

Государственное общеобразовательное учреждение
Ярославской области
«Средняя школа «Провинциальный колледж»

Пятая региональная конференция школьников «Лабиринты науки»

Определение наиболее вероятно наследуемого цвета глаз у человека с помощью нейронных сетей

Секция: биотехнология

Выполнена ученицей
11 информационно-
технологического класса ГОУ
ЯО Средней школы
«Провинциальный колледж»
**Чвановой Владиславой
Олеговной**

Научный руководитель —
учитель математики и
информатики
ГОУ ЯО Средней школы
«Провинциальный колледж»
Румянцева Мария Николаевна

Ярославль, 2024

Оглавление

ВВЕДЕНИЕ.....	3
1. ПРИНЦИПЫ НАСЛЕДОВАНИЯ ЦВЕТОВ ГЛАЗ. НЕЙРОННЫЕ СЕТИ	5
1.1. ПРИНЦИП НАСЛЕДОВАНИЯ ЦВЕТА ГЛАЗ	5
1.2. НЕЙРОННЫЕ СЕТИ.....	5
2. РЕАЛИЗАЦИЯ ПРОГРАММНОГО ПРОДУКТА.....	6
2.1. СРЕДА РАЗРАБОТКИ	6
2.2. ДАТАСЕТ	6
2.3. РАБОТА НЕЙРОННЫХ СЕТЕЙ	7
2.4. РЕЗУЛЬТАТ РАБОТЫ НЕЙРОННЫХ СЕТЕЙ	8
2.5. РАБОТА ПРОГРАММЫ.....	8
ЗАКЛЮЧЕНИЕ	10
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ И ИСТОЧНИКОВ.....	11
ПРИЛОЖЕНИЯ	12
ПРИЛОЖЕНИЕ 1. ИСКУССТВЕННЫЙ НЕЙРОН	12
ПРИЛОЖЕНИЕ 2. СВЕРТОЧНАЯ НЕЙРОННАЯ СЕТЬ	12
ПРИЛОЖЕНИЕ 3. НЕЙРОННАЯ СЕТЬ ДЕТЕКЦИИ	12
ПРИЛОЖЕНИЕ 4. КОД АРХИТЕКТУРЫ НЕЙРОННОЙ СЕТИ ДЕТЕКЦИИ	13
ПРИЛОЖЕНИЕ 5. АРХИТЕКТУРА НЕЙРОННОЙ СЕТИ КЛАССИФИКАЦИИ.....	13
ПРИЛОЖЕНИЕ 6. СТРУКТУРА РАБОТЫ НЕЙРОННОЙ СЕТИ ДЕТЕКЦИИ И КЛАССИФИКАЦИИ.....	14
ПРИЛОЖЕНИЕ 7. ПРОЦЕНТ ВЕРНОГО ОПРЕДЕЛЕНИЯ ЦВЕТОВ ГЛАЗ НА ТЕСТОВОЙ И ТРЕНИРОВОЧНОЙ ВЫБОРКАХ	15
ПРИЛОЖЕНИЕ 8. ОПРЕДЕЛЕНИЕ МЕСТОПОЛОЖЕНИЯ ГЛАЗ И ИХ ЦВЕТА	16
ПРИЛОЖЕНИЕ 9. ПРОВЕРКА ДАННЫХ. ДИАЛОГ С ПОЛЬЗОВАТЕЛЕМ	16
ПРИЛОЖЕНИЕ 10. КОД ПРОГРАММЫ	17

Введение

Актуальность исследования: Установлено, что индивидуальные особенности человека передаются по наследству. А это значит, что дети могут наследовать от родителей и такие внешние признаки, как цвет глаз, кожи, волос, рост и другие отличительные черты. Изучение закономерности наследования давно занимает ученых, однако в современном мире нельзя точно предсказать, какие признаки будут наследоваться, так как в генетике велика вероятность неопределенности результатов, которая может возникнуть из-за различных мутаций в генах.

На данном этапе развития науки ученые выдвигают предположения о том, какие гены отвечают за наследование различных признаков, возможно ли вносить корректировки в гены, чтобы регулировать наследственность. Для решения подобных задач, анализа генетической информации можно использовать нейронные сети. Они появились в середине прошлого века, но уже активно применяются в генетике.

Наши нейронные сети и программа могут помочь уменьшить неопределенность знаний в генетике, определяя цвет глаз у родителей по фотографиям и предсказывая наиболее вероятно наследуемый цвет глаз у ребенка по данным его родителей.

Цель исследования: написать сверточную нейронную сеть, нейронную сеть детекции и программу, которые на основе закономерностей наследования будут определять наиболее вероятно наследуемый цвет глаз ребенка по данным его родителей.

Задачи исследования:

1. Изучить вопросы генетической наследственности возможных цветов глаз у людей.
2. Изучить структуру искусственных нейронных сетей.

3. Собрать датасет и подготовить его для обучения двух нейронных сетей.
4. Запрограммировать нейронную сеть детекции и обучить ее находить глаза человека на фотографиях.
5. Запрограммировать сверточную нейронную сеть и обучить ее распознавать цвет глаз человека.
6. Написать программу, позволяющую определять наиболее вероятный цвет глаз у ребёнка по данным его родителей и визуализирующую внешность ребенка.

Объект исследования - нейронные сети.

Предмет исследования - закономерность наследования цвета глаз у человека.

