



**ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ
ФИЛИАЛ ПЛОВДИВ**

ФАКУЛТЕТ ПО ЕЛЕКТРОНИКА И АВТОМАТИКА

ПРОЕКТИРАНЕ И РАЗРАБОТВАНЕ НА СИСТЕМА ЗА РАЗПОЗНАВАНЕ НА ОБЕКТИ В ИЗОБРАЖЕНИЕ

Любомир Ламбрев , фак. номер 614973

Email: thelubo1@abv.bg

Github: <https://github.com/thelubo/Master-s-thesis>

Съдържание

1	Промяна на имейл
Мотивация, основна цел, основни задачи	Работа с изображения
2	Детекция и сегментация
Концептуален модел на подсистемата за обучение	Запазване в база данни
Концептуален модел на системата за разпознаване на обекти	История и статистика
Архитектура на системата	5
Софтуерни инструменти	Приложимост на разработената работа
3	7
Псевдокод на подсистемата за обучение	Заклучение
Псевдокод на подсистемата за разпознаване на обекти	
Експеримент и резултати	
4	
Система за вход	
Промяна на парола	

Мотивация

Мотивация:

- Мотивацията зад създаването на такава система е огромният ръст на визуални данни, които се генерират ежедневно в различни индустрии. Неспособността тези данни да бъдат обработвани и анализирани достатъчно бързо може значително да възпрепятства иновациите и ефективността в много сектори по света.
- Затова нашата система помага на потребителите ефективно да откриват и анализират обекти в изображения, като по този начин значително намалява ръчния труд и позволява по- бързо вземане на решение.

Цел и задачи

Цел:

- Да се проектира и реализира уеб-базирана система за разпознаване на обекти в изображения.

Задачи:

- Качване или поставяне на изображения
- Анализиране и разпознаване на конкретни обекти
- Визуализиране на откритите обекти
- Съхраняване на резултатите и изображенията в базата данни
- Предоставяне на история и визуализация на предишни детекции

Основни етапи за работа на системата

Разпознаването на обекти се състои от 2 основни етапа:

- Първи етап – Обучение
- Втори етап – Разпознаване на обекти (взимане на решение)

Концептуални модели



Концептуален модел на подсистемата за обучение



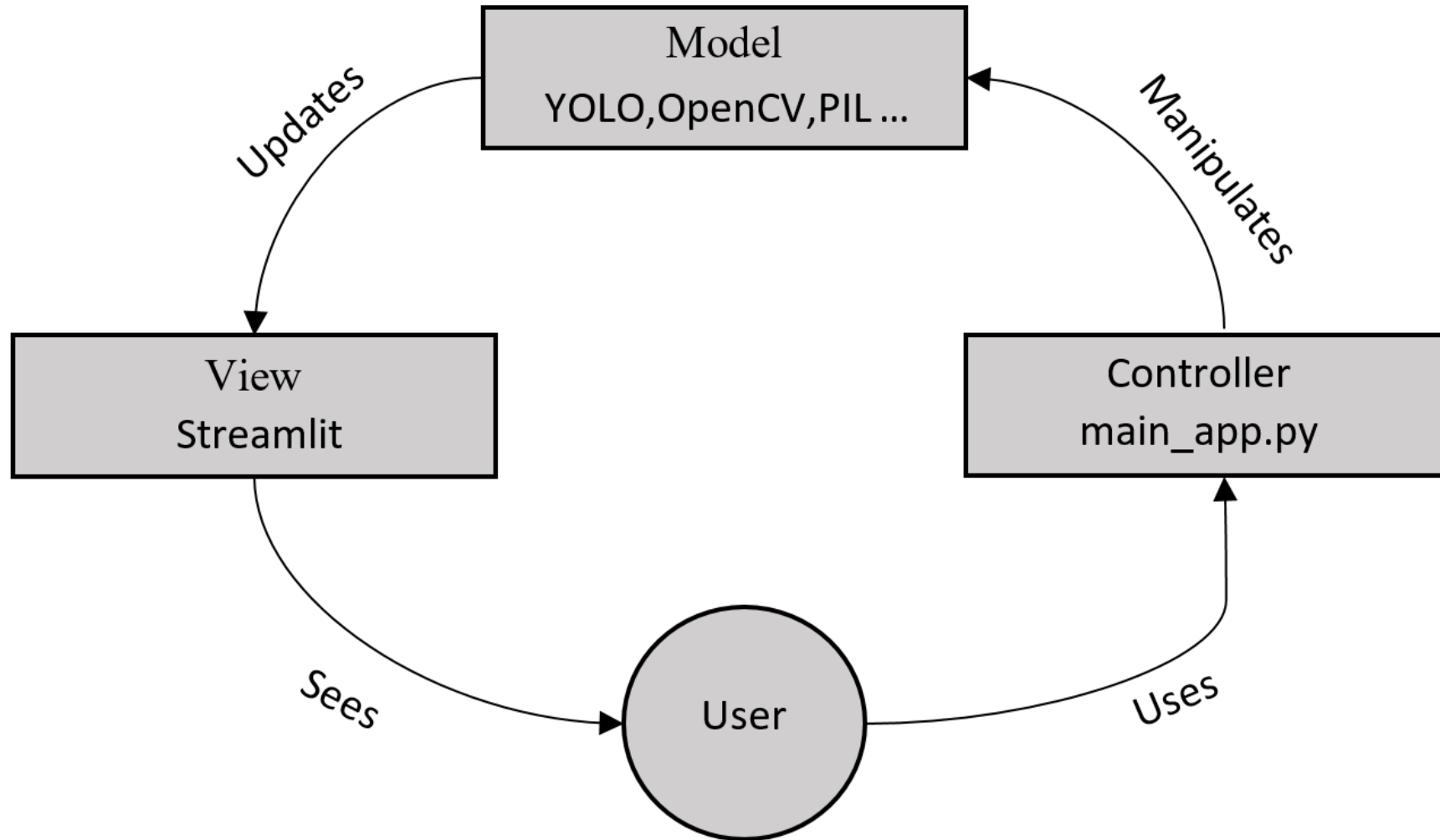
Концептуален модел на подсистемата за разпознаване на обекти

