ЛАНИТ
ЭКСПЕРТИЗА

NeuroControl – платформа для автоматизации тестирования любых мобильных и desktop приложений с использованием методов Computer Vision и нейросетей

Руководство по установке, настройке и эксплуатации

Москва

2022

Содержание

Аннотация

Настоящий документ содержит описание функциональных характеристик Программного комплекса NeuroControl (далее – ПО), а также сведения, необходимые для:

- установки ПО;
- конфигурирования ПО;
- эксплуатации ПО;
- поддержания жизненного цикла ПО, в том числе устранения неисправностей и совершенствования ПО;

а также информацию о персонале, необходимом для обеспечения такой поддержки.

Список сокращений

Сокращение	Определение
NeuroControl, ПО	NeuroControl – платформа для автоматизации тестирования любых мобильных и desktop приложений с использованием методов Computer Vision и нейросетей
QA	специалист по обеспечению качества разработки программного обеспечения
АРМ	автоматизированное рабочее место
АТ	автоматизированное тестирование; автотест
СТП	служба технической поддержки
ТК	тест кейс, тестовый сценарий
ФТ	функциональное тестирование; тестовый сценарий для выполнения функционального тестирования

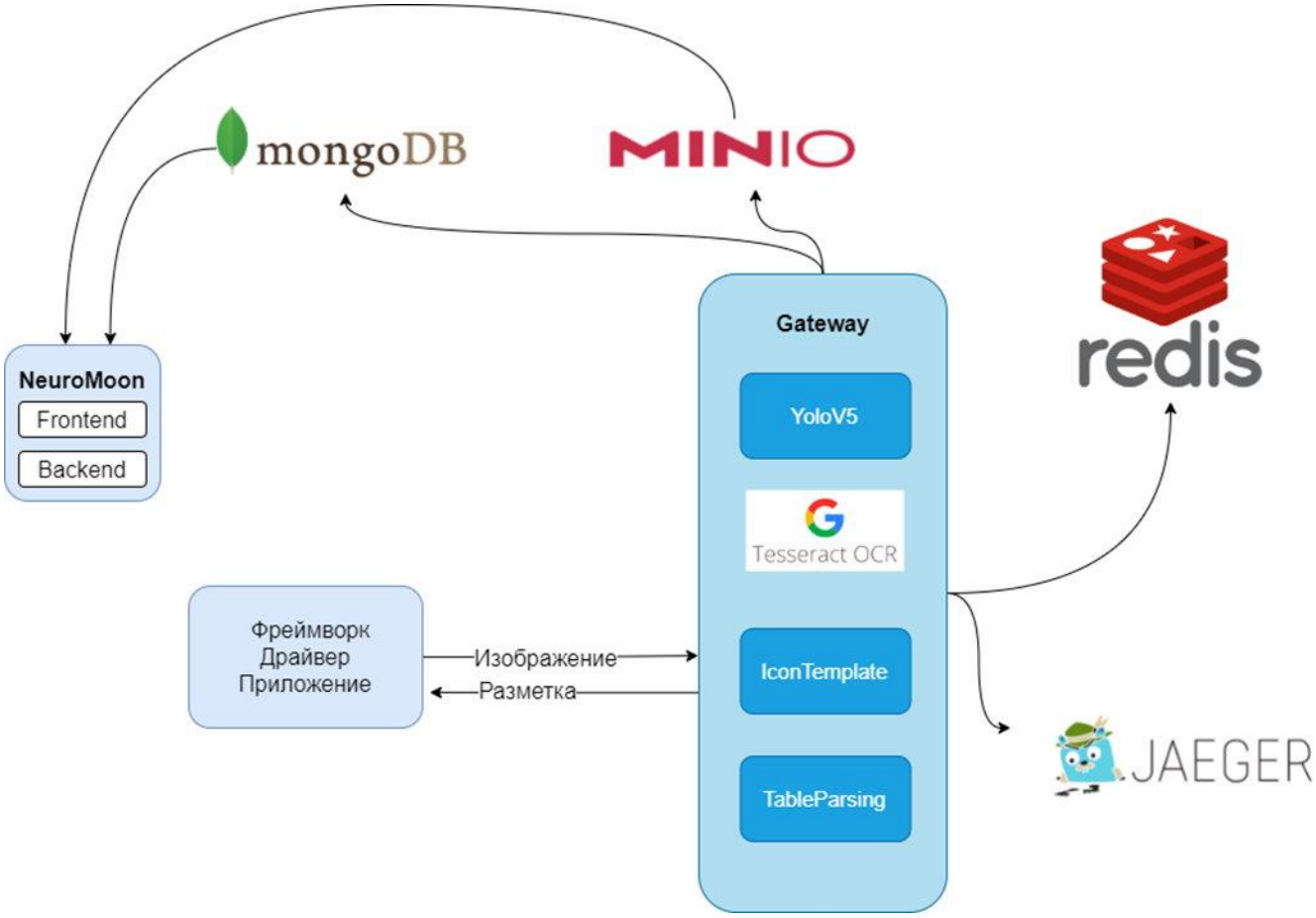
1 Общие сведения

NeuroControl — платформа для автоматизации тестирования любых мобильных и desktop приложений. Решение распознает элементы UI-интерфейса с использованием методов Computer Vision и нейросетей. Координаты распознанных элементов и их атрибуты могут быть использованы любым типовым фреймворком автоматизации.

2 Описание функциональных характеристик

NeuroControl - система для автоматизации десктоп-приложений с помощью методов и технологий искусственного интеллекта.

2.1 Инфраструктура системы



Таким образом, можно разделить инфраструктуру на 3 составных части:

1. Алгоритмическая

К алгоритмической части можно отнести:

- Фреймворк - фреймворк тестирования, написанный на языке Python, Java (в т.ч. Open JDK) или Java Script.
- Драйвер - драйвер для подключения к приложению
- Приложение - само тестируемое приложение. На самом деле не является частью инфраструктуры, а лишь объектом, над которым проводятся все действия.

По указанию тестировщика фреймворк через драйвер снимает и отправляет скриншот приложения для получения разметки в нейросетевую часть.

2. Нейросетевая

В нейросетевой части происходит распознавание компонентов, текста на скриншоте, сопоставление иконок и шаблонов и т.д. В эту часть входят следующие микросервисы:

- YoloV5 - сервис, содержащий нейронные сети для распознавания компонентов и их локализации. Кроме распознавания типов компонентов (например, кнопок) может локализовать текст и определить атрибуты компонентов (доступна кнопка или нет, поставлен checkbox или нет, и т.д)
- OCR Tesseract - сервис для распознавания текста на изображении
- IconTemplate - сервис для сопоставление иконок или шаблонов на изображении
- TableParsing - парсинг таблиц, получение их структуры. На самом деле является функционалом сервиса IconTemplate
- Gateway - шлюз для обращения ко всем сервисам. Содержит основную логику работы, предоставляет API для фреймворка. Именно в нем происходит запись в БД и хранилище.

3. Отчетная

Система NeuroMoon

Отчетная часть позволяет визуализировать результаты запросов к нейросетевой части с помощью веб-интерфейса. В планах развитие от сугубо отчетной системы до платформы

- Backend - бэк системы
- Frontend - фронт системы

Хранилище

В качестве хранилища используются следующие технологии:

- MongoDB - NoSQL база данных для хранения результатов запросов
- MINIO - S3 хранилище изображений и датасетов
- Redis - используется в качестве кэша

Вспомогательные инструменты

- Jaeger - система трассировки
- TelegramBot - бот в соц.сети *Telegram* с нейронными сетями

3 Подготовка к установке ПО и начало работы

3.1 Руководство по развертыванию через docker-compose

В этом разделе будет описываться развертывание системы NeuroControl и NeroMoon на "чистой инфраструктуре" с помощью docker-compose

3.2 Предварительные требования

- docker
- docker-compose

Также для выгрузки изображений Docker требуется быть залогиненным в Docker-registry

3.3 Переменные окружения

Для развертывания системы используются переменные окружения, поэтому перед развертыванием следует использовать файл .env Docker-compose

Для развертывания основного функционала системы NeuroControl, а также системы репорта NeuroMoon требуется два следующих docker-compose файла.

3.3.1 Только основная система

Для основной системы файл docker-compose.yml

Поднять контейнеры

```
docker-compose pull
docker-compose up -d
```

3.3.2 Совместно с NeuroMoon

Для системы NeuroMoon docker-compose.neuromoon.yml

После чего можно будет развернуть системы с помощью docker-compose

```
docker-compose -f docker-compose.yml -f docker-compose.neuromoon.yml pull
docker-compose -f docker-compose.yml -f docker-compose.neuromoon.yml up -d
```

Для непрерывного обновления изображений в системе NeuroMoon требуется произвести репликацию базы данных MongoDB. Для этого следует зайти в контейнер с БД и запустить утилиту mongo

```
docker exec -it reports_mongo_1  mongo
```

В утилите необходимо создать реплики

```
rs.initiate(
  {
    _id : 'rs0',
    members: [
      { _id : 0, host : "mongo1:27017" }
    ]
  }
)
```

Также необходимо перейти в MINIO (host:9000 или доменное имя), создать бакет files и requests и сделать их публичными.

В дальнейшем планируется создавать бакет и делать его публичным автоматически.

После этого сервисы будут доступны на указанных портах и готовы к работе.

3.4 Swagger

Документация API сервисов доступна на {host_name}{host_port_service}/docs

Альтернативная документация доступна на {host_name}{host_port_service}/redoc

3.4.1 OpenTelemetry, Jaeger, Prometheus

Soon...

3.4.2 Kubernetes

В директории с docker-compose.yml файлом и файлом .env

```
kompose convert
kubectl apply -f .
```

4 Распознавание элементов UI

4.1 Введение

Так как система NeuroControl является микросервисной, для удобства обращения выделен отдельный сервис. Сервис Gateway - шлюз и связующее звено между сервисами системы NeuroControl. Сервис предоставляет API для выполнения запросов на локализацию компонентов и распознавания текста.

Фреймворк автотестирования напрямую взаимодействует только с сервисом Gateway. Gateway, в свою же очередь, отправляет запросы в сервисы, получает и обрабатывает ответы от сервисов в зависимости от входных данных запроса.

Ниже будут приведены объяснения и примеры, как использовать систему NeuroControl.

4.2 Распознавание элементов

Основная функция системы NeuroControl – это распознавание компонентов UI на скриншотах изображения

Система предоставляет как распознавание различных компонентов, таких как кнопки, выпадающие списки, таблицы с помощью нейронной сети локализации компонентов, так и распознавание областей на изображении, содержащем текст, с помощью нейронной сети локализации текста. Каждый из этих видов локализации не совместим с другим, хотя для обоих возможно распознавание текста. Сам текст распознается с помощью OCR на локализованных компонентах.

Info

Хотя локализация текстовых элементов и не совместима с локализацией стандартных компонентов, локализация текста в уже найденных компонентах используется для улучшения распознавания текста. Также можно самому выбрать тип локализации текста: нашей нейронной сетью или OCR!

Рассмотрим сначала локализацию с помощью нейронной сети локализации компонентов

4.2.1 Распознавание элементов (без TextElement)

Функционал распознания доступен на эндпоинте find_by

Классы, которые может распознать система (и их наименования) зависят от обученной нейронной сети. По умолчанию в предоставляемой сети заданы следующие классы:

- Button
- Checkbox
- Label
- Radio

- Table
- List
- TreeView
- Input
- TextArea
- Scrollbar
- Calendar

Для локализации всех компонентов на изображении следует отправить следующий запрос к системе.

```
curl -X 'POST' \
  'http://192.168.88.102:9464/find_by' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'file=@image.png;type=image/png'
```

Это же соответствует пустому (без изменения параметров по умолчанию) запросу в swagger

POST /find_by Find By

Parameters

Name	Description
type_ui string (query)	Component type type_ui
attributes string (query)	Attributes name attributes
text string (query)	Text text
ocr_strategy string (query)	Strategy for recognition texts ocr_strategy
localize (query)	all
use_ocr_cache boolean (query)	Use ocr cache false
token string (query)	Token for record token

Request body required

file * required

Info

Увидеть Swagger сервиса вы можете на странице {host}{service_port}/docs Для Gateway порт по умолчанию 9464.

В ответ вернется JSON с результатами распознавания.

Info

Все эти запросы вернут локализацию элементов, их атрибуты, но не вернут их текст. Для получения текста необходимо явно задать искомый текст в параметр text, или же использовать поиск TextElement. Подробнее ниже.

4.2.2 Фильтрация компонентов

Для получения не всех компонентов, а только некоторых классов, надо добавить требуемые классы в query-параметр type_ui

Например, следующий запрос вернет только элементы с классом Button

```
curl -X 'POST' \
  'http://192.168.88.102:9464/find_by?type_ui=Button' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'file=@image.png;type=image/png'
```

Несколько классов задаются через разделитель |

```
curl -X 'POST' \
  'http://192.168.88.102:9464/find_by?type_ui=Button|Input|Label' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'file=@image.png;type=image/png'
```

Этот запрос вернет компоненты, относящиеся к классам Button, Input, Label

Warning

В следующих версиях конкатенация посредством | будет убрана. Выбор нескольких компонентов будет заменен на множественные query-параметры.

4.3 Получение текста

До сих пор мы получали только локализацию компонентов и его атрибуты, перейдем к получению текста.

4.3.1 Получение текста с локализацией на стандартных компонентах

Получение текста на всех компонентах по умолчанию отключено для экономии времени, так как операция распознавания текста довольно трудозатратная. Поэтому получить текст можно при условии, что вы ищете какой-то **определенный** текст. Для этого необходимо передать в query-параметр text

```
curl -X 'POST' \
  'http://192.168.88.102:9464/find_by?type_ui=Button&text=OK' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'file=@image.png;type=image/png'
```

В результате этого запроса, вернутся элементы, на которых удалось обнаружить текст, который **содержит** в себе текст *"OK"*.

4.3.2 Локализация TextElement

Как говорилось выше, возможна локализация не на заданные классы нейронной сети, а на отдельные области, содержащие текст. Для этого используется тип элемента TextElement. Запрос с параметром type-ui=TextElement вернет список (с координатами и текстом) областей на изображении с текстом.

```
curl -X 'POST' \
  'http://192.168.88.102:9464/find_by?type_ui=TextElement' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'file=@image.png;type=image/png'
```

Для выбора элементов только с заданным текстом также необходимо передать в параметр text искомый текст. Также передача заданного текста позволит запустить прогон всех стратегий.

```
curl -X 'POST' \
  'http://192.168.88.102:9464/find_by?type_ui=TextElement&text=OK' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'file=@image.png;type=image/png'
```

Info

Распознавание текста – одна из самых сложных операций со множеством параметров и отдельных концепций.

Info

Для всех элементов нейронной сети кроме TextElement для получения текста необходимо передавать параметр text. Для типа TextElement параметр text необязателен. Переданный параметр text позволяет запустить прогон стратегий.

4.3.3 Получение текста на одном компоненте

Допустим, мы хотим получить текст уже на полученном компоненте. Этот функционал реализован в энподинте POST /find_text и альтернативном варианте POST /api/v2/find/text

/find_text

POST /find_text Find Text

Прогон изображения по всем стратегиям OCR без сохранения состояния

Входное изображение передается как byte-object multipart-data.

Полученное изображение асинхронно передается в OCR (N запросов = N стратегий),

В результате формируется выходной словарь {"strategy": "name"},

Parameters

Name	Description
suppression_loc boolean (query)	Not use Yolo localization *Default value* : false false
use_ocr_cache boolean (query)	Use ocr cache *Default value* : false false

Request body required

file * required
string($binary) Image to process

Прежде чем перейти к параметрам запроса, необходимо уточнить, как именно локализуется и распознается текст.

1) Локализация с помощью OCR

В этом случае локализация и распознавание текста ложится целиком на OCR, мы не уточняем локализацию текста с помощью других технологий.

2) Локализация с помощью нейронной сети и распознавание OCR

Также существует возможность сначала более точно определять границы текста, и затем уже уточненное изображение передать на обработку OCR. В большинстве случаев точность распознавания текста при этом повышается.

Сам же метод принимает два query-параметра:

- yolo_loc - использовать ли локализацию текста с помощью нейронной сети.
- По умолчанию *True*.
- use_ocr_cache - использовать ли кэш. Если данное изображение уже подвергалось обработке с данными параметрами, при use_ocr_cache результаты будут получены из кэша. Это значительно ускоряет время работы. Не стоит беспокоиться, что кэш не даст получить актуальные данные: изображение идет в обработку при любом его изменении или же изменении входных параметров.
- token - токен. Используется для выбора модели сети локализации текста, исходя из конфигурации токена.

Info

Само же изображение, как и в методе POST find_by, идет как multipart-form/data.

Пример запроса:

```
curl -X 'POST' \
  'http://192.168.88.102:9464/find_text?yolo_loc=true&use_ocr_cache=false' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'file=@canvas.png;type=image/png'
```

На выходе мы получим словарь JSON components, где значением будем список словарей с полями:

- *name* - наименование стратегии
- *text* - распознанный текст, полученный с использованием стратегии, указанной в *name*

Code	Details
200	Response body


```
{
  "components": [
    {
      "name": "find_text",
      "text": "ок"
    },
    {
      "name": "find_button",
      "text": "ок"
    },
    {
      "name": "find_text_bbox",
      "text": "ok"
    },
    {
      "name": "find_text_bbox_default",
      "text": "пw оу сти, 2"
    },
    {
      "name": "find_with_threshold",
      "text": "к сти, 2"
    },
    {
      "name": "main",
      "text": "ок"
    }
  ]
```


Response headers

```
content-length: 268
content-type: application/json
date: Wed,15 Dec 2021 11:33:06 GMT
server: uvicorn
```

Info

/api/v2/find/text

Метод, аналогичный /find_text с отличием в том, что заменяет собой два запроса различными значениями параметра yolo_loc (**True** и **False**), затрачивая при этом меньше времени за счет асинхронности запросов к OCR, а также возвращает список только уникальных значений полученных с прогона стратегий.

Таким образом, этот метод просто облегчает взаимодействие и за один запрос выполняет двойной функционал, но с уменьшением по времени, как если бы посылали на /find_text два различных запроса.

Принимает такие же параметры как /find_text, кроме параметра yolo_loc

Пример запроса

```
curl -X 'POST' \
  'http://192.168.88.102:9464/api/v2/find/text?use_ocr_cache=false' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'file=@canvas.png;type=image/png'
```

5 Сессионность и конфигурация

Для удобства отслеживания прохождения теста в UI NeuroMoon существует возможность разделения прогонов по сессиям. Для этого необходимо получить токен. Так как для разных проектов скорее всего потребуется различные версии нейронных сетей, то система также предоставляет механизм конфигурации версий. Механизм конфигурации является частью механизма сессионности.

Использование токена не является обязательным. Результаты запросов без токена также доступны в UI NeuroMoon

Info

Токен является данными для входа в NeuroMoon. Для того чтобы увидеть результаты запроса без токена, необходимо оставить пустым окно ввода.

5.1 Получение токена

Время жизни токена составляет 2 часа на его **бездействие**. С каждым запросом на эндпоинты в которых имеется параметр token, время жизни продлевается на два часа относительно времени совершения запроса. Если за 2 часа не было совершено никаких запросов, то токен закрывается и запросы с ним будут уже недействительны.

Для получения токена существует эндпоинт POST /token. Метод не имеет query-параметров, но имеет необязательное тело запроса, которое служит для конфигурации.

Пример получения токена без конфигурации

```
curl -X 'POST' \
  'http://192.168.88.102:9464/token' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d ''
```

Ответ:


```
{
  "token": "e3f83b5e-e001-4b36-bc18-823ac33c6b58",
  "time_created": "2022-01-13T11:13:43.887283",
  "time_finished": "2022-01-13T13:13:43.887283",
  "is_closed": false,
  "config": {
    "model_version": 2,
    "model_localize_version": 1
  }
}
```


Значение поля token из ответа используется как значение query-параметра в функциональных запросах. time_finished показывает время, когда токен будет закрыт (без использования запросов).

Warning

Срок возможного бездействия токена, когда он еще будет действителен, составляет **2 часа**. С каждым функциональным запросом время жизни будет продлеваться на 2 часа относительно времени сделанного запроса.

5.2 Конфигурация

Сконфигурировать можно версии модели детекции компонентов (основная НС) и модели локализации текста. По умолчанию для основной модели используется версия 2, для модели локализации текста версия 1.

Для того чтобы поставить другую версию модели, необходимо при запросе получения токена отправить конфигурационный словарь в теле запроса с требуемыми значениями версий.

```
{
  "model_version": 2,
  "model_localize_version": 1
}
```

Например, для установки основной модели на 3 версию, без изменении модели локализации текста необходимо сформировать следующий запрос:

```
curl -X 'POST' \
  'http://192.168.88.102:9464/token' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "model_version": 3
}'
```

Ответ:

```
{
  "token": "fc9caa25-c83c-445a-8e78-d5bf9f8f990f",
  "time_created": "2022-01-13T11:30:10.599988",
  "time_finished": "2022-01-13T13:30:10.599988",
  "is_closed": false,
  "config": {
    "model_version": 3,
    "model_localize_version": 1
  }
}
```

В дальнейшем при всех запросах, где будет указан полученный токен, детекция компонентов будет производиться с помощью модели версии 3

Info

Информация о формате запроса и выходных данных также имеется в Swagger.

6 Распознавание текста

Текст на компонентах изображения распознается с помощью технологии OCR. В качестве OCR используется open-source решение Tesseract. Однако Tesseract является очень требовательным к качеству изображения (не менее 300 dpi, отсутствие шумов, сглаженный текст).

Info

Подробнее про OCR Tessract можно найти здесь

Изображения и вырезанные компоненты зачастую не отвечают требованиям, поэтому для улучшения качества производится предварительная обработка. Параметры использования этой предобработки и режимы работы самого OCR (язык, режим сегментации страницы) вынесены в отдельные параметры. Из-за большого числа этих параметров и различных результатов их комбинаций для управлением повышения качества OCR была введена сущность **Стратегия**.

6.1 Стратегии

Стратегия является просто набором параметров обработки изображений и настройки работы OCR, которые были подобраны эмпирическим путем и обеспечивают относительно (других комбинаций параметров) работоспособность OCR.

При распознавании текста на компонентах возможно передать только имя стратегии, и тогда в качестве настроек OCR будут использоваться настройки выбранной стратегии.

К сожалению, использование стратегии и привязка стратегии к определенным элементам как требует и предварительного перебора для нахождения оптимальной, так и уменьшает гибкость. Чтобы хоть немного сгладить эти моменты, в API доступно задать прогон всех стратегий, что позволяет покрыть больше случаев, чем при использовании только одной стратегии.

6.1.1 Механизм работы стратегий при распознавании элементов

Распознавание текста и использование стратегий применяется только в 3 отдельных случаях:

1. find_by - полное распознавание, в том числе и с текстом
2. find_text - поиск текста на уже вырезанном компоненте
3. Показ результатов прогона всех стратегий в UI NeuroMoon
4. /api/v2/find/text - Альтернатива find_text для удобства использования

6.1.1.1 Find_by

За использование стратегии отвечает необязательный query-параметр *ocr_strategy*.

Info

Стоит помнить, что в **find_by** распознавание текста будет производиться, только если передан запрашиваемый текст или тип TextElement. Описание концепций распознавания

Если этот параметр не передать, то будет произведен прогон стратегий последовательно до тех пор, пока не будет найдено совпадение или не пройдут все стратегии.

Warning

Последовательный прогон длится гораздо дольше, чем асинхронный (кроме тех случаев, когда результат было найден на первой стратегии). В будущем, возможно, везде будет использован асинхронный прогон всех стратегий. В таком случае отсутствие стратегии является худшим методом прогона всех стратегий, который вернет первое попавшееся совпадение.

Если явно указана стратегия, будет использоваться только указанная стратегия.

Если указан текст (для TextElement) и передано значение all, то будет произведен асинхронный прогон всех стратегий. Будет возвращен лучший результат (для всех типов в **find_by**).

Info

Для TextElement, так как выходные bbox могут различаться, также будет произведен поиск лучшего результата, но одинаковые результаты будут искать не по точным координатам, а по возможным пересечениям и наложениям, что позволит на один компонент выдавать один лучший результат, а не множество.

Warning

При поиске TextElement с помощью локализации Tesseract при прогоне всех стратегий не будет задействованы стратегии **find_text** и **main**. Это связано с параметром сегментации страницы в этих стратегиях.

6.1.1.2 Find_text и прогон стратегий в UI

Find_text и прогон стратегий в UI реализуют практически одно и тоже и обладают одинаковым поведением. Так как в find_text и в прогоне стратегий UI передается уже вырезанный компонент, то будут возвращены результаты всех стратегий.

```
{
  "components": [
    {
      "name": "find_text",
      "text": "ок"
    },
    {
      "name": "find_button",
      "text": "ок"
    },
    {
      "name": "find_text_bbox",
      "text": "ok"
    },
    {
      "name": "find_text_bbox_default",
      "text": "пw оу сти, 2"
    },
    {
      "name": "find_with_threshold",
      "text": "к сти, 2"
    },
    {
      "name": "main",
      "text": "ок"
    }
  ]
```


Info

Формат словарей списка *components* :

- *name* - наименование стратегии
- *text* - распознанный текст, полученный с использованием стратегии, указанной в *name*

Результаты прогона UI и работы find_text на одном компоненте должно быть одинаковые. Также они должны быть одинаковыми с результатами работы **find_by,** если ищется не TextElement

Warning

Результаты, полученные с **find_by** с поиском TextElement, а затем уже выбранные оттуда отдельные компоненты, пропущенные через **find_text** и UI, могут и, скорее всего, будут **отличаться** от того, что выдал **find_by**. Это связано с различными механизами работы: TextElement (с локализацией tesseract) обрабатывает исходную картинку целиком, а в **find_text** обрабатывается только один компонент

6.1.1.3 /api/v2/find/text

Возвращает список уникальных значений, полученных после прогона стратегий без и с использованием нейронной сети для локализации текста. Аналогичен двум запросам на /find_text с различными значениями параметра yolo_loc, затрачивая при этом меньше времени за счет асинхронности запросов к OCR.

6.1.2 Параметры стратегий

lang: str = "rus"
psm: int = 6
zoom: float = 1.5

cut: Union[int, float] = 0
cut_policy: str = "left_right"
thresholding: bool = False
text_post_processing: bool = True
remove_halo: bool = False
use_cache: bool = False

6.1.3 Наименования стратегий

find_text, find_button, find_text_bbox, find_text_bbox_default, find_with_threshold

Info

Увидеть список стратегий и их параметры можно в блоке OCR Strategy в Swagger

7 Сопоставление шаблонов и парсинг таблиц

7.1 Метод find_icon

Зачастую не всегда можно определить компонент по его типу или по тексту (например, иконки). Для таких случаев можно воспользоваться поиском по шаблону. Шаблоном является вырезанное (пользователем) изображение (иконка и прочее).

Для поиска по шаблону существует эндпоинт POST /find_icon. Метод принимает следующие параметры и тело запроса.

- *threshold* - порог уверенности совпадения шаблона на изображении
- *use_cache* - использовать ли кэш
- *token* - токен для сессионности

Тело запроса:

- *image* - изображение, в котором будет произведен поиск
- *template* - шаблон

Пример запроса (curl):

```
curl -X 'POST' \
  'http://192.168.88.102:9464/find_icon?threshold=0.768&use_cache=false' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'image=@2021-09-29_14-51-37.png;type=image/png' \
  -F 'template=@dsf99275441699988714.png;type=image/png'
```

Пример ответа:


```
{
  "components": [
    {
      "name": "Icon",
      "bbox": {
        "x_min": 290,
        "y_min": 573,
        "x_max": 305,
        "y_max": 587
      },
      "attributes": null,
      "text": null,
      "id": 0
    }
  ]
}
```

7.2 Таблицы

Система также предоставляет определение структуры таблицы с помощью анализа контуров. Доступны 3 эндпоинта, способных определить ячейки, столбцы и строки. Все 3 метода схожи между собой.

1.POST /table/rows - позволяет определить строки в таблице. Имеет следующие параметры: - *row_idx* - массив индексов строк, которые необходимо вернуть - *token* - Токен сессионности - *table* - Изображение таблицы

2.POST /table/columns - позволяет определить столбцы в таблице. Имеет следующие параметры: - *col_idx* - массив индексов столбцов, которые необходимо вернуть - *token* - Токен сессионности - *table* - Изображение таблицы

3.POST /table - позволяет определить ячейки в таблице. Имеет следующие параметры: - *row_idx* - массив индексов строк, которые необходимо вернуть - *col_idx* - массив индексов столбцов, которые необходимо вернуть - *token* - Токен сессионности - *table* - Изображение таблицы

Пример запроса(curl):

```
curl -X 'POST' \
  'http://192.168.88.102:9464/table?row_idx=0&row_idx=1&col_idx=1' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'table=@захаров.png;type=image/png'
```

Этот запрос вернет ячейки (координаты и id столбца и строки), которые располагаются в 0 и 1 строки и в 1 столбце

Пример ответа:


```
{
  "components": [
    {
      "name": "Cell",
      "bbox": {
        "x_min": 27,
        "y_min": 6,
        "x_max": 596,
        "y_max": 31
      },
      "attributes": null,
      "text": null,
      "row_idx": 0,
      "col_idx": 1
    },
    {
      "name": "Cell",
      "bbox": {
        "x_min": 27,
        "y_min": 31,
```

```
      "x_max": 596,
      "y_max": 47
    },
    "attributes": null,
    "text": null,
    "row_idx": 1,
    "col_idx": 1
  }
 ]
}
```


Info

Определение структуры таблиц не возвращает текст.

8 Обучение и подготовка

8.1 Получение данных

Для работы системы необходимо проверить и при необходимости (скорее всего) переобучить нейронную сеть. Для этого следует снять некое количество скриншотов изображения (минимум 50). Скриншоты изображения должны охватывать наибольшее количество различных компонентов. Возможен подход, когда пользователь вручную проходит ТК и снимает скриншоты с каждого шага.

Снятые скриншоты необходимо разметить. Для разметки скриншотов существует платформа CVAT. В помощь разметки введены функции превадрительной автоматической разметки с помощью лямбда-функций Nuclio. В этом случае изображения сначала размечаются уже существующими сетями и правятся пользователем.

Перед обучением производится анализ скриншотов, при необходимости производится аугментация изображений. После обучения рассматриваются результаты обучения, метрики, изображения, где нейронная сеть отработала хуже всего, после чего принимаются решения по увеличению эффективности.

Процесс обучения логируется в систему W&B в реальном времени, что позволяет оперативно наблюдать за процессом обучения и при необходимости вносить изменения.

Веса также сохраняются на платформе W&B.

Результаты обучения и значения метрик прогона на тестовом наборе для простоты записываются в Google Таблицы.

8.2 Добавление модели в сервис

После обучения модели для работы с ней необходимо добавить полученные веса в сервис Yolov5.

Сервис способен загружать новые модели, которые находятся в хранилище Minio, поэтому необходимо первым делом загрузить веса в хранилище.

Сделать это можно как с помощью UI, так и из кода.

Для загрузки через UI (как самый простой вариант) необходимо перейти на Minio и загрузить модельку в директорию models. Внутри директории можно создавать новые папки, куда можно загрузить модель.

После загрузки на Minio необходимо сказать сервису, чтобы он ее выгрузил и сделал активной. Для этого существуют два энподинта, обратиться к ним можно как с помощью запросов (например, curl, или requests), так и через *Swagger*.

8.2.1 Выгрузка модели с хранилища

Необходимо отправить запрос POST /models/add?name={name}, где {name} имя вашей загруженной модели

8.2.2 Активация модели

За активацию модели отвечает запрос PUT /models/activate/{version}?name={name}

Здесь {version} - это адрес ресурса, через который можно будет обратиться к модели. Версия является целым числом, не менее 0. Вы можете указать уже существующий ресурс, и тогда модель будет перезаписана. Указание версии модели не терпит пропусков. Если вы указаываете число, на котором модели еще нет, модель будет записана на последнее место.

Например, у нас в сервисе уже активировано две модели, на 1 и 2 версии. Мы хотим поставить новую модель на ресурс с индексом 3 и для этого отправляем запрос PUT /models/activate/3?name={name}. В ответ (если было задано корректное имя модели) придет ответ с успехом и указанием, на каком ресурсе находится модель.

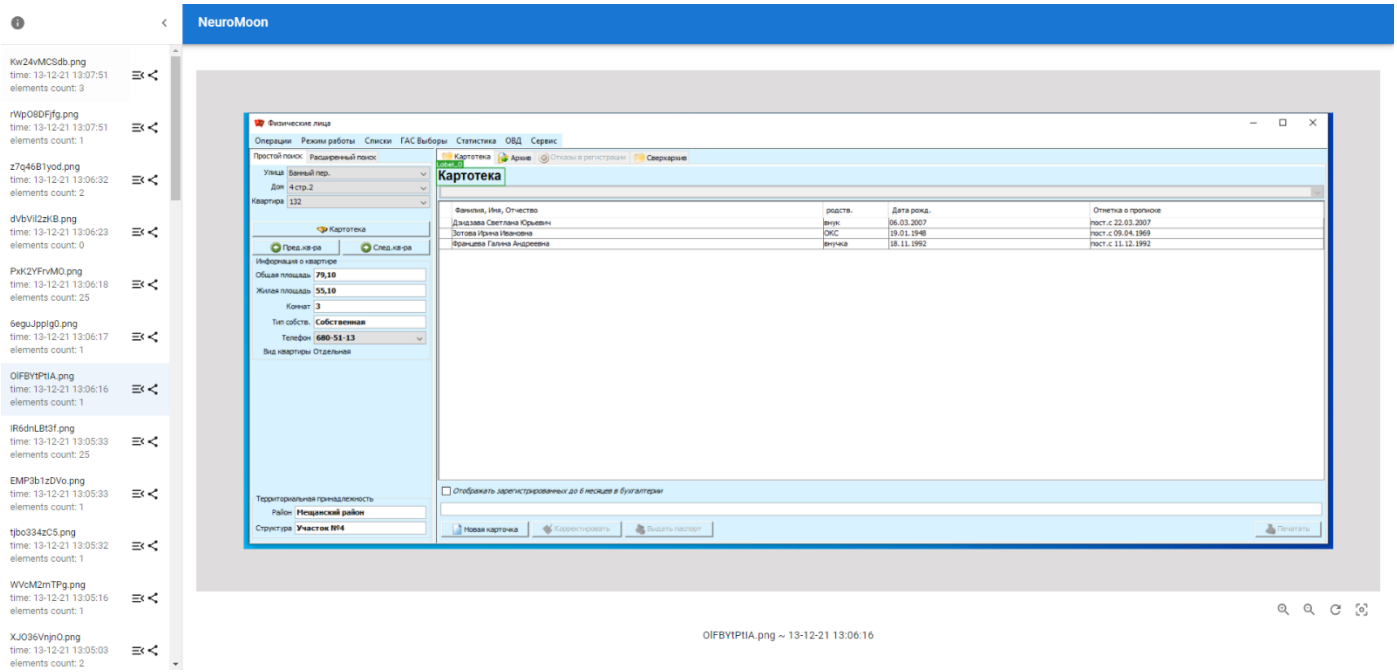
Теперь у нас занято 3 ресурса (1, 2, 3), и мы хотим добавить модель на 5 ресурс. В этом случае модель добавится на 4 индекс, соответствующая версия будет указана в ответе.

Если же модель не найдена, в ответ придет следующее сообщение:

```
{
  "msg": "Model not found",
  "name_model": "bestv2.pt"
}
```

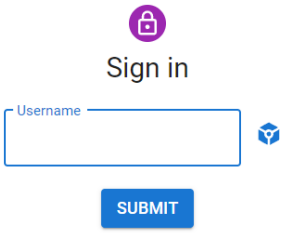
9 NeuroMoon

NeuroMoon - система для визуализации работы нейронной сети



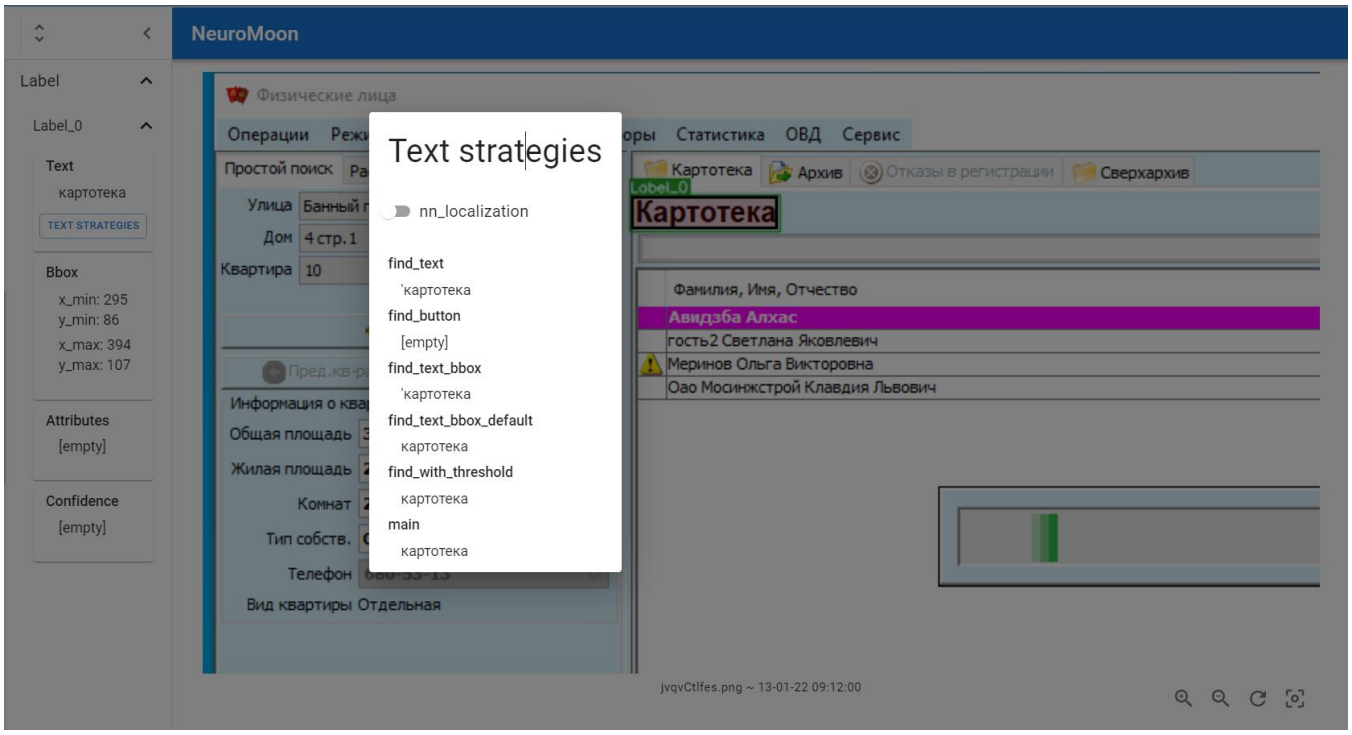
9.1 Вход в NeuroMoon

Окно входа



Данными для входа является токен. Можно войти и без токена, тогда вы попадаете в общий поток запросов. Токен можно сгенерировать и на странице входа, а затем использовать его в запросах, или получить напрямую через API. [Подробнее о токенах...]

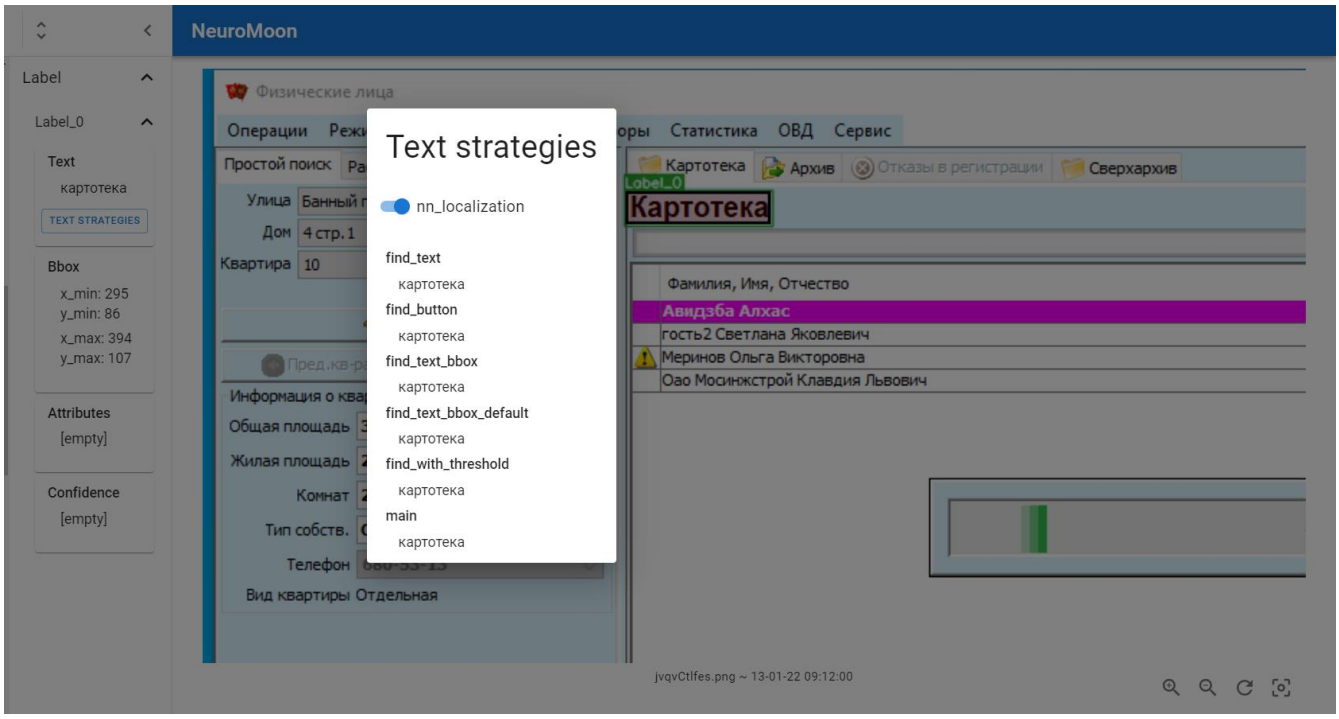
9.2 Информация о компоненте



Переключатель позволяет увидеть результаты, когда для локализации текста используется нейронная сеть, а не OCR.

Info

OCR на самом деле тоже нейронная сеть, но опустим здесь это, чтобы не вводить в заблуждение, и будем называть нейронной сетью технологию, не связанную с OCR.



Приложение 1. Регламент поддержания жизненного цикла программного обеспечения

Поддержание жизненного цикла ПО «NeuroControl – платформа для автоматизации тестирования любых мобильных и desktop приложений с использованием методов Computer Vision и нейросетей» обеспечивается за счет его сопровождения и проведения обновлений в соответствии с собственным планом разработки ПО. В рамках технической поддержки ПО оказываются следующие услуги:

- помощь в настройке и администрировании;
- пояснение функционала модулей ПО, помощь в эксплуатации ПО;
- предоставление документации.

Устранение неисправностей, выявленных в ходе эксплуатации программного обеспечения

Неисправности, выявленные в ходе эксплуатации ПО, могут быть исправлены следующим образом:

- Массовое автоматическое обновление компонентов ПО;
- Устранение неисправности по запросу пользователя на адрес службы технической поддержки: lanit_exp@lanit.ru.

Совершенствование программного обеспечения

ПО регулярно дорабатывается специалистами ООО «ЛАНИТ ЭКСПЕРТИЗА» в соответствии с собственным планом разработки ПО.

Пользователь может самостоятельно повлиять на совершенствование ПО, для этого предложение по усовершенствованию ПО необходимо направить на адрес службы технической поддержки: lanit_exp@lanit.ru

Предложение будет рассмотрено и, в случае признания его эффективности, будет добавлено в план разработки ПО.

Техническая поддержка

Для оказания технической поддержки ПО выделен адрес службы технической поддержки: lanit_exp@lanit.ru. По срочным вопросам, связанным с работой ПО, можно позвонить на номер +7 (495) 967-66-50 доб. 16397.

Информация о персонале, необходимом для обеспечения поддержки работоспособности ПО

Администраторы ПО «NeuroControl – платформа для автоматизации тестирования любых мобильных и desktop приложений с использованием методов Computer Vision и нейросетей» должны обладать навыками администрирования операционных систем семейств Linux, а также:

- навыками программирования на языке Python – для работы с бекэнд частью решения;
- навыками программирования на языке Java Script – для работы с фронтенд частью решения;
- навыками программирования на языке Java (Open JDK) – для работы с фреймворками автоматизации тестирования;
- опытом работы с Docker и Docker Compose.

Для работы администраторы системы должны изучить Руководство администратора «NeuroControl – платформа для автоматизации тестирования любых мобильных и desktop приложений с использованием методов Computer Vision и нейросетей».

Гарантийное обслуживание программного обеспечения осуществляется силами штатных специалистов службы технической поддержки ООО «ЛАНИТ ЭКСПЕРТИЗА». Команда службы поддержки включает 2 специалистов: Разработчик и Ведущий разработчик.

Техническая поддержка программного обеспечения осуществляется силами штатных специалистов службы технической поддержки ООО «ЛАНИТ ЭКСПЕРТИЗА». Команда службы поддержки включает 2 специалистов: Разработчик и Ведущий разработчик.

Модернизация программного обеспечения осуществляется силами штатных специалистов службы технической поддержки ООО «ЛАНИТ ЭКСПЕРТИЗА». Команда службы поддержки включает 2 специалистов: Разработчик и Ведущий разработчик.

Контакты службы технической поддержки ПО

Адрес: 129075, г. Москва, Мурманский проезд, д.14, корп.1, техн. эт. 2, помещ. 58

Почта: lanit_exp@lanit.ru

Телефон: +7 (495) 967-66-50 доб. 13397