Методы исследования:

- анализ (используется для изучения литературы и источников по генетическому наследованию; литературы и источников по машинному обучению),
- сравнение (используется для сравнения наследуемости признаков),
- моделирование (используется для написания кода программы и нейронной сети),
- эксперимент (используется для подбора параметров нейронной сети),
- обобщение (используется для подведения итога работы программы и нейронной сети).

Продукт нашей исследовательской деятельности может быть востребован среди молодежи, а также в центрах планирования семьи.

1. Принципы наследования цветов глаз. Нейронные сети

1.1. Принцип наследования цвета глаз

«Наследование - передача генетической информации от одного поколения организмов к другому»¹. Одним из признаков общности генов является цвет глаз. При наследовании цвета глаз доминирующими являются темные цвета (карий, черный), а рецессивными – светлые (голубой, зеленый).²

1.2. Нейронные сети

Нейронная сеть — математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей — сетей нервных клеток живого организма.³ Искусственный нейрон (приложение 1) состоит из ядра (единицы обработки), нескольких дендритов (аналогично входам), и одного аксона (аналогично выходу).

Базовая нейронная сеть содержит три слоя нейронов: входной, скрытый, выходной. Нейронная сеть, которая имеет в своей архитектуре сверточные слои, называется сверточной, её структура представлена в приложении 2.

Нейронные сети классификации являются сверточными. Классификация – это задача, при которой по некоторому объекту, исходным данным, нужно предсказать класс объекта.

Нейронная сеть детекции - одна из разновидностей нейронных сетей, задачей которой является нахождение и распознавание объектов, структура представлена в приложении 3.

¹ Наследование (биология): [Электронный ресурс] // Википедия. URL: [https://ru.wikipedia.org/wiki/Наследование_\(биология\)](https://ru.wikipedia.org/wiki/Наследование_(биология)). (Дата обращения: 10.11.2024).

² Тейлор Д., Грин Н., Стаут У. Биология / Под ред. Р.Сопера. М., 2021. С.139.

³ Нейронная сеть: [Электронный ресурс] // Википедия. URL: https://ru.wikipedia.org/wiki/Нейронная_сеть. (Дата обращения: 10.11.2024).

2. Реализация программного продукта

2.1. Среда разработки

В ходе работы были написаны две нейронные сети: детекции и классификации. Они создавались в интерактивном блокноте Jupyter Notebook и Google Colaboratory – облачном сервисе, направленном на упрощение исследований в области машинного и глубокого обучения.

Для разметки изображений было использован LabelImg - инструмент для разметки изображений, имеющий готовые собранные файлы для Windows.

Для написания нейронных сетей был использован язык программирования Python, он отлично подходит для обработки больших объемов данных, а также обладает большим количеством библиотек, необходимых для создания нейронных сетей.

2.2. Датасет

Для обучения нейронной сети детекции был собран тренировочный датасет из 595 изображений лиц людей. Все они были размечены в LabelImg. А также тестовый датасет из 301 изображения.

Для обучения нейронной сети классификации был собран датасет из 2380 изображений глаз людей. Из них 1190 были получены благодаря применению аугментации - методике создания дополнительных данных из имеющихся данных.⁴ А также тестовый датасет, состоящий из 602 изображений.

В тренировочном датасете было: 192 изображения глаз зеленого цвета, 174 изображения глаз черного цвета, 406 изображений глаз серого цвета, 776 изображений глаз карего цвета, 832 изображения глаз голубого цвета.

⁴ Практика реализации нейронных сетей: [Электронный ресурс] // Викиконспекты. URL: https://neerc.ifmo.ru/wiki/index.php?title=Практики_реализации_нейронных_сетей. (Дата обращения: 10.11.2024).

В тестовом датасете было: 38 изображений глаз зеленого цвета, 31 изображение глаз черного цвета, 96 изображений глаз серого цвета, 279 изображений глаз карего цвета, 158 изображений глаз голубого цвета.

Неравномерность количества изображений разных цветов глаз обусловлена тем, что наиболее часто встречающийся цвет глаз – карий и голубой, а зеленый и черный считаются редкими.

Среди цветов глаз выделяют пять основных классов. Однако качество определения цвета зависит от различных внешних факторов, таких как освещение, а также цветовое восприятие. Так как серый цвет является производным от голубого, их можно объединить в один класс. Черный и карий цвет также можно объединить, так как эти цвета обусловлены большим количеством меланина в радужке. Следовательно, с научной точки зрения данное объединение возможно.

Наша нейронная сеть предсказывает принадлежность изображения к одному из трех цветовых классов: 0 – зеленый цвет и его оттенки, 1 – голубой цвет и его оттенки, 2 – карий цвет и его оттенки.

2.3. Работа нейронных сетей

Код архитектуры нейронной сети детекции представлен в приложении 4. Нейронная сеть детекции содержит семь сверточных слоев и один линейный. А также выходной слой. Скрытые слои (между входным и выходным) содержат 32, 64, 128 нейронов, выходной слой – 8 нейронов (по 4 координаты рамки на один глаз). Всего использовано девять слоев, так они дают оптимальный результат, что выявлено эмпирически. После сверток и в выходном слое использовалась функция активации Relu. В структуре сети также использовался Dropout со значениями 0,4 и 0,2. В работе был использован метод оптимизации Adam, так как он очень эффективен и сочетает в себе идеи нескольких других оптимизаторов; функция потерь – IoU.

В приложении 5 представлена архитектура нейронной сети классификации.

Нейронная сеть классификации состоит из трех сверточных слоев, трех линейных слоев и выходного слоя. Меньшее количество скрытых слоев не дает хороших результатов, а использование большего количества невозможно, так как ограничена производительность центрального процессора. Каждый из сверточных слоев содержит одну свертку, Maxpooling2D, Batchnormalization и Dropout со значением 0.2; функция активации – Relu; слои содержат 32, 64, 128 нейронов. Линейные слои содержат 256, 128, 64 нейрона; функция активации – Relu; применен Dropout со значением 0,2 и Batchnormalization. Выходной слой, имеет 3 нейрона на выходе, по количеству классов, функция активации - Softmax. Использован метод оптимизации Adam, функция потерь – Кросс-энтропия. В приложении 6 представлена структура работы нейронной сети детекции и классификации, весь исходный код выложен на GitHub и доступен по ссылке: <https://github.com/Vlada-chv/neural-networks>.

2.4. Результат работы нейронных сетей

В результате обучения нейронных сетей точность определения цвета глаз составила 67%. Приложение 7 содержит процент верного определения цветов глаз на тестовой и тренировочной выборках.

Тренировочный датасет состоит из 2380 изображений, из них верно определены 2240, то есть 94%. Тестовый датасет состоит из 602 изображений, из них верно определены 403, что составляет 67 процентов.

2.5. Работа программы

Программа была создана в облачном сервисе Google Colaboratory и интерактивном блокноте Jupyter Notebook.

Программа создает бинарный код картинки лица ребенка, а далее рисует её. Это необходимо для того, чтобы раскрашивать глаза ребенка в нужные цве-

та. Далее пользователь загружает свою фотографию, она обрабатывается нейронными сетями детекции и классификации: определяется местоположение глаз и происходит распознавание их цвета (приложение 8).

К фотографиям предъявляются определенные требования: изображение должно быть размером 1024 на 1024 пикселей. Лицо должно занимать 2/3 от всей фотографии и находится по центру. Глаза открыты, ничем не заслонены, без прищура.

После распознавания цвета идет сверка данных с пользователем (приложение 9). Если пользователь подтверждает корректность данных, программа обрабатывает информацию, раскрашивает картинку и выводит наиболее вероятно наследуемый цвет глаз у ребенка по данным его родителей. Код программы представлен в приложении 10.

Инструкция по использованию продукта:

1. Скачать файлы из папки и перенести их на свой Google Диск в основной каталог. Папка доступна по ссылке <https://drive.google.com/drive/folders/105Y6LW-5WPoggpXCtDwzaLlWaATyGrfa?usp=sharing>.
2. Загрузить в основной каталог фотографию мамы и папы, они должны называться «мама» и «папа», а также соответствовать стандарту.
3. Двойным кликом открыть файл «Result_drive».
4. Нажать на запуск первой ячейки и авторизироваться, это необходимо для подключения диска к Google Colaboratory.
5. По очереди активировать ячейки и дожждаться появления зеленой галочки возле каждой из них.
6. Подтвердить или подкорректировать данные (после запуска последней ячейки).
7. Дождаться результата. Фотография появится в боковом меню и будет называться «Результат».

Заключение

В ходе проделанной работы мы выяснили, что чаще всего наследуются доминантные (темные, в данном случае) цвета глаз.

Для решения поставленной цели мы выбрали сверточные нейронные сети, так как они отлично подходят для решения задач классификации. Нейронная сеть классификации предсказывает цвет глаз человека, нейронная сеть детекции находит и распознает глаза на изображении.

Мы собрали датасеты (объемом 2380 и 595 изображений), написали две нейронные сети и программу для определения наиболее вероятно наследуемого цвета глаз у человека. Точность определения цвета во многом зависит от внешних факторов, поэтому на тестовой выборке она составила 67%.

Нейронные сети применяются во всех областях науки, но в генетике их использование не так широко. В нашей работе мы применили нейронные сети для решения задач классификации в генетике. Это и составляет ценность данного исследования.

Цель нашего исследования была достигнута.

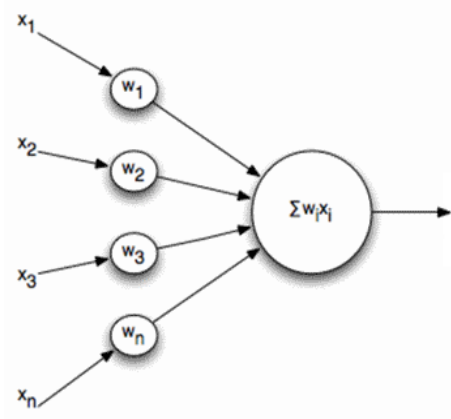
Исследование может быть продолжено, для этого необходимо собрать датасет на основе фотографий с ошибочными предсказаниями и обучить нейронную сеть на этих данных, чтобы повысить точность определения цвета.

Список использованной литературы и источников

1. Наследование (биология): [Электронный ресурс] // Википедия. URL: [https://ru.wikipedia.org/wiki/Наследование_\(биология\)](https://ru.wikipedia.org/wiki/Наследование_(биология)). (Дата обращения: 10.11.2024).
2. Нейронная сеть: [Электронный ресурс] // Википедия. URL: https://ru.wikipedia.org/wiki/Нейронная_сеть. (Дата обращения: 10.11.2024).
3. Практика реализации нейронных сетей: [Электронный ресурс] // Викиконспекты. URL: https://neerc.ifmo.ru/wiki/index.php?title=Практики_реализации_нейронных_сетей. (Дата обращения: 10.11.2024).
4. Тейлор Д., Грин Н., Стаут У. Биология / Под ред. Р.Сопера. М., 2021. 449 с.

Приложения

Приложение 1. Искусственный нейрон



Приложение 2. Сверточная нейронная сеть

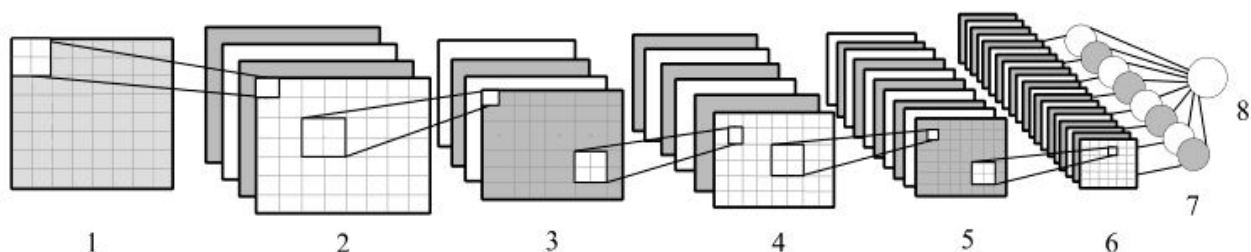
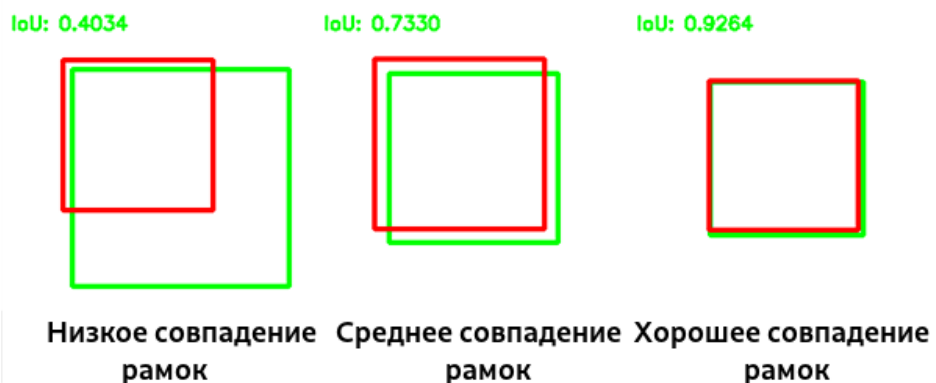


Рис. 1. Архитектура свёрточной нейронной сети: 1 – вход; 2, 4, 6 – сверточные слои; 3, 5 – подвыборочные слои; 7 – слой из обычных нейронов; 8 – выход

Приложение 3. Нейронная сеть детекции



Приложение 4. Код архитектуры нейронной сети детекции

```
inputs = preprocess(Input((64,64,3)))
x = Conv2D(32, 3, activation = 'relu', padding = 'same')(inputs)
x = Conv2D(64, 3, activation = 'relu', padding = 'same', strides = 2)(x)
x = Conv2D(64, 3, activation = 'relu', padding = 'same')(x)
x = Conv2D(64, 3, activation = 'relu', padding = 'same', strides = 2)(x)
x = Conv2D(128, 3, activation = 'relu', padding = 'same')(x)
x = Conv2D(128, 3, activation = 'relu', padding = 'same', strides = 2)(x)
x = Conv2D(128, 3, activation = 'relu', padding = 'same')(x)
x = Flatten()(x)
x = Dropout(0.4)(x)
x = Dense(256, activation = 'relu')(x)
x = Dropout(0.2)(x)
x = Dense(8)(x)

outputs = x

boxregressor = keras.Model(inputs, outputs)
```

Приложение 5. Архитектура нейронной сети классификации

```
inp = Input(shape=(wei, hei, 3))

x = Conv2D(32, (3, 3), padding='same')(inp)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = MaxPooling2D(2)(x)
x = Dropout(0.2)(x)

x = Conv2D(64, (3, 3), padding='same')(inp)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = MaxPooling2D(2)(x)
x = Dropout(0.2)(x)

x = Conv2D(128, (3, 3), padding='same')(inp)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = MaxPooling2D(2)(x)
x = Dropout(0.2)(x)

x = Flatten()(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.2)(x)
x = BatchNormalization()(x)

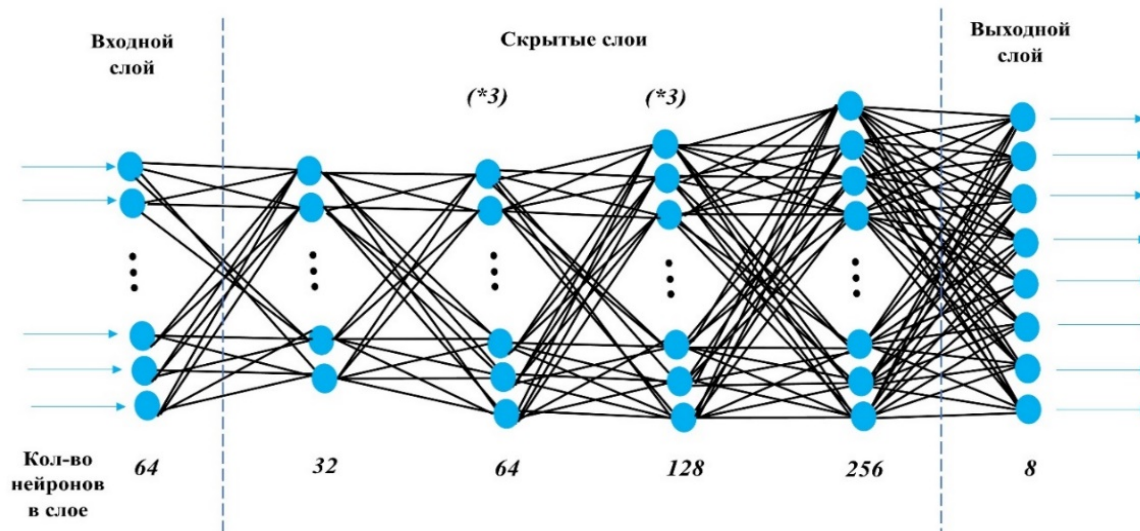
x = Dense(128, activation='relu')(x)
x = Dropout(0.2)(x)
x = BatchNormalization()(x)

x = Dense(64, activation='relu')(x)
x = Dropout(0.2)(x)
x = BatchNormalization()(x)

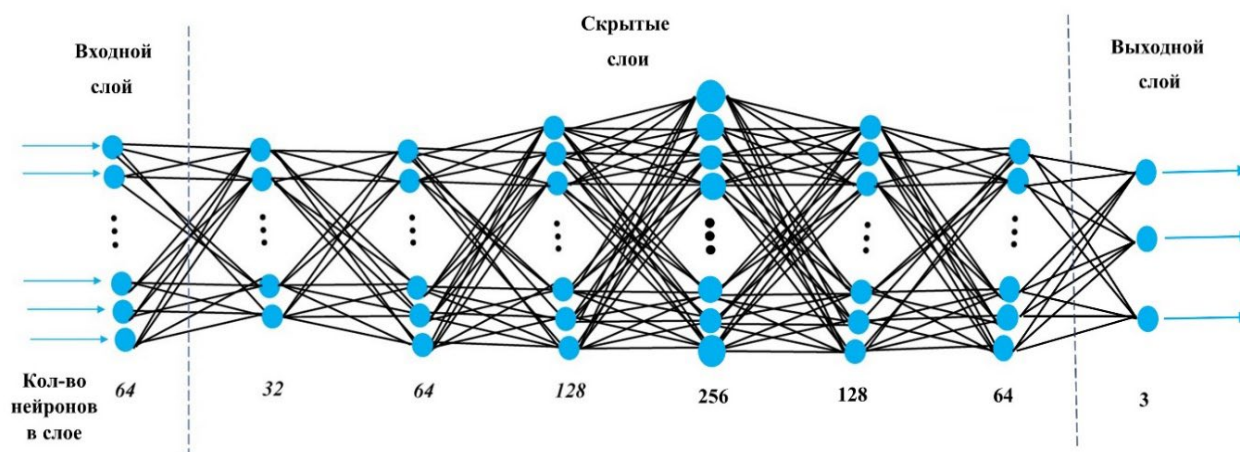
x = Dense(3, activation='softmax')(x)
model = Model(inputs=inp, outputs=x)

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

Приложение 6. Структура работы нейронной сети детекции и классификации

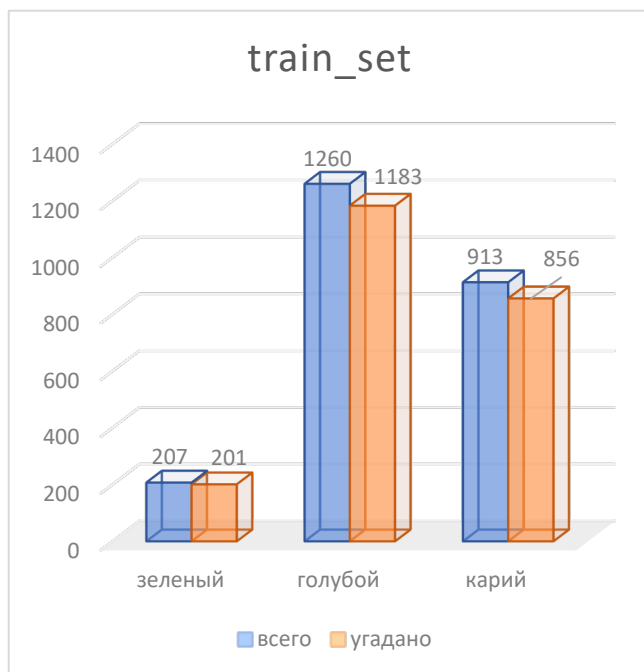


Нейронная сеть детекции

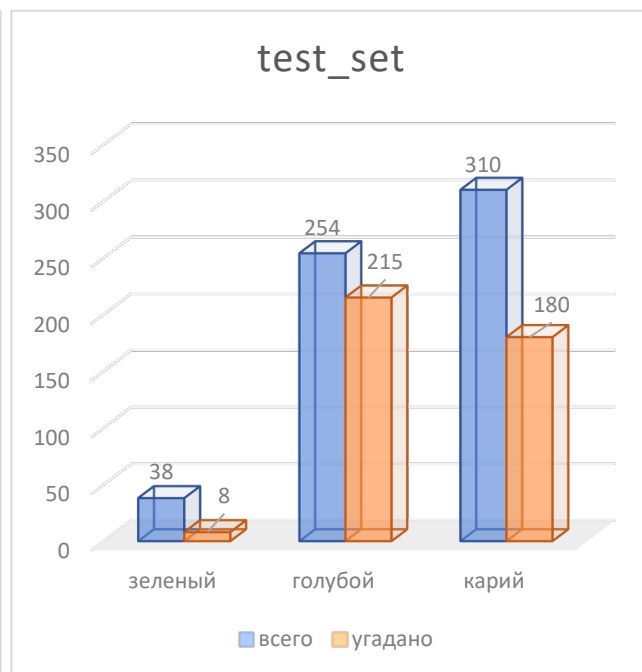


Нейронная сеть классификации

***Приложение 7. Процент верного определения цветов глаз на
тестовой и тренировочной выборках***



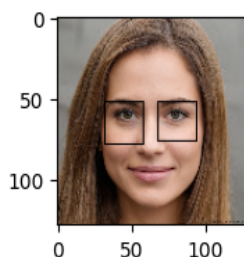
*Процент верного определения цветов глаз на
тренировочной выборке*



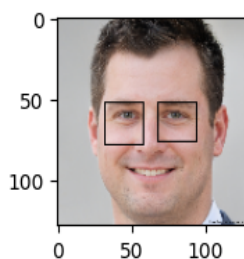
*Процент верного определения цветов глаз на
тестовой выборке*

Приложение 8. Определение местоположения глаз и их цвета

```
[[16.538277 26.23441 29.556416 39.25255 ]  
 [34.86907 26.513607 47.166763 38.811302]]
```



```
[[16.556393 26.236986 29.563063 39.243656]  
 [34.860466 26.524197 47.151276 38.815006]]
```



карий

голубой

*Определение местоположения глаз и
их цвета*

Приложение 9. Проверка данных. Диалог с пользователем

Каталог цветов глаз: голубой, зеленый, карий.

Верны ли наши данные?

Необходимо ответить "да" или "нет", если мы однозначно определили цвет;

"1" или "2" - "цифру правильного цвета: 1 - если первый цвет, 2 - если второй.

Цвет глаз мамы - карий?

1f

Некорректное введение данных, попробуйте ещё раз:

да

Отлично!

Цвет глаз папы - голубой?

да

Ожидайте

Наиболее возможный вариант, что у вашего ребенка будет карий или голубой цвет глаз.

Приложение 10. Код программы

```
"""Нейросеть"""
import tensorflow as tf
from tensorflow import keras
from keras.models import Model
from tensorflow.keras.models import load_model
from os import listdir
from os.path import isfile, join
import cv2
import numpy as np
import matplotlib.pyplot as plt

#загрузим модели
localizator = tf.keras.models.load_model('bounding_box_for_many_objects.h5')
import matplotlib

#функция работы с нейросетями: подготовка изображения, детекция
def Detect_objects(img):
    img = tf.cast(img, tf.float32)/256
    img = tf.image.resize(img,(64,64))
    big_image = tf.image.resize(img, (128, 128))
    image_exp = tf.expand_dims(img, axis = 0)
    #локализация
    bb_cords = localizator(image_exp)
    bb_cords = tf.squeeze(bb_cords, axis = 0)
    #нормализация по размеру картинки
    bb_cords = (bb_cords+1)/2*64
    bb_cords = tf.reshape(bb_cords, [2, 4])
    #разделяем на элементы
    fxmin, ymin, fxmax, ymax = tf.split(bb_cords, 4, axis = 1)
    #нормализуем
    xmin = tf.minimum(fxmin, fxmax)
    xmax = tf.maximum(fxmin, fxmax)
    ymin = tf.minimum(ymin, ymax)
    ymax = tf.maximum(ymin, ymax)
    xmin = tf.clip_by_value(xmin, 0, 64)
    ymin = tf.clip_by_value(ymin, 0, 64)
    size = xmax - xmin
    #сумма координаты и размера должны быть <= 64
    xsize = tf.clip_by_value(size, 1, 64 - xmin)
    ysize = tf.clip_by_value(size, 1, 64 - ymin)
    #нарезаем и собираем в массив (4, 32, 32, 3)
    ymin*= 2
    xmin*= 2
    ysize*= 2
    xsize*= 2
    for n in range(2):
        ii = tf.image.crop_to_bounding_box(big_image, int(ymin[n][0]), int(xmin[n][0]), int(ysize[n][0]), int(xsize[n][0]))
        ii = tf.image.resize(ii, (32,32))
        ii = tf.expand_dims(ii, axis = 0)
        if n == 0:
            cropped = ii
        else:
            cropped = tf.concat([cropped, ii], axis = 0)
    #собираем координаты в нормальный вид
    cords = tf.concat([xmin/2, ymin/2, xmin/2+xsize/2, ymin/2+ysize/2], axis = 1)
    cords = cords.numpy()
    print(cords)
    return cords

def visualize(in_image, cords):
    big_image = tf.image.resize(in_image, (128,128)).numpy() / 256
    thickness = 1#толщина
    for i in range(len(cords)):
        #ведем цвет
        color = (0, 0, 0)
        big_image = cv2.rectangle(big_image ,(int(cords[i][0])*2, int(cords[i][1])*2),
                                   (int(cords[i][2])*2,int(cords[i][3])*2),color,1)

    return big_image
```

```
from PIL import Image
import os
from keras.preprocessing.image import ImageDataGenerator
os.chdir(r'C:/Users/Vlada/anaconda3/envs/tf-gpu/neural/result/') #смена директории
import cv2
img_path = cv2.imread('3.jpg')
im = Image.open('3.jpg')
img = cv2.cvtColor(img_path, cv2.COLOR_BGR2RGB)
cords = Detect_objects(img)
im_crop_1 = im.crop((cords[0][0]*16, cords[0][1]*16, cords[0][2]*16, cords[0][3]*16))
im_crop_2 = im.crop((cords[1][0]*16, cords[1][1]*16, cords[1][2]*16, cords[1][3]*16))
im_crop_1.save("res"+"_1.jpg", quality=100)
im_crop_2.save("res" + "_2.jpg", quality=100)
result = visualize(img, cords)
plt.figure(figsize=(10, 6))
ax = plt.subplot(3,1,1)
plt.imshow(result)
plt.show()

img_path = cv2.imread('4.jpg')
im = Image.open('4.jpg')
img = cv2.cvtColor(img_path, cv2.COLOR_BGR2RGB)
cords = Detect_objects(img)
im_crop_1 = im.crop((cords[0][0]*16, cords[0][1]*16, cords[0][2]*16, cords[0][3]*16))
im_crop_2 = im.crop((cords[1][0]*16, cords[1][1]*16, cords[1][2]*16, cords[1][3]*16))
im_crop_1.save("res2"+"_1.jpg", quality=100)
im_crop_2.save("res2" + "_2.jpg", quality=100)
result = visualize(img, cords)
plt.figure(figsize=(10, 6))
ax = plt.subplot(3,1,1)
plt.imshow(result)
plt.show()
```

```
os.chdir("C:/Users/Vlada/anaconda3/envs/tf-gpu/neural/result") #смена директории
model = load_model('classification_3class_2res.h5') #загрузка нейросети из файла
os.chdir("C:/Users/Vlada/anaconda3/envs/tf-gpu/neural/result")
```

```
img_path_1 = cv2.imread('res_1.jpg')
img1 = cv2.cvtColor(img_path_1, cv2.COLOR_BGR2RGB)
img1 = cv2.resize(img1, (64, 64)) / 255.
pred_img_1 = model.predict(img1.reshape(1,64,64,3))
print('Pred_img_1: ',np.argmax(pred_img_1))
```

```
img_path_2 = cv2.imread('res_2.jpg')
img2 = cv2.cvtColor(img_path_2, cv2.COLOR_BGR2RGB)
img2 = cv2.resize(img2, (64, 64)) / 255.
pred_img_2 = model.predict(img2.reshape(1,64,64,3))
print('Pred_img_2: ',np.argmax(pred_img_2))
```

```
if (np.argmax(pred_img_1) == 0):
    cv_gl_1 = "зеленый"
if (np.argmax(pred_img_1) == 1):
    cv_gl_1 = "голубой"
if (np.argmax(pred_img_1) == 2):
    cv_gl_1 = "карий"
```

```
if (np.argmax(pred_img_2) == 0):
    cv_gl_2 = "зеленый"
if (np.argmax(pred_img_2) == 1):
    cv_gl_2 = "голубой"
if (np.argmax(pred_img_2) == 2):
    cv_gl_2 = "карий"
```

```
if (cv_gl_1 == cv_gl_2):
    cvet_glaz_mama = cv_gl_1
else:
    cvet_glaz_mama = cv_gl_1 + " или " + cv_gl_2
print(cvet_glaz_mama)
```

```
img_path_1 = cv2.imread('res2_1.jpg')
img1 = cv2.cvtColor(img_path_1, cv2.COLOR_BGR2RGB)
img1 = cv2.resize(img1, (64, 64)) / 255.
pred_img_1 = model.predict(img1.reshape(1,64,64,3))
print('Pred_img_1: ',np.argmax(pred_img_1))
```

```
img_path_2 = cv2.imread('res2_2.jpg')
img2 = cv2.cvtColor(img_path_2, cv2.COLOR_BGR2RGB)
img2 = cv2.resize(img2, (64, 64)) / 255.
pred_img_2 = model.predict(img2.reshape(1,64,64,3))
print('Pred_img_2: ',np.argmax(pred_img_2))
```

```
if (np.argmax(pred_img_1) == 0):
    cv_gl_1p = "зеленый"
if (np.argmax(pred_img_1) == 1):
    cv_gl_1p = "голубой"
if (np.argmax(pred_img_1) == 2):
    cv_gl_1p = "карий"

if (np.argmax(pred_img_2) == 0):
    cv_gl_2p = "зеленый"
if (np.argmax(pred_img_2) == 1):
    cv_gl_2p = "голубой"
if (np.argmax(pred_img_2) == 2):
    cv_gl_2p = "карий"

if (cv_gl_1p == cv_gl_2p):
    cvet_glaz_papa = cv_gl_1p
else:
    cvet_glaz_papa = cv_gl_1p + " или " + cv_gl_2p
print(cvet_glaz_papa)

import cv2
from PIL import Image, ImageDraw
g = []
shet = 0
f = open('Бинарная картинка.txt')
m = f.read()
for i in range(len(m)):
    if m[i] == '0':
        g.append(0)
    elif m[i] == '1':
        g.append(1)
imm = Image.new('RGB', (1216,1358), (255, 255, 255))#рисую полотно
draw = ImageDraw.Draw(imm)
x1 = 0
for y in range(1358):
    for h in range(1216):
        if shet != 1651328:
            if g[shet] == 0:
                draw.rectangle([(x1, y), (x1+1, y+1)], fill = 'white')#создание квадратного пикселя
                x1+=1
                if x1 == 1216:
                    x1 = 0
                    shet+=1
            elif g[shet] == 1:
                draw.rectangle([(x1, y), (x1+1, y+1)], fill = 'black')#создание квадратного пикселя
                x1+=1
                if x1 == 1216:
                    x1 = 0
                    shet+=1
        a = 1

print('Каталог цветов глаз: голубой, зеленый, карий.')
a = 0
print('Верны ли наши данные?\nНеобходимо ответить "да" или "нет", если мы однозначно определили цвет;\n"1" или "2" - "цифру п')
s = input().lower()
if (len(cvet_glaz_mama) <= 7):
    while (s != "да" and s != "нет"):
        print('Некорректное введение данных, попробуйте ещё раз:\n')
        s = input().lower()
    if (s == "да"):
        print("Отлично!")
    else: a = 1
else:
    while (s != "1" and s != "2" and s != "нет"):
        print('Некорректное введение данных, попробуйте ещё раз:\n')
        s = input().lower()
    if (s == "1"):
        cvet_glaz_mama = cv_gl_1
    elif (s == "2"):
        cvet_glaz_mama = cv_gl_2
    else: a = 1
while a == 1:
    print('Введите цвет глаз мамы:')
    cvet_glaz_mama = input().lower()
    while cvet_glaz_mama != 'зеленый' and cvet_glaz_mama != 'зелёный' and cvet_glaz_mama != 'карий' and cvet_glaz_mama != 'голубой':
        print('Некорректное введение данных, информацию можно вводить только из каталога, попробуйте ещё раз:')
        cvet_glaz_mama = input().lower()
    a = 0
```

```
print("\nЦвет глаз папы -", cvet_glaz_papa + '?')
s = input().lower()
if (len(cvet_glaz_papa) <= 7):
    while (s != "да" and s != "нет"):
        print('Некорректное введение данных, попробуйте ещё раз:\n')
        s = input().lower()
    if (s == "да"):
        print("Ожидайте")
        a = 2
    else: a = 11
else:
    while (s != "нет" and s != "1" and s != "2"):
        print('Некорректное введение данных, попробуйте ещё раз:\n')
        s = input().lower()
    if (s == "1"):
        cvet_glaz_papa = cv_g1_1p
        print("Ожидайте")
        a = 2
    elif (s == "2"):
        cvet_glaz_papa = cv_g1_2p
        print("Ожидайте")
        a = 2
    else: a = 11

while a == 11:
    print('Введите цвет глаз папы:')
    cvet_glaz_papa = input().lower()
    while cvet_glaz_papa != 'зеленый' and cvet_glaz_papa != 'зелёный' and cvet_glaz_papa != 'карий' and cvet_glaz_papa != 'голубой':
        print('Некорректное введение данных, информацию можно вводить только из каталога, попробуйте ещё раз:')
        cvet_glaz_papa = input().lower()
    print('Ожидайте')
    a = 2

while a == 2:
    if cvet_glaz_mama == cvet_glaz_papa:
        if cvet_glaz_mama == 'голубой':
            cvet_glaz_rebenka = (0, 191, 255)
            cvetg = 'голубой'
        if cvet_glaz_mama == 'карий':
            cvet_glaz_rebenka = (139, 69, 19)
            cvetg = 'карий'
        if cvet_glaz_mama == 'черный' or cvet_glaz_mama == 'чёрный':
            cvet_glaz_rebenka = (0,0,0)
            cvetg = 'черный'
    elif cvet_glaz_mama == 'карий' or cvet_glaz_papa == 'карий':
        if cvet_glaz_papa == 'голубой' or cvet_glaz_mama == 'голубой':
            cvet_glaz_rebenka = (139, 69, 19)
            cvetg = 'карий или голубой'
        else:
            cvet_glaz_rebenka = (139, 69, 19)
            cvetg = 'карий или жёлтый'
    elif (cvet_glaz_mama == 'зеленый' or cvet_glaz_mama == 'зелёный' and cvet_glaz_papa == 'голубой') or (cvet_glaz_papa == 'зеленый' or cvet_glaz_papa == 'зелёный' and cvet_glaz_mama == 'голубой'):
        cvet_glaz_rebenka = (124, 252, 0)
        cvetg = 'голубой (может быть с примесью серого) или зелёный'
    a = 3

def Cras_glaz_pixl(x,y):
    svet_gran = (0, 0, 0)
    svet_crask = cvet_glaz_rebenka
    pixl = (x,y)
    stack_list = []
    stack_list.append(pixl)#Занесение затравочного пиксела в стек.
    while(len(stack_list) != 0):#Пока стек не пуст выполнить следующие действия:
        s = stack_list.pop()
        imm.putpixel(s, (cvet_glaz_rebenka))
        if imm.getpixel((s[0]+1, s[1])) != svet_gran and imm.getpixel((s[0]+1, s[1])) != svet_crask:
            n = (s[0]+1, s[1])
            stack_list.append(n)
        if imm.getpixel((s[0]-1, s[1])) != svet_gran and imm.getpixel((s[0]-1, s[1])) != svet_crask:
            n1 = (s[0]-1, s[1])
            stack_list.append(n1)
        if imm.getpixel((s[0], s[1]+1)) != svet_gran and imm.getpixel((s[0], s[1]+1)) != svet_crask:
            n2 = (s[0], s[1]+1)
            stack_list.append(n2)
        if imm.getpixel((s[0], s[1]-1)) != svet_gran and imm.getpixel((s[0], s[1]-1)) != svet_crask:
            n3 = (s[0], s[1]-1)
            stack_list.append(n3)
    Cras_glaz_pixl(714,893)
    Cras_glaz_pixl(303,893)
    imm.show()
    print('Наиболее возможный вариант, что у вашего ребенка будет', cvetg, 'цвет глаз.')
    imm.save('Результат.jpg', quality=1000)
```