Архитектура на системата

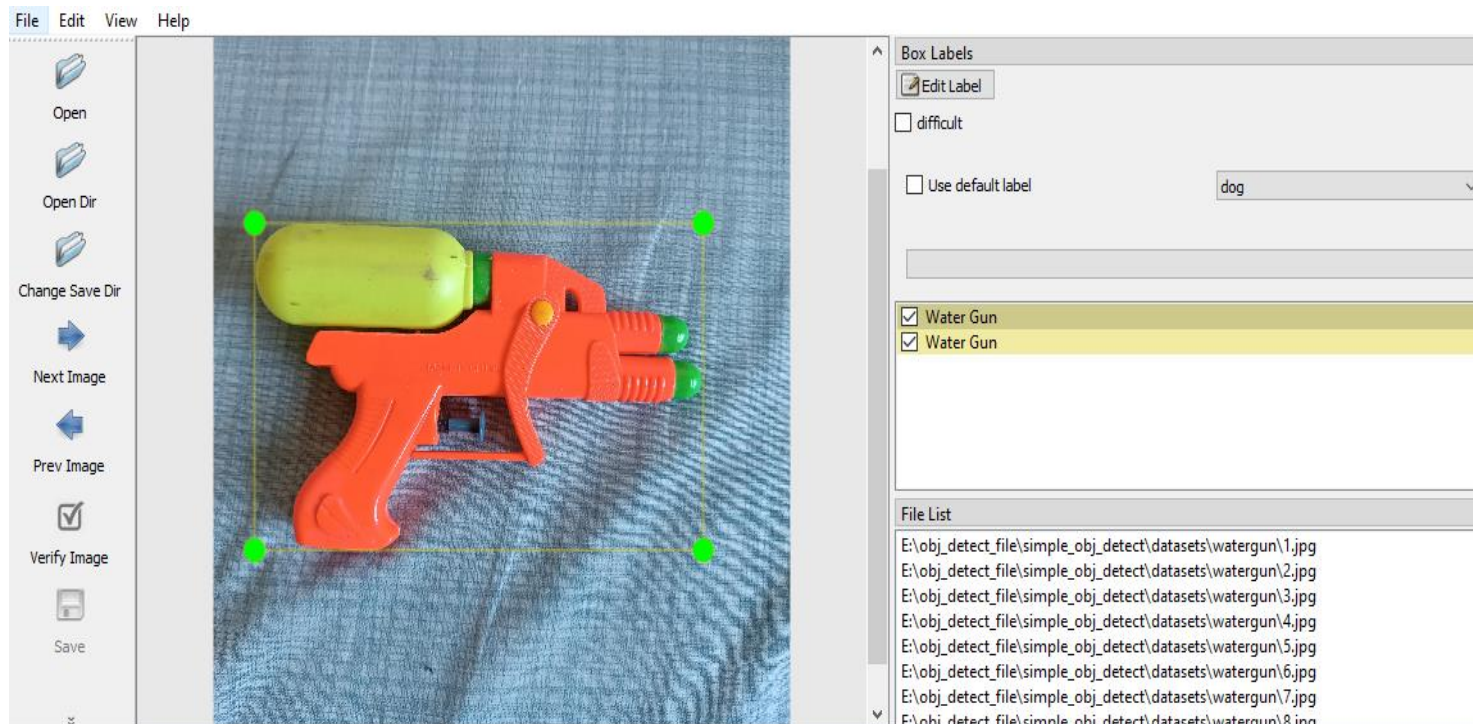
Използваме MVC като системна архитектура, което означава, че кодът ще бъде разделен на 3 части:

- **Модела** - Работи с данни, които се записват преди и след обработката на изображението.
- **Изгледа** - Streamlit интерфейс който позволява на потребителя да качва изображения, настройва параметри и вижда резултати.
- **Контролера** - