Министерство образования и науки Российской Федерации
Национальный исследовательский технологический университет «МИСиС»
Институт информационных технологий и
автоматизированных систем управления
Кафедра Инженерной Кибернетики

Курсовая работа

по дисциплине «Технологии программирования»
на тему «Распознавание объектов Tensorflow/OpenCV»

Выполнил:
студент гр. БПМ-18-2
Соседка А. В.

Проверил:
доцент кафедры ИК, к.т.н.
Полевой Д.В.

Москва, 2019

1. Задача

Создание кросс платформенного приложения для помощи с выбором продуктов питания при помощи Tensorflow/OpenCV/Qt. Модель способна определять съедобные предметы на фотографии из списка и предоставлять пользователю ссылку для покупки данного предмета.
Функционал программы:

- Открывать картинку на компьютере и определять её;
- Использовать веб-камеру для получения фотографии и определять её;
- По определённой фотографии предоставлять пользователю ссылку для покупки товара.

2. Описание программы

При разработке приложения были созданы графический интерфейс при помощи Qt и библиотека для работы с моделью Tensorflow через OpenCV. Основной вид приложения (файловый режим) (см. Рисунок 1)

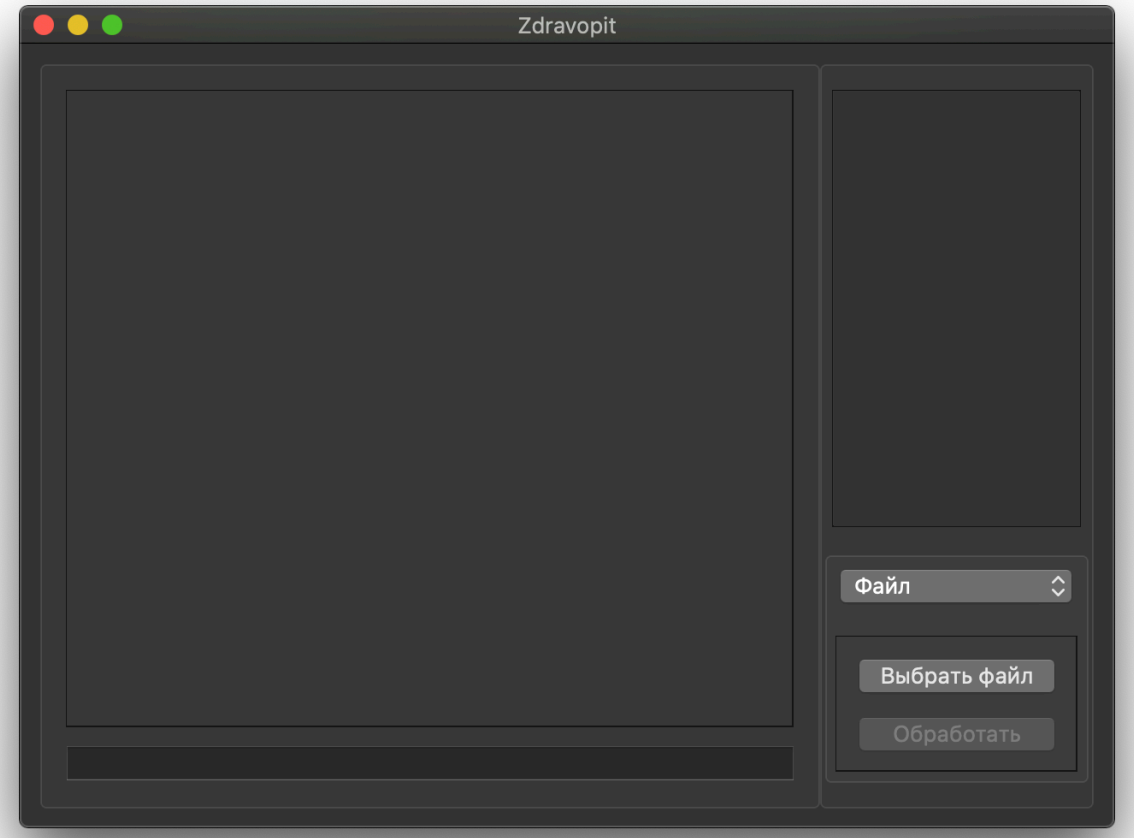

Рисунок 1

В файловом режиме пользователь может выбрать файл изображение на своём компьютере для определения на нём объектов. Когда было выбрано изображение появляется возможность обработать фото (см. Рисунок 2). После обработки, если на фото были найдены искомые предметы, они будут обведены в рамку, а в окне справа появятся их иконки с возможностью перехода по ссылке для покупки или удаления данного продукта из текущего списка. (см. Рисунок 3)

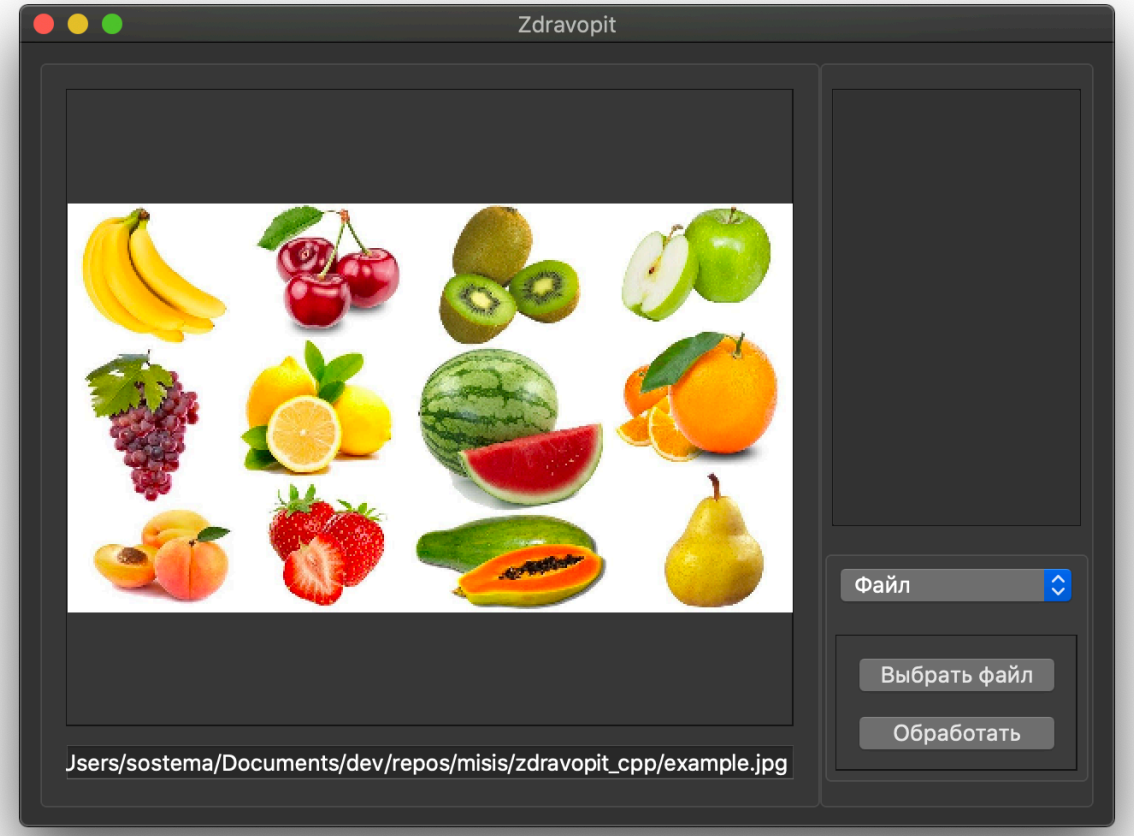

Рисунок 2

Рисунок 3

3. Инструкция по сборке

Ссылка на репозиторий: https://github.com/sostema/zdravopit_cpp
Для сборки требуется наличие: OpenCV (проверено на версии 4.20), Qt 5.13.2+ (проверено на версии 5.13.2), CMake 3.15+ (проверено на 3.15+), Visual Studio (проверено на Visual Studio 2019, должно работать на версии Visual Studio 2017)
Опционально: Python3 (проверено на 3.7+), Doxygen (для создания документации)

1. Установить OpenCV, Qt, CMake, Visual Studio при необходимости
2. Скачать репозиторий с проектом и распаковать в удобном месте
3. Открыть директорию с проектом в CMake GUI
4. Установить опции QT_PROJECT_PATH и OPENCV_PROJECT_PATH в проекте (это - пути к папкам, содержащим Qt5Config.cmake и OpenCVConfig.cmake соответсвенно) (см. Рисунок 4)
5. Сконфигурировать проект и запустить его в Visual Studio
6. Собрать проект Zdravopit, выбрав его в списке справа (см. Рисунок 5)
    1. ВНИМАНИЕ! При сборке возможна ошибка C1060 (compiler is out of heap space). Это связано с тем, что CMake не смог найти Qt макрос qt5_add_big_resources, необходимый для корректной работы. Для решения данной проблемы, либо обновите версию Qt, либо используйте следующие инструкции: https://docs.microsoft.com/en-us/cpp/error-messages/compiler-errors-1/fatal-error-c1060?redirectedfrom=MSDN&view=vs-2019
7. Запустить собранное приложение

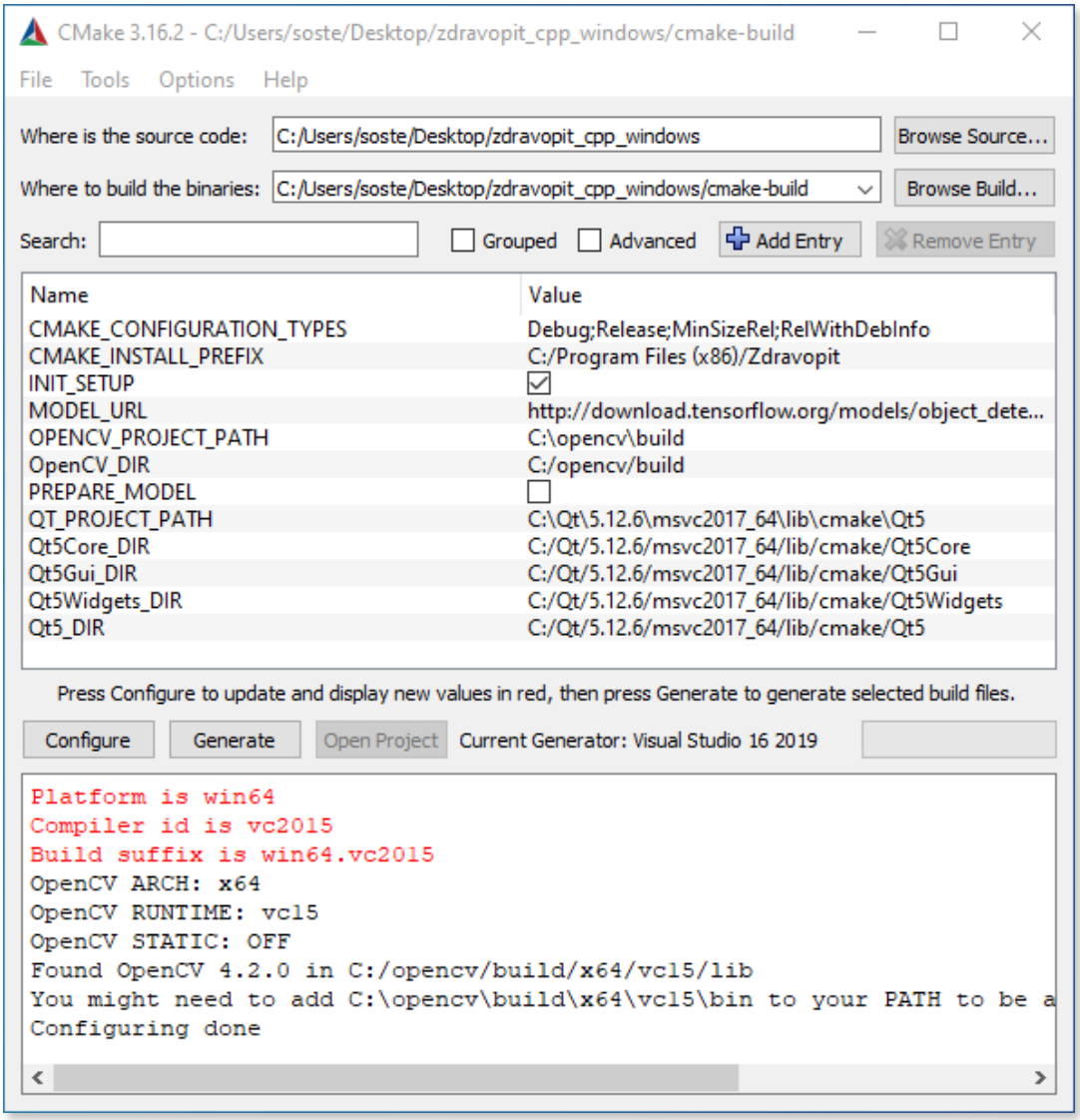

Рисунок 4

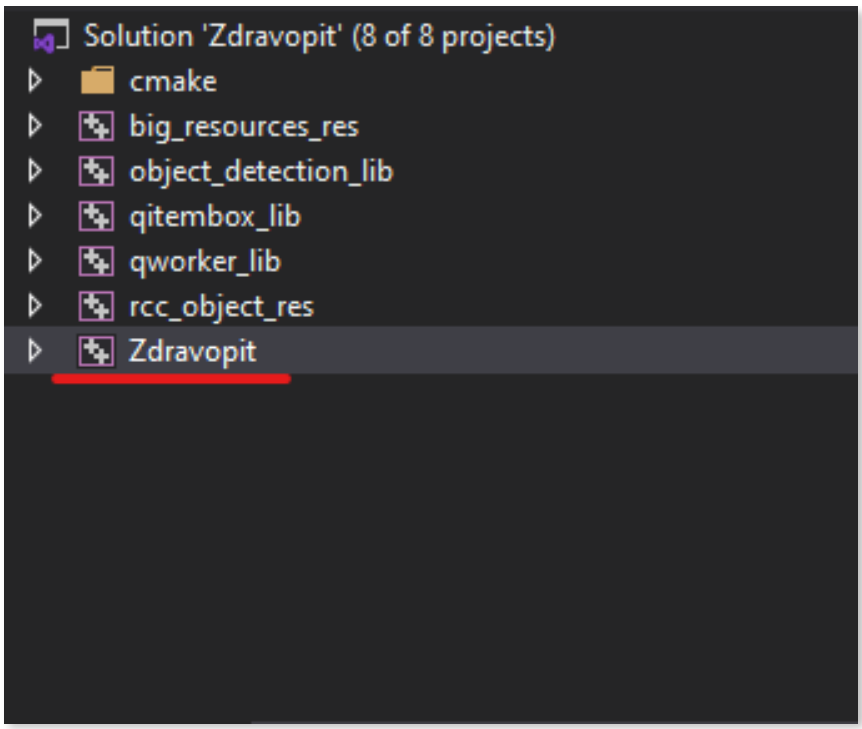

Рисунок 5

4. Реализация классов

4.1 Класс MainWindow

Основной класс для графического интерфейса

#include <mainwindow.h>

Открытые члены

MainWindow (QWidget *parent=0)

Конструктор Qt.

~MainWindow () override

Деструктор Qt.

void **closeEvent** (QCloseEvent *event) override

Событие закрытия

Закрытые слоты

void **on_chooseFileButton_clicked** ()

Событие при нажатии кнопки выбора файла

void **on_processFileButton_clicked** ()

Событие при нажатии кнопки обработки файла

void **on_beginCameraButton_clicked** ()

Событие при нажатии кнопки начала видеопотока

void **on_makePhotoButton_clicked** ()

Событие при нажатии кнопки снимка текущего кадра видеопотока

void **resizeEvent** (QResizeEvent *event) override

Событие изменения размера окна

void **on_modeSelect_currentIndexChanged** (int v)

Событие при изменении текущего состояния

void **update_main_graphics** (const cv::Mat &frame)

Явно изменяет текущую картинку

void **add_products** (const std::vector< **Model::Prediction** > &predictions)

Добавляет продукты в список справа

Закрытые члены

void **startVideoCapture** ()

Метод для явного начала видеопотока

void **endVideoCapture** ()

Метод для явного окончания видеопотока

void **applyPixmap** ()

Метод для явного изменения QLabel.

void **processFrame** (cv::Mat frame)

Метод для вызова обработки кадра

Закрытые данные

Ui::MainWindow * **ui_** {}

*Интерфейс объекта класса **MainWindow**.*

QPixmap **pixmap_** {}

QPixmap, который используется для преобразования cv::Mat для использования в QLabel.

bool **process_** {false}

Текущее состояние видеопотока

cv::VideoCapture **video_capture_**

Видеопоток

std::string **modelConfigPath_** {":/model.pbtxt"}

Путь к описанию модели

std::string **modelPath_** {":/model.pb"}

Путь к модели

std::string **labelsPath_** {":/labels.txt"}

Путь к файлу с текстовыми метками

std::string **urlsPath_** {":/urls.txt"}

Путь к файлу с ссылками на товары

std::string **configPath_** {}

Путь к файлу с конфигурацией

std::map< int, std::string > **labels_**

Словарь пар номер-название для классов

std::map< int, std::string > **urls_**

Словарь пар номер-ссылка для классов

std::map< std::string, std::string > **config_**

Словарь конфигурационного файла

Методы

void **MainWindow::add_products** (const std::vector< Model::Prediction > & predictions)[private], [slot] - Добавляет продукты в список справа

Аргументы

predictions	вектор предсказаний

void **MainWindow::closeEvent** (QCloseEvent * event)[override] - Событие закрытия.
При закрытии приложения проверяет, открыт ли видеопоток, и если да, то закрывает его.

void **MainWindow::on_modeSelect_currentIndexChanged** (int v)[private], [slot] - Событие при изменении текущего состояния.
Изменяет интерфейс при смене текущего состояния (фото/видео)

Аргументы

v	текущий индекс

void **MainWindow::processFrame** (cv::Mat frame)[private] - Метод для вызова обработки кадра

Аргументы

Frame	кадр для обработки

void **MainWindow::update_main_graphics** (const cv::Mat & frame)[private], [slot] - Явно изменяет текущую картинку.

Изменяет текущую картинку на QLabel на новую.

Аргументы

Frame	новая картинка

4.2 Класс Model

Класс для обработки кадра

#include <object_detection.h>

Классы

struct **Prediction**

Структура предсказания

Открытые члены

Model ()=default

Стандартный конструктор (для создания пустого объекта модели)

Model (std::string &modelPath, std::string &configPath, std::vector< std::string > classes, std::string &framework=(std::string &) "", float confThreshold=0.5f, float nmsThreshold=0.5f, float scale=1, int inpWidth=640, int inpHeight=640, cv::Scalar mean=cv::Scalar(0, 0, 0), bool swapRB=true)

Конструктор

~**Model** ()=default

Деструктор

Model (const **Model** &)=default

Конструктор копирования

void **predict** (cv::Mat &frame)

Метод для обработки изображения

void **preprocess** (cv::Mat &frame)

Метод для предобработки изображения

void **postprocess** (cv::Mat &frame, const std::vector< cv::Mat > &outs)

Метод для постобработки изображения

void **drawPred** (int classId, float conf, int left, int top, int right, int bottom, cv::Mat &frame)

Метод для изображения границ найденных объектов

Открытые атрибуты

std::vector< **Prediction** > **currentPredictions** {}

Вектор текущих предсказаний

cv::Mat **currentFrame_** {}

Текущее изображение для обработки

Закрытые данные

cv::dnn::Net **net_**

Объект *с Tensorflow моделью в обёртке OpenCV DNN.*

float **confThreshold_** {0}

float **nmsThreshold_** {0}

float **scale_** {0}

int **inpWidth_** {0}

int **inpHeight_** {0}

cv::Scalar **mean_**

bool **swapRB_** {true}

std::vector< std::string > **classes_**

Вектор названий классов

std::vector< std::string > **outNames_**

4.3 Структура Model::Prediction

Класс для обработки кадра

#include <object_detection.h>

Открытые члены

Prediction (cv::Mat frame, std::string label)

Конструктор

Открытые атрибуты

cv::Mat **frame_**

Изображение предсказанного продукта

std::string **label_**

Название класса предсказанного продукта

4.4 Класс QItemBox

Класс представления продукта

#include <qitembox.h>

Сигналы

void **delete_from_layout** ()

Сигнал при удалении

Открытые члены

QItemBox (QWidget *parent=0, const QImage q_image=QImage(), std::string item_url="", std::string label="")

Конструктор Qt.

~QItemBox () override

Деструктор Qt.

Закрытые слоты

void **on_deleteButton_pressed** ()

Событие при нажатии кнопки удаления

void **on_buyButton_pressed** ()

Событие при нажатии кнопки покупки

Закрытые данные

Ui::QItemBox * **ui_** {}

*Интерфейс объекта класса **QItemBox**.*

QGraphicsScene **scene_** {}

QGraphicsScene для QLabel.

QGraphicsPixmapItem **pixmap_** {}

QGraphicsPixmapItem для QGraphicsScene.

std::string **item_url_** {}

Строка, содержащая ссылку на продукт

std::string **label_** {}

Строка, содержащая название класса продукта

Конструктор(ы)

QItemBox::QItemBox (QWidget * *parent* = `0`, const QImage *q_image* = `QImage()`, std::string *item_url* = `""`, std::string *label* = `""`)`[explicit]`

Конструктор Qt.

Аргументы

Parent	Родительский QWidget
q_image	Фото продукта из картинки
item_url	Ссылка на продукт
Label	Название класса продукта

4.5 Класс QWorker

Промежуточный класс для работы с предсказанием

#include <qworker.h>

Открытые слоты

void **processFrame** (const cv::Mat &frame)

Событие для предсказания изображения

void **throughFrame** (const cv::Mat &frame)

Событие для "пропуска" кадра

Сигналы

void **predictionsReady** (std::vector< **Model::Prediction** > predictions)

Сигнал, передающий готовые предсказания

void **frameReady** (const cv::Mat &frame)

Сигнал, передающий предсказанное изображение, обработанное моделью

Открытые члены

QWorker (std::string &modelPath, std::string &modelConfigPath, std::map< int, std::string > &classes, std::map< std::string, std::string > config)

Конструктор

Закрытые данные

Model model_ {}

Экземпляр модели