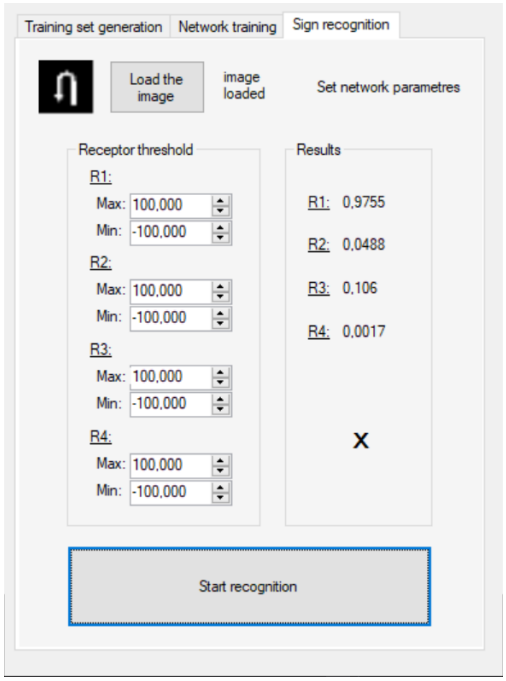
Опробованные наборы ядер:

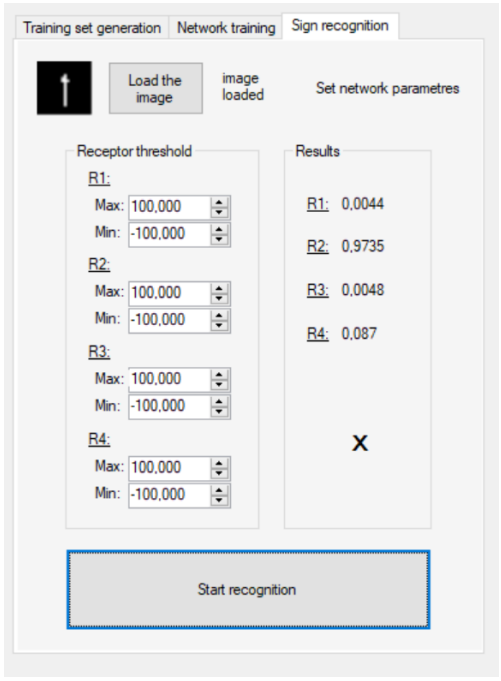


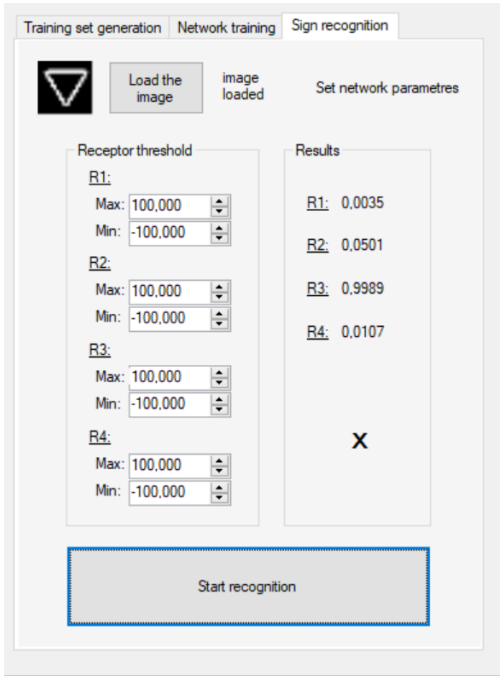
Ядра дающие наилучшие результаты:

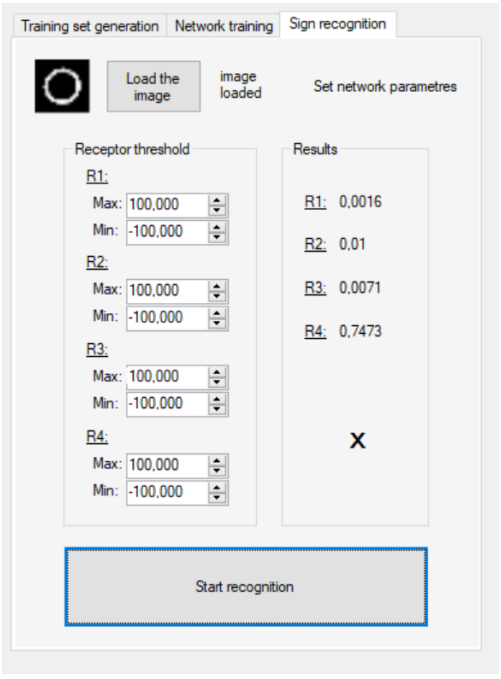
Выбранные знаки:

Результат работы программы:









Алгоритм обучения (порядок подачи изображений из датасета):

100 раз:

берём по небольшому набору (например 20) изображений каждого знака из соответствующей директории (по разу из каждой директории получается 4 набора)

когда изображения в директории заканчиваются алгоритм обнуляет счётчик изображений для каждой директории и берёт изображения с первого.

Время обучения:

Примерно 3 минуты

Код программы:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;

namespace Laba_2
{
  public partial class Form1 : Form
  {

    Bitmap loaded_signs_bitmap;
    Bitmap loaded_kernels_bitmap;

    int[,] matKernel1 = new int[5, 5];
    int[,] matKernel2 = new int[5, 5];
    int[,] matKernel3 = new int[5, 5];
    int[,] matKernel4 = new int[5, 5];

    int[,] matKernelSize = new int[2, 2];

    int[,] matSign = new int[20, 20];

    int nowSet = 0;

    bool imgLoaded = false;

    double[] matAval = new double[512];
    double[] matAvalFunc = new double[512];

    double[] matRval = new double[4];
    double[] matRvalFunc = new double[4];
```

```
double[,] matSAweig = new double[512, 256];
double[,] matARweig = new double[4, 512];

Bitmap imageForRecBitmap;
//Bitmap formatedImageForRecBitmap;
int[] imageForRec = new int[256];

Random randSAconVal = new Random();
Random colCon = new Random();
Random randSAweigVal = new Random();
Random randAactVal = new Random();
Random randARweigVal = new Random();
Random randARweigDouble = new Random();

Random noize_in = new Random();
Random brightness_in = new Random();
Random angle_in = new Random();
Random shift_x_in = new Random();
Random shift_y_in = new Random();

public Form1()
{
  InitializeComponent();
}

private void Load_signs(object sender, EventArgs e)
{
  try
  {
    System.Windows.Forms.OpenFileDialog dlg = new System.Windows.Forms.OpenFileDialog();
    dlg.FileName = "Document";
    dlg.DefaultExt = ".png";
    dlg.Filter = "Text documents (.png)|*.png";

    DialogResult result = dlg.ShowDialog();

    string filename = dlg.FileName;

    loaded_signs_bitmap = new Bitmap(filename);

    img_org_signs.Image = loaded_signs_bitmap;
  }
  catch
  { }
}

private void LoadKernels(object sender, EventArgs e)
{
  try
  {
    System.Windows.Forms.OpenFileDialog dlg = new System.Windows.Forms.OpenFileDialog();
    dlg.FileName = "Document";
    dlg.DefaultExt = ".png";
```

```
    dlg.Filter = "Text documents (.png)|*.png";

    DialogResult result = dlg.ShowDialog();

    string filename = dlg.FileName;

    loaded_kernels_bitmap = new Bitmap(filename);

    imgKernels.Image = loaded_kernels_bitmap;

    for (int x = 0; x < 5; x++)
      for (int y = 0; y < 5; y++)
        if (loaded_kernels_bitmap.GetPixel(x, y).R > 150)
          matKernel1[x, y] = 0;
        else
          matKernel1[x, y] = 1;

    for (int x = 0; x < 5; x++)
      for (int y = 0; y < 5; y++)
        if (loaded_kernels_bitmap.GetPixel(x + 5, y).R > 150)
          matKernel2[x, y] = 0;
        else
          matKernel2[x, y] = 1;

    for (int x = 0; x < 5; x++)
      for (int y = 0; y < 5; y++)
        if (loaded_kernels_bitmap.GetPixel(x + 10, y).R > 150)
          matKernel3[x, y] = 0;
        else
          matKernel3[x, y] = 1;

    string str = "";    //это для проверки что там у нас записалось в матрицу

    for (int x = 0; x < 5; x++)
    {
      for (int y = 0; y < 5; y++)
      {
        if (loaded_kernels_bitmap.GetPixel(x + 15, y).R > 150)
          matKernel4[x, y] = 0;
        else
          matKernel4[x, y] = 1;

        str = str + matKernel4[x, y].ToString();
      }
      str = str + "\n\r";
    }
    //MessageBox.Show(str);   //убрать визуализацию ядра



    labelSAc.Text = "loaded";
    labelSAc.ForeColor = Color.Black;
  }
  catch
```

```
        { }
    }

    private void Generate_set(object sender, EventArgs e)
    {
        try
        {
            for (int k = 0; k < 4; k++)
            {
                FileInfo del = new FileInfo(pathSets.Text + "\\set_" + k + ".bmp");
                del.Delete();
            }
        }
        catch { }

        for (int i = 0; i < 4; i++)     //общий цикл для 4-х изображений
        {
            Bitmap sign = new Bitmap(20, 20);
            Graphics buff_sign = Graphics.FromImage(sign);
            imageForRecBitmap = new Bitmap(loaded_signs_bitmap);
            buff_sign.Clear(Color.Black);
            buff_sign.DrawImage(imageForRecBitmap, 0 - i * 20, 0, 80, 20);

            Bitmap set_sign = new Bitmap(20, 20);
            set_sign = sign;

            for (int m = 0; m < 60; m++)
            {
                //поварачиваем
                set_sign = rotateImage(sign, angle_in.Next(Convert.ToInt32(minAngle.Value),
Convert.ToInt32(maxAngle.Value)));

                //двигаем
                //set_sign = shiftImage(set_sign, shift_x_in.Next(-Convert.ToInt32(minBias.Value),
Convert.ToInt32(maxBias.Value)), shift_y_in.Next(Convert.ToInt32(minBias.Value),
Convert.ToInt32(maxBias.Value)));

                //выбираем яркость
                int brightness_is = brightness_in.Next(Convert.ToInt32(minBrig.Value),
Convert.ToInt32(maxBrig.Value));
                int color_brightness;

                for (int x = 0; x < 20; x++)
                {
                    for (int y = 0; y < 20; y++)
                    {
                        if (set_sign.GetPixel(x, y).R < 200)
                        {
                            color_brightness = brightness_is;
                            set_sign.SetPixel(x, y, Color.FromArgb(color_brightness, color_brightness,
color_brightness));
                        }
                    }
                }
```

```
            //шумим
            for (int x = 0; x < 20; x++)
            {
               for (int y = 0; y < 20; y++)
               {
                  int rand_noise = noize_in.Next(Convert.ToInt32(minNoize.Value),
Convert.ToInt32(maxNoize.Value));
                  int color_noise;

                  color_noise = NoizeNorm(set_sign.GetPixel(x, y).R, rand_noise);

                  set_sign.SetPixel(x, y, Color.FromArgb(color_noise, color_noise, color_noise));
               }
            }

            try
            {
               set_sign.Save(@"set_" + m + ".bmp", System.Drawing.Imaging.ImageFormat.Bmp);
               FileInfo BmpSet = new FileInfo(@"set_" + m + ".bmp");
               BmpSet.MoveTo(pathSets.Text + "\\" + (i + 1).ToString() + "\\set_" + m + ".bmp");
            }
            catch (Exception er)
            {
               MessageBox.Show(er.Message);
            }
         }
      }
   }

   private int NoizeNorm(int pixel, int noize)
   {
      if ((pixel + noize <= 255) && (pixel + noize >= 0))
         return pixel + noize;
      else
      if (pixel + noize > 255)
         return 255;
      else
         return 0;
   }

   private Bitmap rotateImage(Bitmap b, float angle)
   {
      //create a new empty bitmap to hold rotated image
      Bitmap returnBitmap = new Bitmap(20, 20);
      //make a graphics object from the empty bitmap
      Graphics g = Graphics.FromImage(returnBitmap);
      g.Clear(Color.Black);
      //move rotation point to center of image
      g.TranslateTransform(10, 10);
      //rotate
      g.RotateTransform(angle);
      //move image back
      g.TranslateTransform(-10, -10);
```

```
    //draw passed in image onto graphics object
    g.DrawImage(b, new Point(0, 0));

    return returnBitmap;
}

private Bitmap shiftImage(Bitmap b, int x, int y)
{
    //create a new empty bitmap to hold rotated image
    Bitmap returnBitmap = new Bitmap(20, 20);
    //make a graphics object from the empty bitmap
    Graphics g = Graphics.FromImage(returnBitmap);
    g.Clear(Color.Black);
    //move rotation point to center of image
    g.DrawImage(b, new Point(x, y));

    return returnBitmap;
}

private void Label8_Click(object sender, EventArgs e) { }

private void LoadSAconnections(object sender, EventArgs e) { }

private void RandSAconnections(object sender, EventArgs e) { }

private void SaveSAconnections(object sender, EventArgs e) { }

private void LoadSAweights(object sender, EventArgs e)
{
    //try
    //{
    //    System.Windows.Forms.OpenFileDialog dlg = new System.Windows.Forms.OpenFileDialog();
    //    dlg.FileName = "Document";
    //    dlg.DefaultExt = ".csv";
    //    dlg.Filter = "Text documents (.csv)|*.csv";
    //    DialogResult result = dlg.ShowDialog();
    //    string filename = dlg.FileName;
    //    //MessageBox.Show(filename);
    //    try
    //    {
    //        int iRow = 0;
    //        using (StreamReader file = new StreamReader(filename))
    //        {
    //            while (file.Peek() >= 0)
    //            {
    //                string[] rowSplit = file.ReadLine().Split(';');
    //                matA1weig[iRow, 0] = Convert.ToInt32(rowSplit[0]);
    //                matA1weig[iRow, 1] = Convert.ToInt32(rowSplit[1]);
    //                matA1weig[iRow, 2] = Convert.ToInt32(rowSplit[2]);
    //                iRow++;
    //            }
    //        }
    //    }
    //    catch
```

```
    //    {
    //        MessageBox.Show("Load error! Incorrect data.");
    //    }
    //    labelSAw.Text = "loaded";
    //    labelSAw.ForeColor = Color.Black;
    //    l2.Text = "✓";
    //    r2.Text = "";
    //    s2.Text = "";
    //    saveSAweig.Enabled = true;
    //}
    //catch
    //{ }
}

private void RandSAweights(object sender, EventArgs e)
{
    for (int i = 0; i < 512; i++)
    {
        for (int j = 0; j < 256; j++)
        {
            matSAweig[i, j] = randSAweigVal.NextDouble() - 0.5;
        }
    }

    labelSAw.Text = "generated";
    labelSAw.ForeColor = Color.Black;
    l2.Text = "";
    r2.Text = "✓";
    s2.Text = "";
    saveSAweig.Enabled = true;
}

private void SaveSAweights(object sender, EventArgs e)
{
    string listText = "";

    for (int i = 0; i < 512; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            listText += matSAweig[i, j] + ";";
        }
        if (i != 511)
            listText += "\r\n";
    }

    SaveFileDialog save = new SaveFileDialog();

    save.Filter = "csv files (*.csv)|*.csv|All files (*.*)|*.*";
    save.FilterIndex = 1;
    save.RestoreDirectory = true;

    File.WriteAllText(pathToWorkDir.Text + "\\2_SA-weights.csv", listText);
```

```
    s2.Text = "✓";
}

private void LoadAactivation(object sender, EventArgs e) { }

private void RandAactivation(object sender, EventArgs e) { }

private void SaveAactivation(object sender, EventArgs e) { }

private void LoadARweights(object sender, EventArgs e)
{
    //try
    //{
    //    System.Windows.Forms.OpenFileDialog dlg = new System.Windows.Forms.OpenFileDialog();
    //    dlg.FileName = "Document";
    //    dlg.DefaultExt = ".csv";
    //    dlg.Filter = "Text documents (.csv)|*.csv";
    //    DialogResult result = dlg.ShowDialog();
    //    string filename = dlg.FileName;
    //    //MessageBox.Show(filename);
    //    try
    //    {
    //        int iRow = 0;
    //        using (StreamReader file = new StreamReader(filename))
    //        {
    //            while (file.Peek() >= 0)
    //            {
    //                string[] line = file.ReadLine().Split(';');
    //                for (int i = 0; i < 512; i++)
    //                {
    //                    matARweig[iRow, i] = Convert.ToDouble(line[i]);
    //                }
    //                iRow++;
    //            }
    //        }
    //    }
    //    catch
    //    {
    //        MessageBox.Show("Load error! Incorrect data.");
    //    }
    //}
    //catch
    //{ }
    //labelARw.Text = "loaded";
    //labelARw.ForeColor = Color.Black;
    //l4.Text = "✓";
    //r4.Text = "";
    //s4.Text = "";
    //saveARweig.Enabled = true;
}

private void RandARweights(object sender, EventArgs e)
{
    for (int i = 0; i < 4; i++)
```

```
    {
        for (int j = 0; j < 512; j++)
        {
            matARweig[i, j] = randARweigDouble.NextDouble() - 0.5;
        }
    }

    labelARw.Text = "generated";
    labelARw.ForeColor = Color.Black;
    l4.Text = "";
    r4.Text = "✓";
    s4.Text = "";
    saveARweig.Enabled = true;
}

private void SaveARweights(object sender, EventArgs e)
{
    string listText = "";

    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 512; j++)
        {
            listText += matARweig[i, j] + ";";
        }
        if (i != 511)
            listText += "\r\n";
        //MessageBox.Show(i.ToString());
    }

    SaveFileDialog save = new SaveFileDialog();

    save.Filter = "csv files (*.csv)|*.csv|All files (*.*)|*.*";
    save.FilterIndex = 1;
    save.RestoreDirectory = true;

    File.WriteAllText(pathToWorkDir.Text + "\\4_AR-weights.csv", listText);

    s4.Text = "✓";
}

private void BrowseWorkDir(object sender, EventArgs e)
{
    using (var dialog = new System.Windows.Forms.FolderBrowserDialog())
    {
        System.Windows.Forms.DialogResult result = dialog.ShowDialog();

        if (result == System.Windows.Forms.DialogResult.OK)
        {
            pathToWorkDir.Text = dialog.SelectedPath;
        }

    }
}
```

```
        private void StartTraining(object sender, EventArgs e)
        {
            //matA1weig
            //matA2weig
            //matRval
            //signNumber

            //try
            //{
            int[] dataCounter = new int[] {-1, -1, -1, -1};
            //double dif = 100;

            for (int it = 0; it < 100; it++)
            {
                for (int ds = 0; ds < 4; ds++)
                {
                    nowSet = ds;
                    for (int im = 0; im < datasizeNum.Value-1; im++)
                    {
                        string path = dataPathWindow.Text;

                        if(dataCounter[ds] < 59)
                            dataCounter[ds]++;
                        else
                            dataCounter[ds] = 0;

                        int[,] sign20x20 = ImageFromDataset(dataCounter[ds], path + "\\" + (ds+1).ToString());

                        //Here I am!
                        //img.Image = new Bitmap(path + "\\" + (ds + 1).ToString() + "\\set_" + im + ".bmp");
                        //MatrixShow("sign 20x20", sign20x20, 20, 20, true);

                        int[,] signMap16x16_1 = Convolution(sign20x20, matKernel1);
                        int[,] signMap16x16_2 = Convolution(sign20x20, matKernel2);
                        int[,] signMap16x16_3 = Convolution(sign20x20, matKernel3);
                        int[,] signMap16x16_4 = Convolution(sign20x20, matKernel4);

                        //MatrixShow("map 16x16", signMap16x16_1, 16, 16, true);

                        int[,] signMap8x8_1 = Zip(signMap16x16_1);
                        int[,] signMap8x8_2 = Zip(signMap16x16_2);
                        int[,] signMap8x8_3 = Zip(signMap16x16_3);
                        int[,] signMap8x8_4 = Zip(signMap16x16_4);

                        //MatrixShow("map 8x8", signMap8x8_1, 8, 8, true);

                        int[] vectorProp = ForNeuro(signMap8x8_1, signMap8x8_2, signMap8x8_3,
signMap8x8_4);

                        Net(vectorProp);

                        double[] difArr = new double[4];
                        double[] difArrA = new double[512];
```

```
                double[] difArrAFunc = new double[512];

                //вычисляю ошибку на выходном слое
                for (int i = 0; i < 4; i++)
                {
                   //if (Convert.ToInt32(signNumber.Text) == i + 1)
                   if (nowSet == i)
                      difArr[i] = 0.01*(1 - matRvalFunc[i]) * Sigmoida(matRval[i]) * (1 -
Sigmoida(matRval[i]));
                   else
                      difArr[i] = 0.01 * (0 - matRvalFunc[i]) * Sigmoida(matRval[i]) * (1 -
Sigmoida(matRval[i]));
                   //MessageBox.Show(difArr[i].ToString());
                }

                //вычисляю корректировку весов между выходным и скрытыми слоями
                double[,] matARweightsCorr = new double[4, 512];

                for (int i = 0; i < 4; i++)
                {
                   for (int j = 0; j < 512; j++)
                   {
                      matARweightsCorr[i, j] = difArr[i] * matAvalFunc[j];
                   }
                }

                //вычисляю ошибку на нейронах скрытого слоя
                double sumErA = 0;

                for (int i = 0; i < 512; i++)
                {
                   sumErA = 0;

                   for (int j = 0; j < 4; j++)
                      sumErA += difArr[j] * matARweig[j, i];

                   difArrA[i] = sumErA;
                   difArrAFunc[i] = difArrA[i] * Sigmoida(matAval[i]) * (1 - Sigmoida(matAval[i]));
                   //MessageBox.Show((Sigmoida(matAval[i]) * (1 - Sigmoida(matAval[i]))).ToString());
                   //MessageBox.Show(difArrA[i].ToString());
                }

                //вычисляю корректировку весов между входным и скрытыми слоями
                double[,] matSAweightsCorr = new double[512, 256];

                for (int i = 0; i < 512; i++)
                {
                   for (int j = 0; j < 256; j++)
                   {
                      matSAweightsCorr[i, j] = difArrAFunc[i] * vectorProp[j];
                      //MessageBox.Show(matSAweightsCorr[i,j].ToString());
                      //MessageBox.Show(difArrAFunc[j].ToString());
                   }
                }
```

```
                //обновляю веса SA
                for (int i = 0; i < 512; i++)
                  for (int j = 0; j < 256; j++)
                  {
                    matSAweig[i, j] += matSAweightsCorr[i, j];
                    //MessageBox.Show(matSAweightsCorr[i, j].ToString());
                  }

                for (int i = 0; i < 4; i++)
                  for (int j = 0; j < 512; j++)
                  {
                    matARweig[i, j] += matARweightsCorr[i, j];
                  }



                //double max = Math.Max(Math.Max(difArr[0], difArr[1]), Math.Max(difArr[2], difArr[3]));
                //double min = Math.Min(Math.Min(difArr[0], difArr[1]), Math.Min(difArr[2], difArr[3]));

                //MessageBox.Show(Math.Min(Math.Min(difArr[0], difArr[1]), Math.Min(difArr[2],
difArr[3])).ToString());
                //MessageBox.Show(Math.Max(Math.Max(difArr[0], difArr[1]), Math.Max(difArr[2],
difArr[3])).ToString());
                //MessageBox.Show("Max: " + max + "\n\r" + "Min: " + min);
              }
            }
          }

          //for (int r = 0; r < 600; r++)
          //{
          //  if (dataCounter < datasizeNum.Value - 1)
          //    dataCounter++;
          //  else
          //    dataCounter = 0;

          //  string path = dataPathWindow.Text;

          //  int[,] sign20x20 = ImageFromDataset(dataCounter, path);

          //  //img.Image = new Bitmap(dataPathWindow.Text + "\\set_" + dataCounter + ".bmp");
          //  //MatrixShow(sign20x20, 20, 20, true);

          //  int[,] signMap16x16_1 = Convolution(sign20x20, matKernel1);
          //  int[,] signMap16x16_2 = Convolution(sign20x20, matKernel2);
          //  int[,] signMap16x16_3 = Convolution(sign20x20, matKernel3);
          //  int[,] signMap16x16_4 = Convolution(sign20x20, matKernel4);

          //  int[,] signMap8x8_1 = Zip(signMap16x16_1);
          //  int[,] signMap8x8_2 = Zip(signMap16x16_2);
          //  int[,] signMap8x8_3 = Zip(signMap16x16_3);
          //  int[,] signMap8x8_4 = Zip(signMap16x16_4);

          //  int[] vectorProp = ForNeuro(signMap8x8_1, signMap8x8_2, signMap8x8_3, signMap8x8_4);
```

```
//   Net(vectorProp);

//   double[] difArr = new double[4];
//   double[] difArrA = new double[512];
//   double[] difArrAFunc = new double[512];

//   //вычисляю ошибку на выходном слое
//   for (int i = 0; i < 4; i++)
//   {
//      if (Convert.ToInt32(signNumber.Text) == i + 1)
//         difArr[i] = (1 - matRvalFunc[i]) * Sigmoida(matRval[i]) * (1 - Sigmoida(matRval[i]));
//      else
//         difArr[i] = (0 - matRvalFunc[i]) * Sigmoida(matRval[i]) * (1 - Sigmoida(matRval[i]));
//      //MessageBox.Show(difArr[i].ToString());
//   }

//   //вычисляю корректировку весов между выходным и скрытыми слоями
//   double[,] matARweightsCorr = new double[4, 512];

//   for (int i = 0; i < 4; i++)
//   {
//      for (int j = 0; j < 512; j++)
//      {
//         matARweightsCorr[i, j] = difArr[i] * matAvalFunc[j];
//      }
//   }

//   //вычисляю ошибку на нейронах скрытого слоя
//   double sumErA = 0;

//   for (int i = 0; i < 512; i++)
//   {
//      sumErA = 0;

//      for (int j = 0; j < 4; j++)
//         sumErA += difArr[j] * matARweig[j,i];

//      difArrA[i] = sumErA;
//      difArrAFunc[i] = difArrA[i]*Sigmoida(matAval[i])*(1 - Sigmoida(matAval[i]));
//      //MessageBox.Show((Sigmoida(matAval[i]) * (1 - Sigmoida(matAval[i]))).ToString());
//      //MessageBox.Show(difArrA[i].ToString());
//   }

//   //вычисляю корректировку весов между входным и скрытыми слоями
//   double[,] matSAweightsCorr = new double[512, 256];

//   for (int i = 0; i < 512; i++)
//   {
//      for (int j = 0; j < 256; j++)
//      {
//         matSAweightsCorr[i, j] = difArrAFunc[i] * vectorProp[j];
//         //MessageBox.Show(matSAweightsCorr[i,j].ToString());
//         //MessageBox.Show(difArrAFunc[j].ToString());
```

```
        //      }
        //  }

        //  //обновляю веса SA
        //  for (int i = 0; i < 512; i++)
        //      for (int j = 0; j < 256; j++)
        //      {
        //          matSAweig[i, j] += matSAweightsCorr[i, j];
        //          //MessageBox.Show(matSAweightsCorr[i, j].ToString());
        //      }

        //  for (int i = 0; i < 4; i++)
        //      for (int j = 0; j < 512; j++)
        //      {
        //          matARweig[i, j] += matARweightsCorr[i, j];
        //      }



