Министерство образования и науки Российской Федерации
Федеральное государственное образовательное учреждение высшего образования
«Национальный исследовательский университет «МЭИ»
Институт ИВТ
Кафедра ПМИИ

Отчет по курсовой работе

Тема: “Создание нейронной сети для классификации животных”

Выполнил
Студент группы А-05-19
Желтиков Александр Алексеевич
Проверил Михайлов Илья Сергеевич

Москва 2021

Введение

Что такое нейронные сети

Нейронные сети, известные также как искусственные нейронные сети (ANN) или смоделированные нейронные сети (SNN), являются подмножеством алгоритмов машинного обучения и служат основой для алгоритмов глубокого обучения. Понятие «нейронные сети» возникло при попытке смоделировать процессы, происходящие в человеческом мозге при передаче сигналов между биологическими нейронами.

Искусственные нейронные сети (ANN) состоят из образующих слои узлов: слой входных данных, один или несколько скрытых слоев и слой выходных данных. Каждый узел (искусственный нейрон) связан с другими узлами с определенным весом и пороговым значением. Если вывод какого-либо узла превышает пороговое значение, то этот узел активируется и отправляет данные на следующий уровень сети. В противном случае данные на следующий уровень сети не передаются.

Визуальная диаграмма нейронных сетей прямого распространения, состоящая из слоя входных данных, скрытых слоев и слоя выходных данных

Для обучения и постепенного повышения точности нейронных сетей применяются обучающие данные. При достижении требуемой точности алгоритмы обучения превращаются в мощные инструменты для вычислений и искусственного интеллекта, что позволяет использовать их для классификации и кластеризации данных с высокой скоростью. Задачи из области распознавания речи или изображений можно выполнить за несколько минут, а не за несколько часов, как при распознавании вручную. Одной из наиболее известных нейронных сетей является алгоритм поиска Google.

Преимущество языка C# , для создания многооконных приложений

Коротко о языке C#

C# (произносится си шарп) — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота, как язык разработки приложений для платформы Microsoft.NET Framework.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, переменные, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Преимущества языка C#

1. Это Объектно-ориентированный язык программирования (Далее будет использовано сокращения - ЯП), что дает большое преимущество при разработке приложений
2. Автоматическая «сборка мусора». Это значит, что нам в большинстве случаев не придётся заботиться об освобождении памяти.
3. Большое количество готовых программных конструкций, которые помогают разбирать и понимать написанный код
4. Достаточно низкий порог вхождения, так как C# имеет много общего с другими ЯП. То есть перейти на данный ЯП с любого другого очень просто.

Формирование требований к ПО

Удобство использования ПО

Самое главное при разработке ПО помнить , что не только разработчик будет им пользоваться , но и потребитель данного ПО, поэтому нельзя нагромождать интерфейс сложными и не нужными вещами. В конечной версии продукта , всё должно быть просто , красиво и понятно.

Качество работы ПО

При использовании потребителем ПО , разработчик должен убедиться в том , что в конечном продукте отсутствуют какие либо баги , глюки , недоработки.

Скорость работы ПО

Так как данное по включает в себя нейронную сеть , которая уже обучена, разработчик должен сконцентрировать своё внимание на скорости работы конечного продукта , а именно многооконного приложения.

Простота реализации ПО

При передаче исходного кода ПО другим разработчикам (для обновления , добавления новых “плюшек” , исправления каких-либо непредвиденных ошибок) , он должен быть простым в написании , то есть без использования каких либо пользовательских (самонаписанных) библиотек

Ход создания ПО

Разработка Нейронной Сети

Знакомство с keras и tensorflow

Для того что бы написать простую и рабочую нейронную сеть мы будем использовать две подключаемые к python библиотеки: Keras и TensorFlow.

Keras — это библиотека глубокого обучения, представляющая из себя высокоуровневый API, написанный на Python и способный работать поверх TensorFlow, Theano или CNTK. Он был разработан с расчетом на быстрое обучение. Способность переходить от гипотез к результаты с наименьшими временными затратами является ключом к проведению успешных исследований.

TensorFlow — Открытая программная библиотека для машинного обучения, разработанная компанией Google для решения задач построения и тренировки нейронной сети с целью автоматического нахождения и классификации образов, достигая качества человеческого восприятия. Применяется как для исследований, так и для разработки собственных продуктов Google. Основной API для работы с библиотекой реализован для Python, также существуют реализации для R, C Sharp, C++, Haskell, Java, Go и Swift.

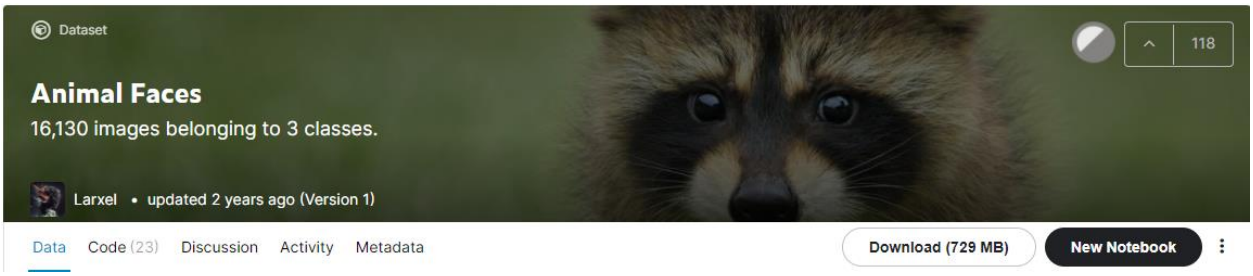
Подключение библиотек

Для того что бы использовать нужные библиотеки , сначала подключим их:

```
import os
from sys import exit
import numpy as np
import time
import matplotlib.pyplot as plt
import keras, tensorflow as tf
from keras.models import Model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.layers import Input, ReLU, Dense, Flatten, Reshape , Conv2D , MaxPooling2D
from keras.layers import Dropout, BatchNormalization
```

