**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ   
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |
| --- | --- |
| СОГЛАСОВАНО  Старший преподаватель факультета компьютерных наук департамента Программной инженерии  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.В.Пантюхин  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2017 г. | УТВЕРЖДАЮ  Академический руководитель образовательной программы «Программная инженерия»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_2017 г. |
| |  |  | | --- | --- | | Подп. и дата |  | | Инв. № дубл. |  | | Взам. Инв. № |  | | Подп. и дата |  | | Инв. № подл. | RU.17701729.502840-01 12 01-1 | | **Программа распознавания рукописных букв латинского алфавита на основе нейронной сети**  **Текст программы**  **ЛИСТ УТВЕРЖДЕНИЯ**  **RU.17701729.502840-01 12 01-1-ЛУ**  Исполнитель  Студент группы 162 ПИ  \_\_\_\_\_\_\_\_\_\_\_/Кузнецов Д.С./  «\_\_\_»\_\_\_\_\_\_\_\_\_\_2017 г.  **2017** | |  |

УТВЕРЖДЕН

RU.17701729.502840-01 12 01-1-ЛУ

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Подп. и дата |  | | Инв. № дубл. |  | | Взам. Инв. № |  | | Подп. и дата |  | | Инв. № подл. | RU.17701729.502840-01 12 01-1 | | **Программа распознавания рукописных букв латинского алфавита на основе нейронной сети**  **Текст программы**  **RU.17701729.502840-01 12 01-1**  **Листов 11**  **2017** |  |

**Содержание**

[1. ТЕКСТ ПРОГРАММЫ 3](#_Toc482716356)

[1.1 Графический интерфейс для создания изображений на распознавание 3](#_Toc482716357)

[1.2 ИНС для распознавания рукописных латинских букв 6](#_Toc482716358)

[Приложение 1 10](#_Toc482716359)

[ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ 11](#_Toc482716360)

# ТЕКСТ ПРОГРАММЫ

## **1.1 Графический интерфейс для создания изображений на распознавание**

public partial class Form1 : Form

{

Point p1;

string path\_to\_image = @"image.jpg";

string path\_to\_tf\_server = @"http://localhost:5000/";

WebClient wc = new WebClient();

static ImageCodecInfo jpgEncoder = GetEncoder(ImageFormat.Jpeg);

static Encoder enc = Encoder.Quality;

EncoderParameter eParameter = new EncoderParameter(enc, 100L);

EncoderParameters eParameters = new EncoderParameters(1);

public Form1()

{

InitializeComponent();

pictureBox1.Size = new Size(280, 280);

textBox2.Size = new Size(280, 280);

textBox2.Location = pictureBox1.Location;

textBox2.Text = "Загрузка...";

pictureBox1.Image = (Image)new Bitmap(28, 28);

eParameters.Param[0] = eParameter;

}

private void pictureBox1\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

Point p2 = new Point((int)Math.Round(e.X / 10.0), (int)Math.Round(e.Y / 10.0));

Bitmap im = (Bitmap)pictureBox1.Image;

using (Graphics graphics = Graphics.FromImage(im))

{

graphics.DrawLine(new Pen(Color.Black, 3), p1, p2);

}

pictureBox1.Image = im;

p1 = p2;

}

}

private void pictureBox1\_MouseDown(object sender, MouseEventArgs e)

{

p1 = new Point((int)Math.Round(e.X / 10.0), (int)Math.Round(e.Y / 10.0));

}

private void button1\_Click(object sender, EventArgs e)

{

pictureBox1.Image = (Image)new Bitmap(28, 28);

textBox1.Text = "";

pictureBox1.Enabled = true;

}

private void button2\_Click(object sender, EventArgs e)

{

pictureBox1.Enabled = false;

Bitmap savedBit = new Bitmap(28, 28);

pictureBox1.Width = 28;

pictureBox1.Height = 28;

pictureBox1.DrawToBitmap(savedBit, pictureBox1.ClientRectangle);

pictureBox1.Width = 280;

pictureBox1.Height = 280;

savedBit.Save(path\_to\_image, jpgEncoder, eParameters);

try

{

textBox1.Text = wc.DownloadString(path\_to\_tf\_server + path\_to\_image);

}

catch(System.Net.WebException)

{

MessageBox.Show("Ошибка распознавания.\r\n Запустите python-server из Debug каталога проекта и повторите попытку.");