        //  //double max = Math.Max(Math.Max(difArr[0], difArr[1]), Math.Max(difArr[2], difArr[3]));
        //  //double min = Math.Min(Math.Min(difArr[0], difArr[1]), Math.Min(difArr[2], difArr[3]));

        //  //MessageBox.Show(Math.Min(Math.Min(difArr[0], difArr[1]), Math.Min(difArr[2],
difArr[3])).ToString());
        //  //MessageBox.Show(Math.Max(Math.Max(difArr[0], difArr[1]), Math.Max(difArr[2],
difArr[3])).ToString());
        //  //MessageBox.Show("Max: " + max + "\n\r" + "Min: " + min);
        //}



        //}
        //catch (Exception er)
        //{
        //  MessageBox.Show(er.Message);
        //}
    }

    private int[,] ImageFromDataset(int imageNum, string path)
    {
        int[,] image = new int[20, 20];

        //MessageBox.Show(path + "\\set_" + imageNum + ".bmp");
        Bitmap propBitmap = new Bitmap(path + "\\set_" + imageNum + ".bmp");

        Bitmap buff = new Bitmap(20, 20);

        Graphics buff_graph = Graphics.FromImage(buff);

        buff_graph.DrawImage(propBitmap, 0, 0, 20, 20);

        for (int i = 0; i < 20; i++)
        {
            for (int j = 0; j < 20; j++)
            {
```

```
            image[i, j] = NormBrightness(buff.GetPixel(i, j).R);
          }
        }

        return image;
      }



      private void eatThis(Bitmap pict)
      {
        //int[] pictLine = new int[256];
        //int pictCounter = -1;
        //for (int i = 0; i < 16; i++)
        //{
        //   for (int j = 0; j < 16; j++)
        //   {
        //      pictCounter++;
        //      if (pict.GetPixel(i, j).R >= Convert.ToInt32(dataTreshold.Value))
        //         pictLine[pictCounter] = 1;
        //      else
        //         pictLine[pictCounter] = 0;
        //   }
        //}
        //if ((labelSAc.Text != "required") && (labelSAw.Text != "required") && (labelAact.Text !=
"required") && (labelARw.Text != "required"))
        //{
        //   for (int i = 0; i < 512; i++)   //суммируем сигналы на нейронах скрытого слоя
        //   {
        //      int sum = 0;
        //      int conCount = 0;
        //      if (matSAcon[i, 0] != -1)
        //      {
        //         sum += pictLine[matSAcon[i, 0]] * matSAweig[i, 0];
        //         conCount++;
        //      }
        //      if (matSAcon[i, 1] != -1)
        //      {
        //         sum += pictLine[matSAcon[i, 1]] * matSAweig[i, 1];
        //         conCount++;
        //      }
        //      if (matSAcon[i, 2] != -1)
        //      {
        //         sum += pictLine[matSAcon[i, 2]] * matSAweig[i, 2];
        //         conCount++;
        //      }
        //      if (strategySelect.Text == "strategy 1")
        //      {
        //         if (sum >= 1)
        //            matAval[i] = 1;
        //         else
        //            matAval[i] = 0;
        //      }
        //      if (strategySelect.Text == "strategy 2")
```

```
    //      {
    //          if ((sum >= 2) && (conCount > 2))
    //              matAval[i] = 1;
    //          else if ((sum >= 1) && (conCount <= 2))
    //              matAval[i] = 1;
    //          else
    //              matAval[i] = 0;
    //      }
    //      if (strategySelect.Text == "strategy 3")
    //      {
    //          if ((sum >= 3) && (conCount >= 3))
    //              matAval[i] = 1;
    //          else if ((sum < 3) && (conCount < 3))
    //              matAval[i] = 1;
    //          else
    //              matAval[i] = 0;
    //      }
    //      for (int t = 0; t < 4; t++)
    //      {
    //          if (matAval[i] == 1)
    //          {
    //              if (Convert.ToInt32(signNumber.SelectedItem) == t+1)
    //                  matARweig[t, i]++;
    //              else
    //                  matARweig[t, i]--;
    //          }
    //      }
    //  }
    //  for (int k = 0; k < 4; k++)   //суммируем сигналы на нейронах выходного слоя
    //  {
    //      double sumR = 0;
    //      for (int j = 0; j < 512; j++)
    //      {
    //          sumR += matAval[j] * matARweig[k, j];
    //      }
    //      matRval[k] = sumR;
    //  }
    //}
    //else
    //{
    //  MessageBox.Show("Don't forget to load an image and set net parametrs!");
    //}
}

private void SaveParameters(object sender, EventArgs e)
{
    string listText = "";

    for (int i = 0; i < 512; i++)
    {
        for (int j = 0; j < 512; j++)
        {
            listText += matSAweig[i, j] + ";";
        }
```

```
        if (i != 511)
            listText += "\r\n";
    }

    SaveFileDialog save = new SaveFileDialog();

    save.Filter = "csv files (*.csv)|*.csv|All files (*.*)|*.*";
    save.FilterIndex = 1;
    save.RestoreDirectory = true;

    File.WriteAllText(pathToWorkDir.Text + "\\4_AR-weights_smart.csv", listText);
}

private void Form1_Load(object sender, EventArgs e)
{
    l1.Text = "";
    r1.Text = "";
    s1.Text = "";

    l2.Text = "";
    r2.Text = "";
    s2.Text = "";

    l3.Text = "";
    r3.Text = "";
    s3.Text = "";

    l4.Text = "";
    r4.Text = "";
    s4.Text = "";

    bird1.Text = "";
    bird2.Text = "";
    bird3.Text = "";
    bird4.Text = "";
}

private void BrowsePathToSaveSet(object sender, EventArgs e)
{
    using (var dialog = new System.Windows.Forms.FolderBrowserDialog())
    {
        System.Windows.Forms.DialogResult result = dialog.ShowDialog();
        if (result == System.Windows.Forms.DialogResult.OK)
        {
            pathSets.Text = dialog.SelectedPath;
        }
    }
}

public double[] Neuro(int[] image, int[,] conSA, int[,] weigSA, int[] actA, double[,] weigAR)
{
    double[] output = new double[4];
    int[] resultOfFirstStep = new int[512];
```

```
// Step 1

for (int i = 0; i < 512; i++)
{
   int sum = 0;

   int colOfZero = 0;

   for (int j = 0; j < 3; j++)
   {
      if (weigSA[i, j] != -2)
         sum += weigSA[i, j] * image[conSA[i, j]];
   }

   switch (actA[i])
   {
      case 1:
         if (sum >= 1)
            resultOfFirstStep[i] = 1;
         else
            resultOfFirstStep[i] = 0;
         break;

      case 2:
         colOfZero = 0;

         for (int z = 0; z < 3; z++)
         {
            if (weigSA[i, z] == -2)
               colOfZero++;
         }

         if (colOfZero == 0)
         {
            if (sum >= 2)
               resultOfFirstStep[i] = 1;
            else
               resultOfFirstStep[i] = 0;
         }
         else
         {
            if (sum >= 1)
               resultOfFirstStep[i] = 1;
            else
               resultOfFirstStep[i] = 0;
         }
         break;

      case 3:
         colOfZero = 0;

         for (int z = 0; z < 3; z++)
         {
            if (weigSA[i, z] == -2)
```

```
                colOfZero++;
            }

            if (colOfZero == 0)
            {
               if (sum >= 3)
                  resultOfFirstStep[i] = 1;
               else
                  resultOfFirstStep[i] = 0;
            }
            else
            {
               resultOfFirstStep[i] = 1;
            }
            break;
      }  //the end of the switch(){}
   }

   // Step 2.

   for (int i = 0; i < 4; i++)
   {
      double sum = 0;

      for (int j = 0; j < 512; j++)
      {
         sum += resultOfFirstStep[j] * weigAR[i, j];
      }

      output[i] = sum;
   }

   return output;
}

private void Load16x16Image(object sender, EventArgs e)
{
   System.Windows.Forms.OpenFileDialog dlg = new System.Windows.Forms.OpenFileDialog();
   dlg.FileName = "Document";
   dlg.DefaultExt = ".png";
   dlg.Filter = "Text documents (.png)|*.png";

   DialogResult result = dlg.ShowDialog();

   string filename = dlg.FileName;

   Bitmap buff = new Bitmap(20, 20);

   Graphics buff_graph = Graphics.FromImage(buff);

   imageForRecBitmap = new Bitmap(filename);

   buff_graph.DrawImage(imageForRecBitmap, 0, 0, 20, 20);
```

```
            pictToRec.Image = buff;

            for (int i = 0; i < 20; i++)
            {
               for (int j = 0; j < 20; j++)
               {
                  matSign[i, j] = NormBrightness(buff.GetPixel(i, j).R);
               }
            }

            labelImageRec.Text = "loaded";
            labelImageRec.ForeColor = Color.Black;
            imgLoaded = true;
         }

         private int NormBrightness(int brIn)
         {
            return Convert.ToInt32((brIn*7)/255);
         }

         private void StartRecognitionButt(object sender, EventArgs e)
         {
            if ((imgLoaded) && (labelSAc.Text != "required") && (labelSAw.Text != "required") &&
(labelARw.Text != "required"))
            {
               //сделали 4 карты 16x16 из знака и ядер
               int[,] matSignMap1 = Convolution(matSign, matKernel1);
               int[,] matSignMap2 = Convolution(matSign, matKernel2);
               int[,] matSignMap3 = Convolution(matSign, matKernel3);
               int[,] matSignMap4 = Convolution(matSign, matKernel4);

               //знак(вход), свернутый знак(выход), ядро, "единички и нолики" \ "значения"
               //MatrixShow(matSignMap1, 16, 16, true);

               //сжимаем карту до 8x8
               int[,] matSignMapSize1 = Zip(matSignMap1);
               int[,] matSignMapSize2 = Zip(matSignMap2);
               int[,] matSignMapSize3 = Zip(matSignMap3);
               int[,] matSignMapSize4 = Zip(matSignMap4);


               //делаем из картинок вектор для нейронной сети
               int[] neuroIn = ForNeuro(matSignMapSize1, matSignMapSize2, matSignMapSize3,
matSignMapSize4);

               //пускаем вектор на перцептрон (256, 512, 4) и получаем реузльтат
               Net(neuroIn);

               labelR1.Text = Math.Round(matRvalFunc[0], 4).ToString();
               labelR2.Text = Math.Round(matRvalFunc[1], 4).ToString();
               labelR3.Text = Math.Round(matRvalFunc[2], 4).ToString();
               labelR4.Text = Math.Round(matRvalFunc[3], 4).ToString();

            }
```

```
    else
    {
        MessageBox.Show("Error! Don't forget to load an image, a kernel and set network weights!");
    }
}

private int[,] Convolution(int[,] sign, int[,] kernel)
{
    int[,] signMap = new int[16, 16];

    Bitmap bitmap = new Bitmap(5, 5);
    for (int i = 0; i < 16; i++)
    {
        for (int j = 0; j < 16; j++)
        {
            int sum = 0;

            for (int m = 0; m < 5; m++)
            {
                for (int n = 0; n < 5; n++)
                {
                    //if (!((m == 2) && (n == 2)))
                    //{
                        //int c = matSign[i + m, j + n];
                        //bitmap.SetPixel(m, n, Color.FromArgb(c,c,c));

