Министерство образования и науки Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

Петрозаводский государственный университет

Институт математики и информационных технологий Кафедра информатики и математического обеспечения

Отчет по дисциплине Методы тестирования ПО

Выполнил: Аверков В. А.

Руководитель: к.ф-м.н., доцент К. А. Кулаков

Петрозаводск 2020

Объект тестирования

Детектор человека и защитной экипировки

Описание приложения

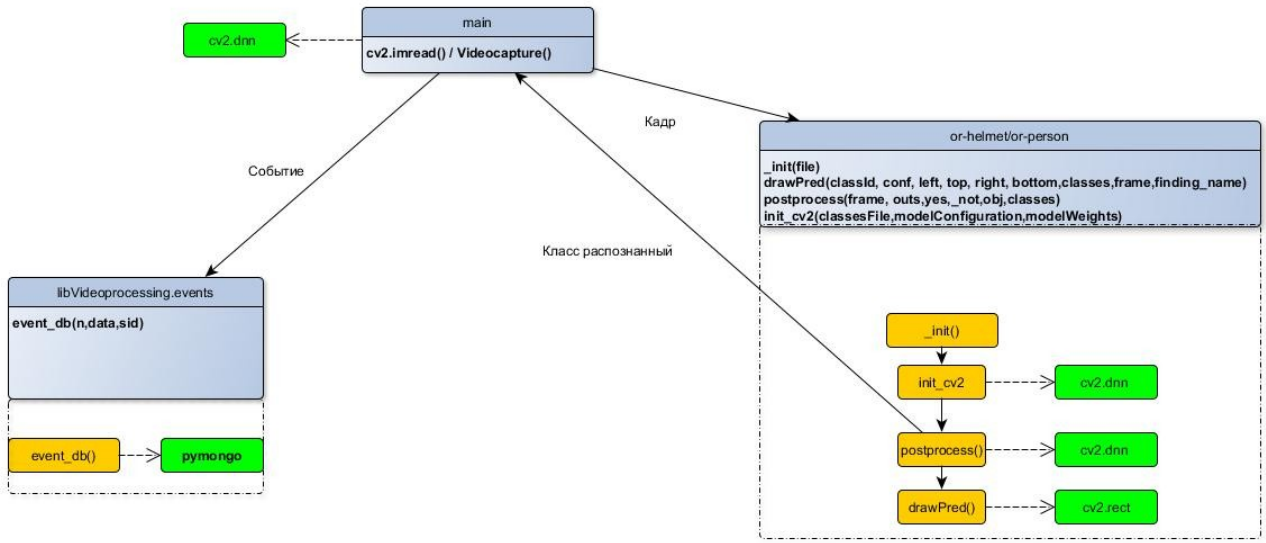
Детектор будет реализован, в виде видеосервиса в системе производственного мониторинга Opti-Repair и будет выполнять контроль над соблюдением техники безопасности через камеру наблюдения

Функциональные требования

1. Распознавание человека
2. Распознавание защитной экипировки
3. Отправка события в Базу Данных

Стратегия тестирования

Архитектура



libvideoprocessing.events. - Модуль отправки события в Базу Данных

or-helmet/person-detection - Модуль распознавания объектов

Тестирование модулей и их методов будет производиться с целью:

- Проверки того что отправляются события в Базу Данных с определенным кодом.
- Проверки того что человек/каска распознаются методы модуля корректно работает без критических сбоев.

Блочное тестирование

Язык программирования

- Python3.6

Инструменты

- Gitlab CI
- PyTest
- Coverage
- MongoDB

Запуск в среде GitlabCI, проверка самостоятельных элементов Успехом будет считаться правильное выполнение функции.

- Тестируется работа метода/ов отвечающих за работу с ini и базой данных
- Не тестируются отдельно методы обработки видеоданных OpenCV так как OpenCV протестирован уже создателями

Специальное

Язык программирования

- Python3.6

Инструменты

- Gitlab CI
- PyTest
- OpenCV

Запуск в среде GitlabCI, проверка того что при разных разрешениях будут на изображении определяться правильные классы. Успехом будет считаться что получены правильные метки классов в соответствии с содержанием фото

- **Тестируются** метод OpenCV с подключенной к нему моделью машинного обучения YOLO на разных разрешениях изображения. **Акцент на поведении нейронной сети.**

Функциональное

Язык программирования

- Python3.6

Инструменты

- Gitlab CI
- PyTest
- OpenCV
- MongoDB

Запуск в среде GitlabCI, проверка того что ПО предоставляет и выполняет одно из функциональное требование на выбор тестировщика.