}

}

private static ImageCodecInfo GetEncoder(ImageFormat f)

{

ImageCodecInfo[] codecList = ImageCodecInfo.GetImageDecoders();

foreach (ImageCodecInfo codec in codecList)

{

if (codec.FormatID == f.Guid)

{

return codec;

}

}

return null;

}

}

private void InitializeComponent()

{

this.pictureBox1 = new System.Windows.Forms.PictureBox();

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

this.button2 = new System.Windows.Forms.Button();

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

this.textBox2 = new System.Windows.Forms.TextBox();

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();

this.SuspendLayout();

//

// pictureBox1

//

this.pictureBox1.BackColor = System.Drawing.Color.White;

this.pictureBox1.Location = new System.Drawing.Point(10, 17);

this.pictureBox1.Name = "pictureBox1";

this.pictureBox1.Size = new System.Drawing.Size(280, 280);

this.pictureBox1.SizeMode = System.Windows.Forms.PictureBoxSizeMode.StretchImage;

this.pictureBox1.TabIndex = 0;

this.pictureBox1.TabStop = false;

this.pictureBox1.MouseDown += new System.Windows.Forms.MouseEventHandler(this.pictureBox1\_MouseDown);

this.pictureBox1.MouseMove += new System.Windows.Forms.MouseEventHandler(this.pictureBox1\_MouseMove);

//

// button1

//

this.button1.BackColor = System.Drawing.SystemColors.ButtonShadow;

this.button1.Font = new System.Drawing.Font("Microsoft YaHei UI Light", 12F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.button1.Location = new System.Drawing.Point(396, 17);

this.button1.Name = "button1";

this.button1.Size = new System.Drawing.Size(137, 80);

this.button1.TabIndex = 1;

this.button1.Text = "Очистить";

this.button1.UseVisualStyleBackColor = false;

this.button1.Click += new System.EventHandler(this.button1\_Click);

//

// button2

//

this.button2.BackColor = System.Drawing.SystemColors.ButtonShadow;

this.button2.Font = new System.Drawing.Font("Microsoft YaHei UI Light", 12F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.button2.Location = new System.Drawing.Point(396, 108);

this.button2.Name = "button2";

this.button2.Size = new System.Drawing.Size(137, 80);

this.button2.TabIndex = 2;

this.button2.Text = "Распознать";

this.button2.UseVisualStyleBackColor = false;

this.button2.Click += new System.EventHandler(this.button2\_Click);

//

// textBox1

//

this.textBox1.Cursor = System.Windows.Forms.Cursors.Default;

this.textBox1.Font = new System.Drawing.Font("Microsoft Sans Serif", 73F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.textBox1.Location = new System.Drawing.Point(396, 198);

this.textBox1.Multiline = true;

this.textBox1.Name = "textBox1";

this.textBox1.ReadOnly = true;

this.textBox1.Size = new System.Drawing.Size(137, 163);

this.textBox1.TabIndex = 3;

this.textBox1.TextAlign = System.Windows.Forms.HorizontalAlignment.Center;

//

// textBox2

//

this.textBox2.BackColor = System.Drawing.SystemColors.ButtonHighlight;

this.textBox2.Font = new System.Drawing.Font("Microsoft YaHei UI Light", 30F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.textBox2.Location = new System.Drawing.Point(10, 17);

this.textBox2.Multiline = true;

this.textBox2.Name = "textBox2";

this.textBox2.ReadOnly = true;

this.textBox2.Size = new System.Drawing.Size(280, 280);

this.textBox2.TabIndex = 4;

this.textBox2.Text = "\r\nЗагрузка...";

this.textBox2.TextAlign = System.Windows.Forms.HorizontalAlignment.Center;

//

// Form1

//

this.AutoScaleDimensions = new System.Drawing.SizeF(8F, 16F);

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

this.BackColor = System.Drawing.SystemColors.ControlDarkDark;

this.ClientSize = new System.Drawing.Size(545, 384);

this.Controls.Add(this.textBox1);

this.Controls.Add(this.button2);

this.Controls.Add(this.button1);

this.Controls.Add(this.pictureBox1);

this.Controls.Add(this.textBox2);

this.Name = "Form1";

this.Text = "Handwritten Letters Classifier";

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();

this.ResumeLayout(false);

this.PerformLayout();