                        sum += sign[i + m, j + n] * kernel[m, n];
                    //
                }
            }

            signMap[i, j] = sum;
            //img.Image = bitmap;
            //tab.SelectTab(1);
            //MessageBox.Show("");
        }
    }

    return signMap;
}

private int[,] Zip(int[,] map)
{
    int[,] zip = new int[8, 8];
    int max = 0;

    for (int i = 0; i < 16; i += 4)
        for (int j = 0; j < 16; j += 4)
        {
            max = 0;
            for (int x = i; x < i+2; x++)
                for (int y = j; y < j+2; y++)
                {
                    if (map[x, y] > max)
```

```
                    max = map[x, y];
                }
            zip[Convert.ToInt32(i / 2), Convert.ToInt32(j / 2)] = max;
            //MessageBox.Show(max.ToString());
            //MessageBox.Show(Convert.ToInt32(i / 2).ToString() + " " + Convert.ToInt32(j /
2).ToString());
        }

      for (int i = 2; i < 16; i += 4)
        for (int j = 0; j < 16; j += 4)
        {
          max = 0;
          for (int x = i; x < i+2; x++)
            for (int y = j; y < j+2; y++)
            {
              if (map[x, y] > max)
                max = map[x, y];
            }
          zip[Convert.ToInt32(i / 2), Convert.ToInt32(j / 2)] = max;
        }

      for (int i = 0; i < 16; i += 4)
        for (int j = 2; j < 16; j += 4)
        {
          max = 0;
          for (int x = i; x < i+2; x++)
            for (int y = j; y < j+2; y++)
            {
              if (map[x, y] > max)
                max = map[x, y];
            }
          zip[Convert.ToInt32(i / 2), Convert.ToInt32(j / 2)] = max;
        }

      for (int i = 2; i < 16; i += 4)
        for (int j = 2; j < 16; j += 4)
        {
          max = 0;
          for (int x = i; x < i+2; x++)
            for (int y = j; y < j+2; y++)
            {
              if (map[x, y] > max)
                max = map[x, y];
            }
          zip[Convert.ToInt32(i / 2), Convert.ToInt32(j / 2)] = max;
        }
      //MatrixShow(map, 16, 16, true);
      //MatrixShow(zip, 8, 8, true);

      return zip;
    }

    private int[] ForNeuro(int[,] map1, int[,] map2, int[,] map3, int[,] map4)
    {
```

```
    int[] vector = new int[256];

    int counter = -1;

    for (int i = 0; i < 8; i++)
        for (int j = 0; j < 8; j++)
        {
            counter++;
            vector[counter] = map1[i, j];
        }

    for (int i = 0; i < 8; i++)
        for (int j = 0; j < 8; j++)
        {
            counter++;
            vector[counter] = map2[i, j];
        }

    for (int i = 0; i < 8; i++)
        for (int j = 0; j < 8; j++)
        {
            counter++;
            vector[counter] = map3[i, j];
        }

    for (int i = 0; i < 8; i++)
        for (int j = 0; j < 8; j++)
        {
            counter++;
            vector[counter] = map4[i, j];
        }

    //string str = "";
    //for (int i = 0; i < vector.Count(); i++)
    //    str += vector[i] + " ";
    //MessageBox.Show(str.ToString());


    return vector;
}

private void Net(int[] vectorNet)
{
    double[] resNet = new double[4];
    double sum = 0;

    //собираем значения на А слое
    for (int i = 0; i < 512; i++)
    {
        sum = 0;

        for (int j = 0; j < 256; j++)
        {
            sum += matSAweig[i, j] * vectorNet[j];
```

```
        }

        matAval[i] = sum;
        matAvalFunc[i] = Sigmoida(matAval[i]);
    }

    //собираем значения на R слое
    for (int i = 0; i < 4; i++)
    {
        sum = 0;

        for (int j = 0; j < 512; j++)
        {
            sum += matARweig[i, j] * matAvalFunc[j];
        }

        matRval[i] = sum;
        matRvalFunc[i] = Sigmoida(matRval[i]);
    }
}

private double Sigmoida(double x)
{
    double y = 1 / (1 + Math.Pow(Math.E, -x));

    return y;
}



private void BrowseDataset(object sender, EventArgs e)
{
    System.Windows.Forms.OpenFileDialog dlg = new System.Windows.Forms.OpenFileDialog();
    dlg.FileName = "Document";
    dlg.DefaultExt = ".bmp";
    dlg.Filter = "Text documents (.bmp)|*.bmp";

    DialogResult result = dlg.ShowDialog();

    string filename = dlg.FileName;

    dataPathWindow.Text = filename;
}

private void MatrixShow(string msg, int[,] mat, int x, int y, bool par)
{
    string str = msg + "\n\r";

    for (int i = 0; i < x; i++)
    {
        for (int j = 0; j < y; j++)
        {
            if (par)
                str += mat[i, j] + " ";
```

```
            else
            {
               if (mat[i, j] != 0)
                  str += 1 + " ";
               else
                  str += 0 + " ";
            }
         }
         str += "\n\r";
      }
      MessageBox.Show(str);
   }

   private void NumericUpDown10_ValueChanged(object sender, EventArgs e){}

   private void Label23_Click(object sender, EventArgs e){}

   private void NumericUpDown9_ValueChanged(object sender, EventArgs e){}

   private void Label18_Click(object sender, EventArgs e){}

   private void Label20_Click(object sender, EventArgs e){}

   private void ComboBox1_SelectedIndexChanged(object sender, EventArgs e){}
  }
}
```