Загрузка изображений

Что бы у нас были данные для обучения нашей модели , загрузим их с сайта [Kaggle]



Далее сформируем набор изображений , которым мы сможем обучать нейронную сеть. Для этого используем функцию ImageDataGenerator и flow_from_directory. Немного информации про данные функции:

- ImageDataGenerator: Генерация пакетов данных тензорного изображения с дополнением данных в реальном времени. Данные будут зациклены (в пакетах).
- flow_from_directory: Создает набор изображений с заданными типами , которые берет в определенной директории

Тренировочный набор

```
image_generate = ImageDataGenerator(rescale=1./255)

train_data = image_generate.flow_from_directory(
    batch_size = batch,
    directory = train_path,
    shuffle = True,
    target_size = (img_shape , img_shape)
                                                )
```

```
Found 14630 images belonging to 3 classes.
```

Не забываем сравнить контрольные суммы изображений

```
image_generate = ImageDataGenerator(rescale=1./255)

val_data = image_generate.flow_from_directory(
    batch_size = batch,
    directory = val_path,
    shuffle = False,
    target_size = (img_shape , img_shape)
                                                )
```

Создание модели НС

После создания наборов данных, перейдем к проектированию нейронной сети.

```
input_shape = (img_shape, img_shape , 3)

inp = Input(shape = input_shape)
x = inp


x = Conv2D(32 , (5,5) , padding='same' , activation='relu')(x)
x = MaxPooling2D(pool_size=(2,2) , strides=2)(x)
x = Conv2D(64 , (5,5) , padding='same' , activation='relu')(x)
x = MaxPooling2D(pool_size=(2,2) , strides=2)(x)
x = Flatten()(x)
x = Dense(units = 256, activation = 'relu')(x)
x = Dropout(rate=0.2)(x)
x = Dense(units = 32, activation = 'relu')(x)

output = Dense(3, activation = 'softmax')(x)
model = Model(inputs = inp, outputs = output)
```

Здесь мы спроектировали нейронную сеть , используя свёрточные слои.

Теперь нам нужно собрать модель.

```
model.compile(optimizer = 'Adam',
              loss = 'mse',
              metrics = ['accuracy'])
```

```
model.summary()
```

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 150, 150, 3)]     0

 conv2d_14 (Conv2D)          (None, 150, 150, 32)      2432

 max_pooling2d_14 (MaxPoolin  (None, 75, 75, 32)       0
 g2D)

 conv2d_15 (Conv2D)          (None, 75, 75, 64)        51264

 max_pooling2d_15 (MaxPoolin  (None, 37, 37, 64)       0
 g2D)

 flatten_5 (Flatten)         (None, 87616)             0

 dense_11 (Dense)            (None, 256)               22429952

 dropout_5 (Dropout)         (None, 256)               0

 dense_12 (Dense)            (None, 32)                8224

 dense_13 (Dense)            (None, 3)                 99

=================================================================
Total params: 22,491,971
Trainable params: 22,491,971
Non-trainable params: 0
_________________________________________________________________
```

Обучение НС

Перейдем к обучению. Так как при обучении я использовал CPU , то это заняло у меня достаточно большое количество времени. За 40 эпох она достигла точности в 99.25% на тренировочном наборе изображений , и 94.53% на валидационном наборе данных.

```
EPOCHS = 40
history = model.fit(
    train_data,
    steps_per_epoch=int(np.ceil(num_all_train / float(batch))),
    epochs=EPOCHS,
    validation_data=val_data,
    validation_steps=int(np.ceil(num_all_val / float(batch)))
)
```

```
Epoch 1/40
147/147 [==============================] - 387s 3s/step - loss: 0.0186 - accuracy: 0.9649 - val_loss: 0.0285 - val_accuracy: 0.9440
Epoch 2/40
147/147 [==============================] - 387s 3s/step - loss: 0.0130 - accuracy: 0.9751 - val_loss: 0.0284 - val_accuracy: 0.9460
```

.....

```
Epoch 37/40
147/147 [==============================] - 380s 3s/step - loss: 0.0028 - accuracy: 0.9948 - val_loss: 0.0281 - val_accuracy: 0.9533
Epoch 38/40
147/147 [==============================] - 381s 3s/step - loss: 0.0026 - accuracy: 0.9953 - val_loss: 0.0285 - val_accuracy: 0.9507
Epoch 39/40
147/147 [==============================] - 380s 3s/step - loss: 0.0034 - accuracy: 0.9938 - val_loss: 0.0400 - val_accuracy: 0.9367
Epoch 40/40
147/147 [==============================] - 380s 3s/step - loss: 0.0041 - accuracy: 0.9925 - val_loss: 0.0333 - val_accuracy: 0.9453
```

Сохранение готовой модели

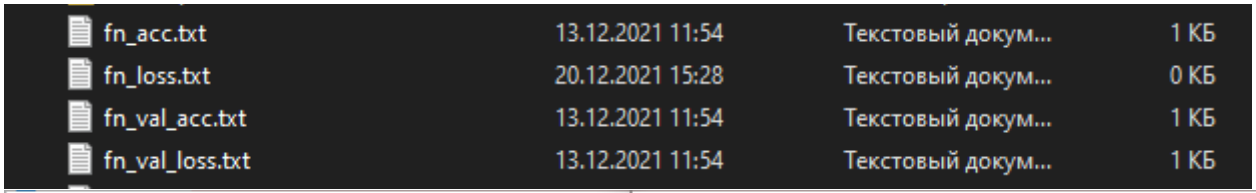
Для того что бы пользоваться в будущем нашей НС, нужно обязательно сохранить её:

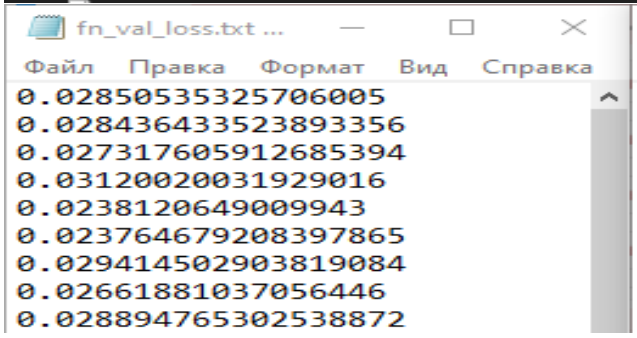
```
print('Модель сохранена в файле', model_path)
model.save(model_path)
history = history.history
```

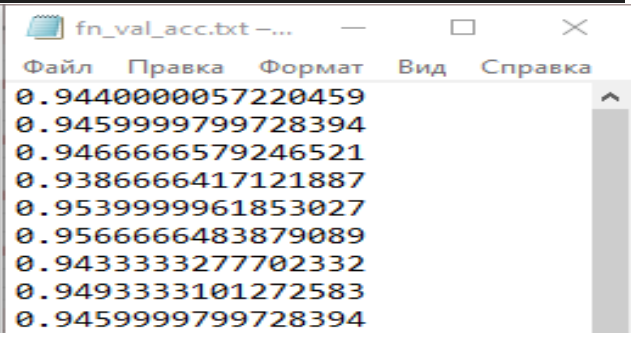
Сохранение потерь и точности обучения

Далее , что бы мы могли построить графики в многооконном приложении , сохраним все переменные в отдельных файлах.

```
with open('/Users/Alexander/Desktop/Курсач ПИ/fn_loss.txt', 'w') as output:
    for val in history['loss']: output.write(str(val) + '\n')
with open('/Users/Alexander/Desktop/Курсач ПИ/fn_acc.txt', 'w') as output:
    for val in history['accuracy']: output.write(str(val) + '\n')
with open('/Users/Alexander/Desktop/Курсач ПИ/fn_val_loss.txt', 'w') as output:
    for val in history['val_loss']: output.write(str(val) + '\n')
with open('/Users/Alexander/Desktop/Курсач ПИ/fn_val_acc.txt', 'w') as output:
    for val in history['val_accuracy']: output.write(str(val) + '\n')
```



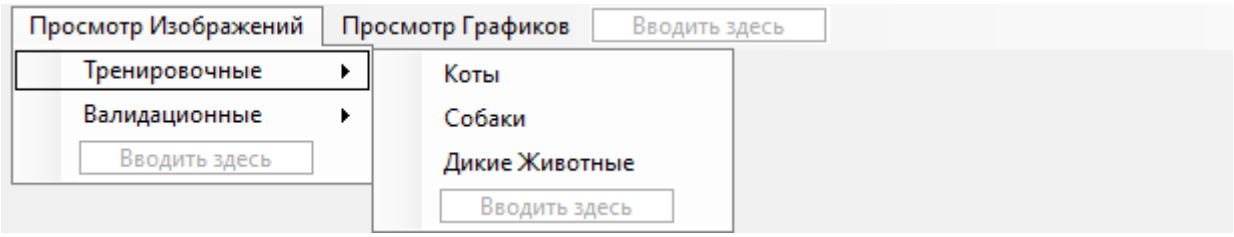




Разработка приложения C#

Реализация просмотра изображений

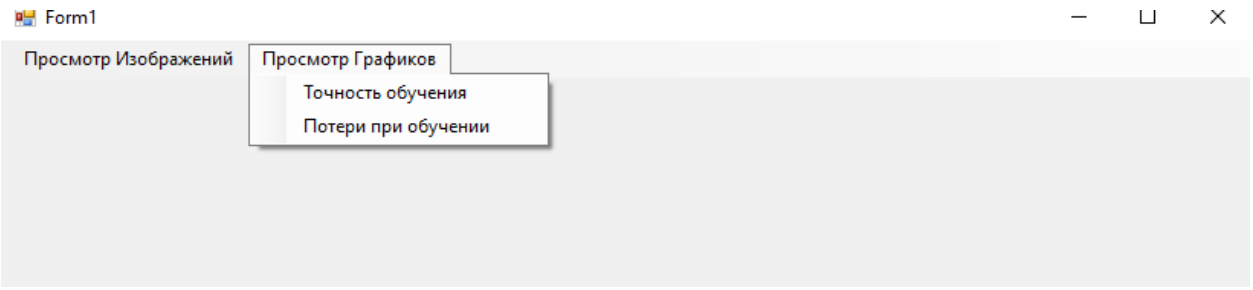
Для того, чтобы пользователю было удобно просматривать тренировочные и валидационные изображения, создадим несколько пунктов меню:



Их реализация достаточно проста, так для всего этого есть встроенные функции Visual Studio.

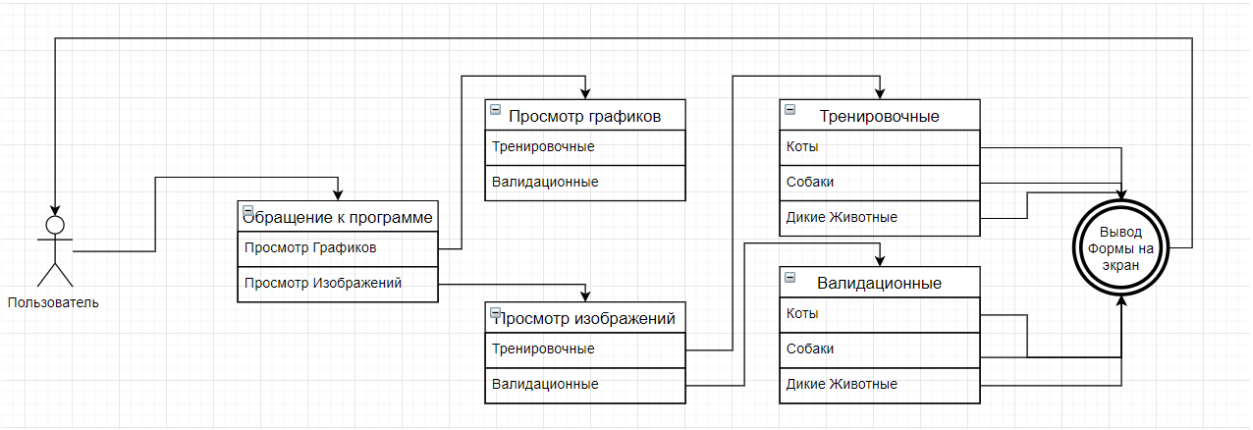
Реализация просмотра графиков

Для выполнения данного пункта мы будем использовать встроенные функции языка, и так же создадим пункты меню для доступа к ним.



Далее в разделе тестирование будет показана работа каждого пункта меню.

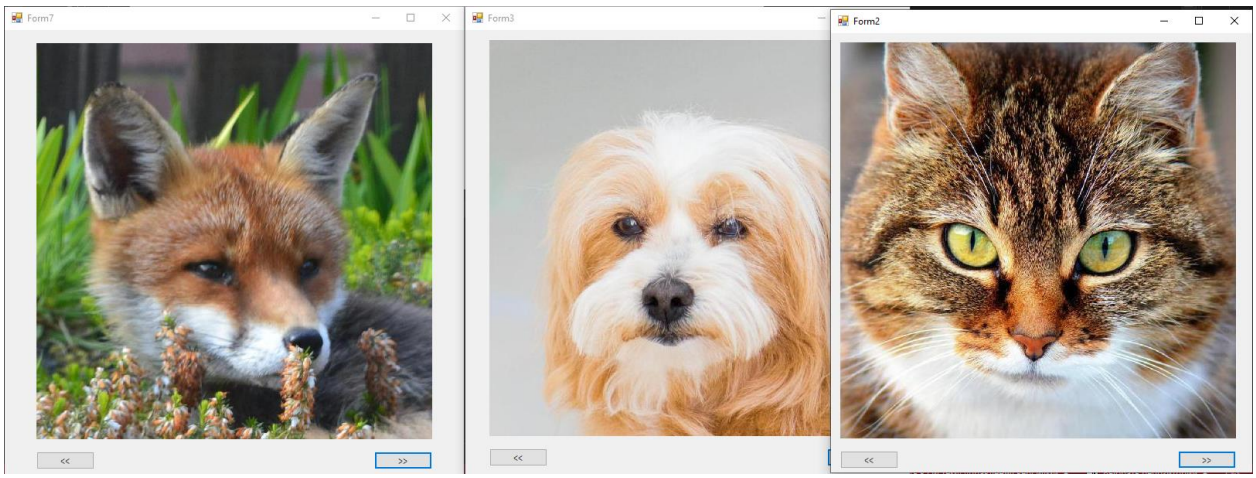
Эскиз проекта (Схема работы пользователя с программой)



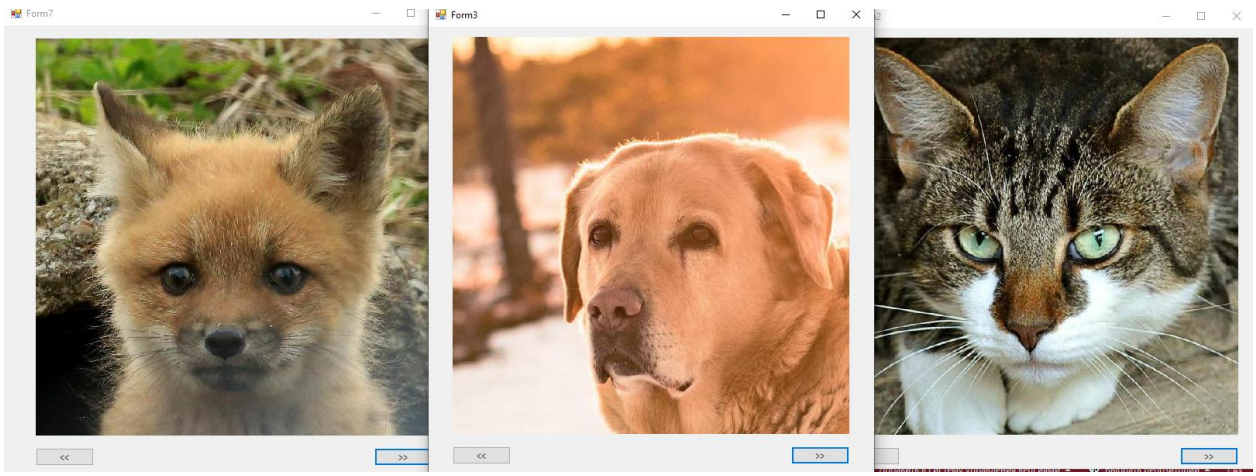
Данная схема отображает возможности пользователя при использовании меню главного окна.

Тестирование ПО

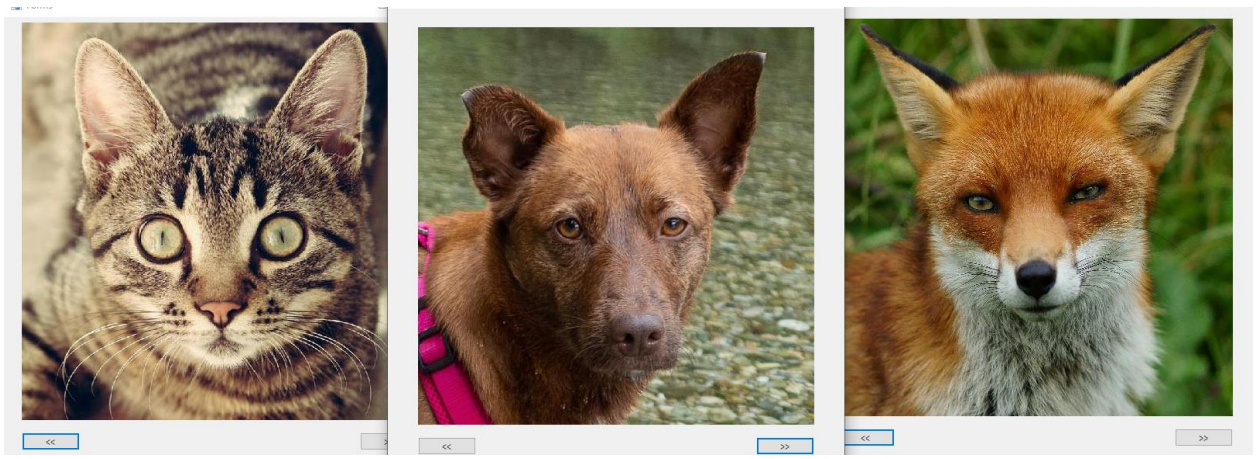
Проверим просмотр изображений, которые использовались для обучения НС



Так же там предусмотрено пролистывание для удобного просмотра изображений.

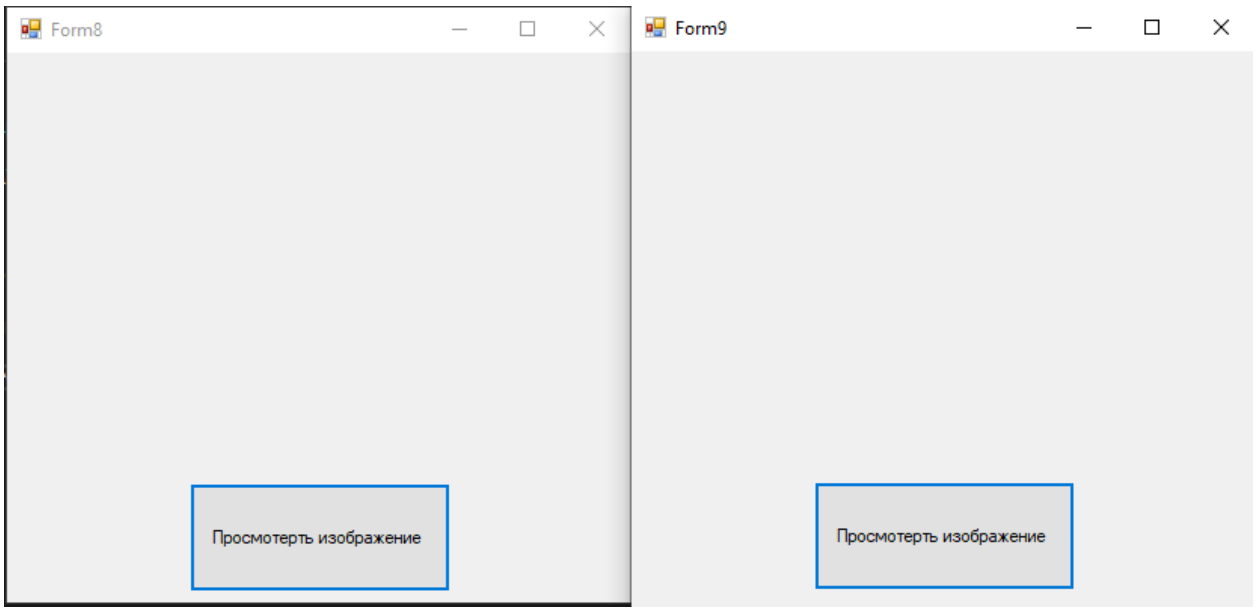


Теперь рассмотрим просмотр валидационных изображений.

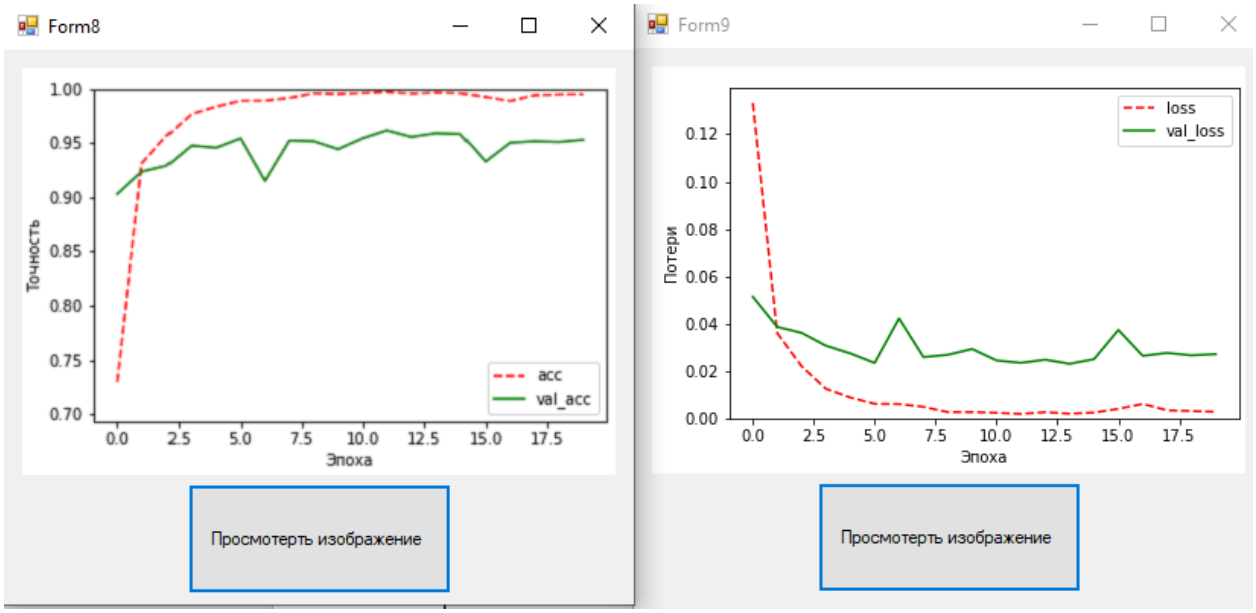


При выборе данных пунктов меню так же предусмотрено пролистывание изображений.

Теперь рассмотрим просмотр графиков обучения нашей НС.



Для того что бы пользователь смог просмотреть их, предусмотрена специальная кнопка.



Проведем исследование точности при изменении размеров пакетов данных:

Размер выборки данных		**Качество классификации**	
Обучающая	***Тестовая***	***Обучающая***	***Тестовая***
14630	1500	99.47%	95.27%
13166	1350	99.48%	94.22%
10971	1125	99.44%	92.80%
7340	750	98.98%	91.73%
3670	375	99.05%	89.07%
1840	195	99.35%	86.67%

Несмотря на то, что мы уменьшили размер выборки данных примерно на 87%, НС показывает хорошее качество классификации (более 80% верно определенных изображений). При это мы получили достаточно высокую скорость обучения – 51-53сек за эпоху, при изначальных данных скорость была около 389-391сек за эпоху. НС прекрасно работает с количеством данных равным 90% от изначального.

Keras и TensorFlow – великолепные библиотеки для создания простых и сложных НС, в них есть все нужные функции, которые облегчают проектирование нейронной сети. Так же документация, переведенная на несколько популярных языков.

Код программы для быстрого ознакомления

Реализация НС

```
import os
from sys import exit
import numpy as np
import time
import matplotlib.pyplot as plt
import keras, tensorflow as tf
from keras.models import Model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.layers import Input, ReLU, Dense, Flatten, Reshape , Conv2D , MaxPooling2D
from keras.layers import Dropout, BatchNormalization

data_path = '/Users/Alexander/Desktop/Курсач ПИ/dataset/'
model_path = '/Users/Alexander/Desktop/Курсач ПИ/' + 'model.pt'

train_path = (data_path + 'train/')
val_path = (data_path + 'val/')

cat_train = (train_path + 'cat/')
dog_train = (train_path + 'dog/')
wild_train = (train_path + 'wild/')

cat_val = (val_path + 'cat/')
dog_val = (val_path + 'dog/')
wild_val = (val_path + 'wild/')

num_cat_train = len(os.listdir(cat_train))
num_dog_train = len(os.listdir(dog_train))
num_wild_train = len(os.listdir(wild_train))

num_cat_val = len(os.listdir(cat_val))
num_dog_val = len(os.listdir(dog_val))
num_wild_val = len(os.listdir(wild_val))

num_all_train = num_cat_train + num_dog_train + num_wild_train
num_all_val = num_cat_val + num_dog_val + num_wild_val

img_shape = 150

batch = 100

def plotImg(image):
    fig, axs = plt.subplots(1, 5, figsize=(20,20))
    axs = axs.flatten()
    for img, ax in zip(image, axs):
        ax.imshow(img)
    plt.tight_layout()
    plt.show()

image_generate = ImageDataGenerator(rescale=1./255)

train_data = image_generate.flow_from_directory(
    batch_size = batch,
    directory = train_path,
    shuffle = True,
    target_size = (img_shape , img_shape)
                                                )

```

```
image_generate = ImageDataGenerator(rescale=1./255)

val_data = image_generate.flow_from_directory(
    batch_size = batch,
    directory = val_path,
    shuffle = False,
    target_size = (img_shape , img_shape)
                                                )

sample_trainImg, _ = next(train_data)
sample_valImg , _ = next(val_data)

input_shape = (img_shape, img_shape , 3)

inp = Input(shape = input_shape)
x = inp

x = Conv2D(32 , (5,5) , padding='same' , activation='relu')(x)
x = MaxPooling2D(pool_size=(2,2) , strides=2)(x)
x = Conv2D(64 , (5,5) , padding='same' , activation='relu')(x)
x = MaxPooling2D(pool_size=(2,2) , strides=2)(x)
x = Flatten()(x)
x = Dense(units = 256, activation = 'relu')(x)
x = Dropout(rate=0.2)(x)
x = Dense(units = 32, activation = 'relu')(x)

output = Dense(3, activation = 'softmax')(x)
model = Model(inputs = inp, outputs = output)

model.compile(optimizer = 'Adam',
              loss = 'mse',
              metrics = ['accuracy'])

model.summary()

EPOCHS = 40
history = model.fit(
    train_data,
    steps_per_epoch=int(np.ceil(num_all_train / float(batch))),
    epochs=EPOCHS,
    validation_data=val_data,
    validation_steps=int(np.ceil(num_all_val / float(batch)))
)

print('Модель сохранена в файле', model_path)
model.save(model_path)
history = history.history

with open('/Users/Alexander/Desktop/Курсач ПИ/fn_loss.txt', 'w') as output:
    for val in history['loss']: output.write(str(val) + '\n')
with open('/Users/Alexander/Desktop/Курсач ПИ/fn_acc.txt', 'w') as output:
    for val in history['accuracy']: output.write(str(val) + '\n')
with open('/Users/Alexander/Desktop/Курсач ПИ/fn_val_loss.txt', 'w') as output:
    for val in history['val_loss']: output.write(str(val) + '\n')
with open('/Users/Alexander/Desktop/Курсач ПИ/fn_val_acc.txt', 'w') as output:
    for val in history['val_accuracy']: output.write(str(val) + '\n')

def one_plot(n, y_lb, loss_acc, val_loss_acc):
    plt.subplot(1, 2, n)
    if n == 1:
        lb, lb2 = 'loss', 'val_loss'
        yMin = 0
        yMax = 1.05 * max(max(loss_acc), max(val_loss_acc))
    else:
        lb, lb2 = 'acc', 'val_acc'
        yMin = min(min(loss_acc), min(val_loss_acc))
        yMax = 1.0
    plt.plot(loss_acc, color = 'r', label = lb, linestyle = '--')
    plt.plot(val_loss_acc, color = 'g', label = lb2)
```

```
    plt.ylabel(y_lb)
    plt.xlabel('Эпоха')
    plt.ylim([0.95 * yMin, yMax])
    plt.legend()

plt.figure(figsize=(15, 4))
plt.subplots_adjust(wspace=0.5)
one_plot(1, 'Потери', history['loss'], history['val_loss'])
plt.savefig('loss.png')

plt.figure(figsize=(15, 4))
plt.subplots_adjust(wspace=0.5)
one_plot(2, 'Точность', history['accuracy'], history['val_accuracy'])
plt.savefig('acc.png')
```

Реализация приложения на C#

```
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;

namespace CourseWork_Sharp
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void котыToolStripMenuItem_Click(object sender, EventArgs e)
        {
            catTrn traincat = new catTrn();
            traincat.Show();
        }

        private void собакиToolStripMenuItem_Click(object sender, EventArgs e)
        {
            dogTrn traindog = new dogTrn();
            traindog.Show();
        }

        private void дикиеЖивотныеToolStripMenuItem_Click(object sender, EventArgs e)
        {
            wildTrn trainWild = new wildTrn();
            trainWild.Show();
        }

        private void котыToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            catVal traincat = new catVal();
            traincat.Show();
        }

        private void собакиToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            dogVal traindog = new dogVal();
            traindog.Show();
        }

        private void дикиеЖивотныеToolStripMenuItem1_Click(object sender, EventArgs e)
```

```
        {
            wildVal trainwild = new wildVal();
            trainwild.Show();
        }

        private void тренировочныеToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            graphTrn graph = new graphTrn();
            graph.Show();
        }

        private void валидационныеToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            graphVal graph = new graphVal();
            graph.Show();
        }
    }
}

//CatTrn

public partial class catTrn : Form
    {

        static string link = "C:\\Users\\Alexander\\Desktop\\Курсач
   ПИ\\dataset\\train\\cat\\pixabay_cat_";
        static int count = 2;
        static string number = "000002";
        public catTrn()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            //  >>
            if (Convert.ToInt32(number) <= 4834)
            {
                string foo;
                pictureBox1.ImageLocation = link + number + ".jpg";
                count++;
                foo = Convert.ToString(count);
                number = number.Remove(number.Length - foo.Length);
                number += foo;
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            if (Convert.ToInt32(number) >= 2)
             {
                 string foo;
                 pictureBox1.ImageLocation = link + number + ".jpg";
                 count--;
                 foo = Convert.ToString(count);
                 number = number.Remove(number.Length - foo.Length);
                 number += foo;
             }
         }
    }

// CatVal

public partial class catVal : Form
    {
        static string link = "C:\\Users\\Alexander\\Desktop\\Курсач
   ПИ\\dataset\\val\\cat\\pixabay_cat_";
         static int count = 2;
         static string number = "000002";
```

```
118.         public catVal()
119.         {
120.             InitializeComponent();
121.         }
122.
123.         private void button1_Click(object sender, EventArgs e)
124.         {
125.             //  >>
126.             if (Convert.ToInt32(number) <= 4834)
127.             {
128.                 string foo;
129.                 pictureBox1.ImageLocation = link + number + ".jpg";
130.                 count++;
131.                 foo = Convert.ToString(count);
132.                 number = number.Remove(number.Length - foo.Length);
133.                 number += foo;
134.             }
135.         }
136.
137.         private void button2_Click(object sender, EventArgs e)
138.         {
139.             if (Convert.ToInt32(number) >= 2)
140.             {
141.                 string foo;
142.                 pictureBox1.ImageLocation = link + number + ".jpg";
143.                 count--;
144.                 foo = Convert.ToString(count);
145.                 number = number.Remove(number.Length - foo.Length);
146.                 number += foo;
147.             }
148.         }
149.
150.     }
151.
152. // DogTrn
153.
154. public partial class dogTrn : Form
155.     {
156.         static string link = "C:\\Users\\Alexander\\Desktop\\Курсач
    ПИ\\dataset\\train\\dog\\flickr_dog_";
157.         static int count = 2;
158.         static string number = "000002";
159.         public dogTrn()
160.         {
161.             InitializeComponent();
162.         }
163.
164.         private void button1_Click(object sender, EventArgs e)
165.         {
166.             //  >>
167.             if (Convert.ToInt32(number) <= 4834)
168.             {
169.                 string foo;
170.                 pictureBox1.ImageLocation = link + number + ".jpg";
171.                 count++;
172.                 foo = Convert.ToString(count);
173.                 number = number.Remove(number.Length - foo.Length);
174.                 number += foo;
175.             }
176.         }
177.
178.         private void button2_Click_1(object sender, EventArgs e)
179.         {
180.             if (Convert.ToInt32(number) >= 2)
181.             {
182.                 string foo;
183.                 pictureBox1.ImageLocation = link + number + ".jpg";
184.                 count--;
185.                 foo = Convert.ToString(count);
```

```
                number = number.Remove(number.Length - foo.Length);
                number += foo;
            }
        }
    }

// DogVal

 public partial class dogVal : Form
    {
        static string link = "C:\\Users\\Alexander\\Desktop\\Курсач ПИ\\dataset\\val\\dog\\flickr_dog_";
        static int count = 2;
        static string number = "000043";
        public dogVal()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            //  >>
            if (Convert.ToInt32(number) <= 4834)
            {
                string foo;
                pictureBox1.ImageLocation = link + number + ".jpg";
                count++;
                foo = Convert.ToString(count);
                number = number.Remove(number.Length - foo.Length);
                number += foo;
            }
        }


        private void button2_Click_1(object sender, EventArgs e)
        {
            if (Convert.ToInt32(number) >= 2)
            {
                string foo;
                pictureBox1.ImageLocation = link + number + ".jpg";
                count--;
                foo = Convert.ToString(count);
                number = number.Remove(number.Length - foo.Length);
                number += foo;
            }
        }
    }

// wildTrn

 public partial class wildTrn : Form
    {
        static string link = "C:\\Users\\Alexander\\Desktop\\Курсач ПИ\\dataset\\train\\wild\\flickr_wild_";
        static int count = 2;
        static string number = "000002";


        public wildTrn()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            //  >>
            if (Convert.ToInt32(number) <= 4834)
            {
                string foo;
```

```
253.                 pictureBox1.ImageLocation = link + number + ".jpg";
254.                 count++;
255.                 foo = Convert.ToString(count);
256.                 number = number.Remove(number.Length - foo.Length);
257.                 number += foo;
258.             }
259.         }
260.
261.         private void button2_Click_1(object sender, EventArgs e)
262.         {
263.             if (Convert.ToInt32(number) >= 2)
264.             {
265.                 string foo;
266.                 count--;
267.                 pictureBox1.ImageLocation = link + number + ".jpg";
268.                 foo = Convert.ToString(count);
269.                 number = number.Remove(number.Length - foo.Length);
270.                 number += foo;
271.             }
272.         }
273.     }
274.
275. // WildVal
276.
277. public partial class wildVal : Form
278.     {
279.         static string link = "C:\\Users\\Alexander\\Desktop\\Курсач
    ПИ\\dataset\\val\\wild\\flickr_wild_";
280.         static int count = 2;
281.         static string number = "000057";
282.
283.
284.         public wildVal()
285.         {
286.             InitializeComponent();
287.         }
288.
289.         private void button1_Click(object sender, EventArgs e)
290.         {
291.             //  >>
292.             if (Convert.ToInt32(number) <= 4834)
293.             {
294.                 string foo;
295.                 pictureBox1.ImageLocation = link + number + ".jpg";
296.                 count++;
297.                 foo = Convert.ToString(count);
298.                 number = number.Remove(number.Length - foo.Length);
299.                 number += foo;
300.             }
301.         }
302.
303.         private void button2_Click(object sender, EventArgs e)
304.         {
305.             if (Convert.ToInt32(number) >= 2)
306.             {
307.                 string foo;
308.                 pictureBox1.ImageLocation = link + number + ".jpg";
309.                 count--;
310.                 foo = Convert.ToString(count);
311.                 number = number.Remove(number.Length - foo.Length);
312.                 number += foo;
313.             }
314.         }
315.
316.
317.     }
318.
319. //graphTrn
320. private void button1_Click(object sender, EventArgs e)
```

```
        {
            string pass = "C:\\Users\\Alexander\\Desktop\\Курсач ПИ\\acc.png";

            pictureBox1.ImageLocation = pass;
        }

//graphVal
private void button1_Click(object sender, EventArgs e)
        {
            string pass = "C:\\Users\\Alexander\\Desktop\\Курсач ПИ\\loss.png";

            pictureBox1.ImageLocation = pass;
        }
```

Используемая литература

1. Документация библиотеки Keras: https://keras.io
2. Библиотека dataset'ов : https://www.kaggle.com/andrewmvd/animal-faces
3. Источник информации о нейронных сетях: https://www.ibm.com/ru-ru/cloud/learn/neural-networks
4. О языке C#: https://ru.wikipedia.org/wiki/C_Sharp