}

}

## **1.2 ИНС для распознавания рукописных латинских букв**

import tensorflow as tf  
import numpy as np  
import pandas as pd  
from PIL import Image  
from flask import Flask  
import random  
  
app = Flask(\_\_name\_\_)  
  
*#Constants:--------------------------*input\_data = np.genfromtxt("test\_set.csv", delimiter=',', dtype=np.uint8)  
input\_label = np.genfromtxt("test\_labels\_set.csv", delimiter=',', dtype=np.uint8)  
input\_label = input\_label.T[:6].T  
N\_EXAMPLES = input\_data.shape[0]  
  
  
  
default\_saver\_path = "./logs/"  
  
Alphabet = list("ABCDEF")  
  
sess = tf.InteractiveSession() *#Main session  
#-------------------------------------*def params\_init(shape, name):  
 w = tf.truncated\_normal(shape=shape, stddev=0.1)  
 *#tf.summary.histogram(name, w)* return tf.Variable(w, name= name)  
  
def conv2D\_init(x, kernel\_shape,output\_channels, name, activation=tf.nn.relu):  
 with tf.name\_scope(name):  
 w = params\_init(kernel\_shape, "W")  
 b = params\_init([output\_channels], "b")  
 return tf.nn.relu(tf.nn.conv2d(x, filter=w, strides=[1, 1,1,1], padding='SAME', name=name))  
  
def pool2x2(x, name):  
 return tf.nn.max\_pool(x, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME', name=name)  
  
def fcl\_init(x, in\_neurons, out\_neurons,name):  
 with tf.name\_scope(name):  
 w = params\_init([in\_neurons, out\_neurons], name="W")  
 b = params\_init([out\_neurons], name="b")  
 return(tf.matmul(x, w) + b)  
  
  
s = pd.Series(Alphabet)  
s = pd.get\_dummies(s)  
  
*# Get one hot encoding of given letter*def get\_one\_hot(c):  
 return s.iloc[Alphabet.index(c)].as\_matrix()[np.newaxis].T  
  
*# Return letter using its index in the latin alphabet*def get\_letter(i):  
 return Alphabet[i]  
  
  
  
  
*#---Composing Latin Classifier Net---  
  
  
#Input:*x = tf.placeholder(tf.float32, shape=[None, 784 \* 3]) *# input vectorized images*y = tf.placeholder(tf.float32, shape=[None, 6]) *# Output labels (Onehot encoding)*x\_image = tf.reshape(x, [-1, 28, 28, 3])  
  
*#Convolution-Pool 1:*conv1 = conv2D\_init(x\_image, [5, 5, 3, 32], 32, "conv1")  
pool1 = pool2x2(conv1, "pool1")  
  
*#Convolution-Pool 2:*conv2 = conv2D\_init(pool1, [5, 5, 32, 64], 64, "conv2")  
pool2 = pool2x2(conv2, "pool2")  
  
*#Full-connected layer 1:*vectorized\_pool2 = tf.reshape(pool2, [-1, 7 \* 7 \* 64])  
fcl1 = tf.nn.relu(fcl\_init(vectorized\_pool2, 7 \* 7 \* 64, 1024, "fcl1"))  
  
*#Drop out:*keep\_prob = tf.placeholder(tf.float32, name="Keep\_prob")  
d\_fcl1 = tf.nn.dropout(fcl1, keep\_prob=keep\_prob, name="fcl1\_regul")  
  
*#Full-connected layer 2:*y\_hat = fcl\_init(d\_fcl1, 1024, 6, "fcl2")  
  
*#Loss*cross\_entropy = tf.nn.softmax\_cross\_entropy\_with\_logits(labels=y, logits=y\_hat)  
  
*#Train Step:*train\_step = tf.train.AdamOptimizer(4e-4).minimize(cross\_entropy)  
  
*#Accuracy:*correct\_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y\_hat, 1))  
accuracy = tf.reduce\_mean(tf.cast(correct\_prediction, tf.float32))  
  