Проверка реализуемости функциональных требований (Например проверить распознается ли человек и отправка событий и тд)

В тесты включены оба модуля :

- libVideoprocessing
- or-helmet/person

Интеграционное

Язык программирования

- Python3.6

Инструменты

- Gitlab CI
- PyTest
- OpenCV
- MongoDB

Проверка взаимодействия методов модулей сервиса

будет протестирована инициализация (or-helmet/human), обнаружение (с помощью заглушки)

и отправка в базу данных (libvideoprocessing)

Аттестационное

Язык программирования

- Python3.6

Инструменты

- Gitlab CI
- PyTest
- OpenCV
- MongoDB

Запуск в среде GitlabCI, проверка того что ПО в полной мере работает на видеоданных, т.е обрабатывает видеопоток и генерирует сообщения

В тест/ы входят все методы модулей. На выходе получаем осмысленные данные т.е события или их кол-во

Перечень методов

libVideoProcessing (events.py)

db_event(n, data, sid) - создание события и загрузка в коллекцию бд, заполнение полей в зависимости от сервиса/события

n – код (int)

data – какие-либо данные (str/int)

sid – id сервиса (str, генерируется при создании канала RabbitMQ при инициализации сервиса)

or-helmet/or-person-recognition

_input (file) – получение путей до ресурсов из ini

file – путь до ini (str)

postprocess (frame, outs, obj, classes) – обработка результатов работы нейронной сети

frame – кадр (array)

outs – результаты (array)

obj – объект для поиска (ini) (str)

classes – классы в классфикаторе (str)

drawPred (classId, conf, left, top, right, bottom, classes, frame, finding_name) – отрисовка “квадрата”

classId – id класса (int)

conf – вероятность (float)

left, top, right, bottom – Координаты (int)

frame – кадр (array)
finding_name – объект для поиска (str)

init_cv2 (classesFile,modelConfiguration,modelWeights) – инициализация модели

classesFile- путь до файла с классами (str)

modelConfiguration — путь до конфига нейронной сети (str)

modelWeights – путь до файла с весами (str)

*Пояснение sid!=пустому значению , содержит строчные латинские символы и цифры. При тестировании id=const, где const=`5f99765c538f2c300340ba43`

* под array подразумевается

```
[[[255 255 255 255]
 [255 255 255 255]
 [255 255 255 255]
 ...
 [255 255 255 255]
 [255 255 255 255]
 [255 255 255 255]]

[[255 255 255 255]
 [255 255 255 255]
 [255 255 255 255]
 ...
 [255 255 255 255]
 [255 255 255 255]
 [255 255 255 255]]

[[255 255 255 255]
 [255 255 255 255]
 [255 255 255 255]
 ...
 [255 255 255 255]
 [255 255 255 255]
 [255 255 255 255]]

...
```

[[255 255 255 255]

[255 255 255 255]

[255 255 255 255]

...

[255 255 255 255]

[255 255 255 255]

[255 255 255 255]]

[[255 255 255 255]

[255 255 255 255]

[255 255 255 255]

...

[255 255 255 255]

[255 255 255 255]

[255 255 255 255]]

[[255 255 255 255]

[255 255 255 255]

[255 255 255 255]

...

[255 255 255 255]

[255 255 255 255]

[255 255 255 255]]]

Модуль libvideoprocessing.events.py

Метод db_event

Блочное тестирование

Тест № 1 test_database

Цель: Проверка работоспособности отправки события по коду обнаружения человека в MongoDB
Тип: Позитивный
Входные данные: n – номер события (int); data — данные (str/int) sid – id сервиса (str)

Ожидаемый результат: Успешное выполнение Обнаружеие данных в коллекции
Описание процесса: требуются:
код события (1)
данные (str/int) (string) , id сервиса sid

Тест № 2 `test_database2`

Цель: Проверка работоспособности отправки события по нужному коду обнаружения каски в MongoDB
Тип: Позитивный
Входные данные: n – номер события (int); data — данные (str/int) sid – id сервиса (str)

Ожидаемый результат: Успешное выполнение
Описание процесса: требуются:
код события (11)
данные (str/int) (string)' '
id сервиса sid

Тест № 3 `test_base_text`

Цель: Проверка работоспособности отправки события с данными и выбор нужного поля (level) в событии MongoDB
Тип: Позитивный
Входные данные: n – номер события (int); data — данные (str/int) sid – id сервиса (str)

Ожидаемый результат: Успешное нахождение записи с полем level=2
Описание процесса: требуются:
код события (11)
данные (str/int) (string) ' '
id сервиса sid

Модуль or-helmet/person-detection

Метод `_input`

Блочное тестирование

Тест № 1 `test_ini`

Цель: Проверка того что нужный ini открывается
Тип: Позитивный

Входные данные: file - строка которая содержит путь до ini файла (str)
Ожидаемый результат: в возвращаемом значении obj находится класс для поиска (Helmet)
Описание процесса: требуется ini и путь до него

Тест № 2 `test_ini_fail`

Цель: Проверка того что нужный ini открывается и он не содержит Helmet, но содержит person
Тип: Позитивный
Входные данные: file - строка которая содержит путь до ini файла (str)
Ожидаемый результат: в возвращаемом значении obj находится класс не равный Helmet
Описание процесса: требуется ini и путь до него

Тест № 3 `test_ini_fail2`

Цель: Проверка того что функция выдает исключения
Тип: Позитивный
Входные данные: file (str) - строка которая содержит информацию которая не является путем до ini (Например: «resourses/djdfkhfskfhs.dkhjdso»)
Ожидаемый результат: сработает `except`

Описание процесса: требуется ini и **НЕСУЩЕСТВУЮЩИЙ ИЛИ ЛОЖНЫЙ** путь до него

Функциональное тестирование

Тест № 1 `test_load_image`

Цель: Проверка того что нужный ini открывается и по данным с него используется нужная модель и на изображении находится 1 шлем
Тип: Позитивный
Входные данные: Путь до изображения png/jpg (str) , путь до ini (str)
Ожидаемый результат: изображении находится 1 шлем
Описание процесса: требуется ini и путь до него

Тест № 2 `test_other_image_1`

Цель: Проверка того что нужный ini открывается и по данным с него используется нужная модель и на изображении находится 2 шлема
Тип: Позитивный
Входные данные: Путь до изображения png/jpg (str) , путь до ini (str)

Ожидаемый результат: изображении находится 2 шлема

Описание процесса: требуется ini и путь до него

Тест № 3 `test_hum_1`

Цель: Проверка того что нужный ini открывается и по данным с него используется нужная модель и на изображении с завода находится 1 человек
Тип: Позитивный
Входные данные: Путь до изображения png/jpg (str) , путь до ini (str)
Ожидаемый результат: изображении находится 1 человек

Описание процесса: требуется ini и путь до него

Тест № 4 `test_hum_2`

Цель: Проверка того что нужный ini открывается и по данным с него используется нужная модель и на изображении есть люди
Тип: Позитивный
Входные данные: Путь до изображения png/jpg (str) , путь до ini (str)
Ожидаемый результат: изображении есть люди
Описание процесса: требуется ini и путь до него

Тест № 5 `test_with_db`

Цель: Нужный ini открывается и по данным с него используется нужная модель и на изображении есть люди и также отправляется событие в MongoDB. Проверка того что событие приходит.
Тип: Позитивный
Входные данные: Путь до изображения png/jpg (str), путь до ini (str)
Ожидаемый результат: Событие приходит
Описание процесса: требуется ini и путь до него , изображение

Специальное тестирование

Цель: Проверка обнаруженных классов на разных разрешениях изображения

Тип каждого теста: Позитивный

Входные данные: изображение (array)

Ожидаемые результаты: Обнаружены правильные классы которые изображены

Номер	Входной слой	Разрешение входного изображения	Класс
1	416;416	1920,1080	Person
2	416;416	720,480	Person
3	416;416	1920,1080	Helmet
4	416;416	720,480	Helmet

Интеграционное тестирование

Тест № 1 `test_integr`

Цель: Проверка того что методы модулей правильно срабатывают в паре.
Тип: Позитивный
Входные данные: путь до ini (str)
Ожидаемый результат: В Mongo DB появляются событие о наличии объектов в кол-ве 1 штука

Описание процесса: требуется ini и путь до него Проверка наличия в коллекции событий *модель инициализировалась (or-helmet/or-person) → объекты есть (заглушка)-> отправляем в БД (libvideoprocessing)

Аттестационное тестирование

Тест № 1 `test_with_video`

Цель: Проверка того что модуль исправно работает с видеоданными, обнаруживает и отправляет события.
Тип: Позитивный
Входные данные: путь до видеофайла avi, mp4 (str) , путь до ini (str)
Ожидаемый результат: В Mongo DB появляются события о наличии объектов

Описание процесса: требуется ini и путь до него , видеофайл. Проверка наличия в коллекции событий

Образец теста

```
frame=cv2.imread("./YoloPyTestS/test.jpg")
frame=resolution(frame,1920,1080)# изменение разрешения
path = './YoloPyTestS/app/ini/or-human_detection.ini'

def test_resolution_1(frame,path):

classesFile,modelConfiguration,modelWeights,obj=app._input(path)
```

```
classes=None
net,classes=app.init_cv2(classesFile,modelConfiguration,modelWeights)
frame=cv2.rotate(frame,cv2.ROTATE_90_COUNTERCLOCKWISE)
blob=cv2.dnn.blobFromImage(
frame,1/255,(inpWidth,inpHeight),[0,0,0],1,crop=False)
net.setInput(blob)# подача обработанного изображения в нейроннуюсеть
outs=net.forward(app.getOutputsNames(net))# вычисление результатов
frame,count,label=app.postprocess(frame,outs,obj,classes) #обработка
изображения
assert label=="person" # проверка ожидаемого и фактического значения
```

Представлен один из Специальных тестов

Отчет об ошибках

Ошибка № 1

Описание: Не объявленная переменная

Тест: `test_with_video`

Описание процесса: Во время обработки видео возникла ошибка UnboundLocalError. Метод postprocess()

Входные данные: frame (array) -кадр , outs – вычисленные моделью классы (array),obj – объект поиска (str из ini),classes — классы нейронной сети (str)

Ожидаемый результат: кадр (array), кол-во объектов (int), класс объекта (str)

Фактический результат: UnboundLocalError: local variable 'unknown_class' referenced before assignment

Возможная причина: Невнимательность программиста

Статус: Исправлено

Журнал тестирования

Дата	Тестировщик	Название теста	Вид теста	Тип теста	Итог
12.12.2020	Аверков В.А	test_database	Блочный	Позитивный	OK
12.12.2020	Аверков В.А	test_database2	Блочный	Позитивный	OK
12.12.2020	Аверков В.А	test_base_text	Блочный	Позитивный	OK

12.12.2020	Аверков В.А	test_ini	Блочный	Позитивный	OK
12.12.2020	Аверков В.А	test_ini_fail	Блочный	Позитивный	OK
12.12.2020	Аверков В.А	test_ini_fail2	Блочный	Позитивный	OK
12.12.2020	Аверков В.А	test_load_image	Функциональный	Позитивный	OK
12.12.2020	Аверков В.А	test_other_image_1	Функциональный	Позитивный	OK
12.12.2020	Аверков В.А	test_hum_1	Функциональный	Позитивный	OK
12.12.2020	Аверков В.А	test_hum_2	Функциональный	Позитивный	OK
12.12.2020	Аверков В.А	test_with_db	Функциональный	Позитивный	OK
12.12.2020	Аверков В.А	test_with_video	Аттестационный	Позитивный	Error/ исправлено
12.12.2020	Аверков В.А	1	Специальный	Позитивный	OK
12.12.2020	Аверков В.А	2	Специальный	Позитивный	OK
12.12.2020	Аверков В.А	3	Специальный	Позитивный	OK
12.12.2020	Аверков В.А	4	Специальный	Позитивный	OK
17.12.2020	Аверков В.А.	test_integr	Интеграционный	Позитивный	OK

Журнал ошибок

Номер	Дата	Тестировщик	Название теста	Значение (ожидаемое)	Значение полученное
1	12.12.2020	Аверков В.А.	test_with_video	`i>0` где i- кол-во записей в базе данных `i=collection.count_documents({ "data.count`	UnboundLocalError: local variable 'unknown_class' referenced before

				": count}) (В начале теста база пуста)	assignment

Описание

Ошибка 1 обнаружена при первом запуске теста. Несла в себе критический характер. При ее возникновении программа переставала работать и совершала вывод ошибки о неинициализированной переменной

Покрытие

- **Покрытие кода составило — 98%**
- **Использовался плагин coverage в среде pytest**

РЕЗУЛЬТАТЫ

Итог

17 - Тестов

98% - Покрытие кода 105(покрыто)/2 не покрыто

В ходе тестирования была обнаружена одна критическая ошибка из-за которой дальнейшая работа сервиса была невозможна. Ошибка была исправлена.

Функциональные требования такие как распознавание человека, распознавание защитной экипировки, отправка события в Базу Данных были удовлетворены и соблюдены.

На текущий момент данный сервис пригоден для использования в ПАК Opti-Repair

Более подробный итоговый отчет

```
5628 $ coverage run -m   --source=YoloPyTestS/app/  pytest YoloPyTestS/test.py
5629 ============================= test session starts ==============================
5630 platform linux -- Python 3.8.6, pytest-6.2.1, py-1.10.0, pluggy-0.13.1
5631 rootdir: /builds/opti-repair/video-processing
5632 plugins: cov-2.10.1
5633 collected 17 items
5634 YoloPyTestS/test.py .................                                    [100%]
5635 ======================== 17 passed in 306.77s (0:05:06) ========================
5636 $ coverage report
5637 Name                        Stmts   Miss  Cover
5638 -----------------------------------------------
5639 YoloPyTestS/app/events.py      14      0   100%
5640 YoloPyTestS/app/main.py        91      2    98%
5641 -----------------------------------------------
5642 TOTAL                         105      2    98%
5643 Cleaning up file based variables
5644 Job succeeded
```