*#--------------------------------------------------------------------*saver = tf.train.Saver()  
  
  
def Initialising\_LatinClassifierNet(restore = False, path = default\_saver\_path, ckpt\_name = ""):  
 sess = tf.InteractiveSession()  
 sess.run(tf.initialize\_all\_variables())  
 sess.run(tf.global\_variables\_initializer())  
 save = tf.train.Saver()  
 if (restore):  
 save.restore(sess,path+ckpt\_name+".ckpt")  
  
def Learn(name,train\_set = input\_data, train\_set\_labels=input\_label, batch\_size=15, N\_ATTEMPTIONS=20,epochs=1001,sess=sess,testing=True):  
 list\_accuracy = []  
 for z in range(N\_ATTEMPTIONS):  
 k = 0  
 N\_EXAMPLES = train\_set.shape[0]  
 inx\_seq = range(N\_EXAMPLES)  
 random.shuffle(inx\_seq)  
 sess.run(tf.global\_variables\_initializer())  
 if (testing):  
 test\_inx = inx\_seq[:N\_EXAMPLES // 10]  
 inx\_seq = inx\_seq[N\_EXAMPLES // 10:]  
 N\_EXAMPLES = N\_EXAMPLES - N\_EXAMPLES // 10  
 sess.run(tf.initialize\_all\_variables())  
 *#print "------Attemtion {0}------".format(z)* for i in range(epochs):  
 if ((k + 1) \* batch\_size >= N\_EXAMPLES):  
 k = 0  
 np.random.shuffle(inx\_seq)  
  
 batch\_data = train\_set[inx\_seq[k \* batch\_size:(k + 1) \* batch\_size]]  
 batch\_label = train\_set\_labels[inx\_seq[k \* batch\_size:(k + 1) \* batch\_size]]  
 k += 1  
  
 if (i % 100 == 0):  
 trainning\_accuracy = sess.run(accuracy, feed\_dict={x: batch\_data, y: batch\_label, keep\_prob: 0.5})  
 *# print i, trainning\_accuracy* sess.run(train\_step, feed\_dict={x: batch\_data, y: batch\_label, keep\_prob: 0.5})  
  
 if (testing):  
 acc = sess.run(accuracy, feed\_dict={x: input\_data[test\_inx], y: input\_label[test\_inx], keep\_prob: 1.})  
 list\_accuracy = np.hstack((list\_accuracy, acc))  
 *#print "Attemption {0}: Accuracy = {1}".format(z, acc)  
 #if (testing):  
 #print "Overall: {0}".format(np.mean(list\_accuracy))* if (not testing):  
 saver.save(sess, default\_saver\_path+name+".ckpt",)  
  
@app.route('/<path>')  
def start(path):  
 *#Initialising\_LatinClassifierNet()  
 #Learn("test", N\_ATTEMPTIONS=20)  
 #Learn("complete\_training", N\_ATTEMPTIONS=1, testing=False,epochs=2000)* saver = tf.train.Saver()  
 saver.restore(save\_path="./logs/complete\_training.ckpt",sess=sess)  
 *#Initialising\_LatinClassifierNet(restore=True,ckpt\_name="complete\_training")* im = Image.open(path)  
 a = np.array(im)  
 a = a.flatten()  
 a = a.reshape(1,2352)  
 *#print sess.run(tf.nn.softmax((y\_hat)), feed\_dict={x:a, keep\_prob:1.})* ans = sess.run((tf.argmax(tf.nn.softmax(y\_hat),1)), feed\_dict={x:a, keep\_prob:1.})  
 return (get\_letter(ans[0]))

**Приложение 1**

**Список используемой литературы**

* 1. ГОСТ 19.101-77 Виды программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
  2. ГОСТ 19.102-77 Стадии разработки. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
  3. ГОСТ 19.103-77 Обозначения программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
  4. ГОСТ 19.105-78 Общие требования к программным документам. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
  5. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
  6. ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
  7. ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
  8. ГОСТ 19.301-79 Программа и методика испытаний. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
  9. Статья «Методы оптимизации нейронных сетей», Павел Садовников [Электронный ресурс]//  
     URL: <https://habrahabr.ru/post/318970/> (Дата обращения: 09.05.2017, режим доступа: свободный).
  10. Статья «Convolutional Neural Networks for Visual Recognition», Stanford University [Электронный ресурс]//

URL: <http://cs231n.github.io/convolutional-networks/> (Дата обращения: 09.05.2017, режим доступа: свободный).

**ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Изм. | Номера листов (страниц) | | | | Всего листов (страниц) в документе | № документа | Входящий № сопроводительного документа и дата | Подпись | Дата |
| измененных | замененных | новых | аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |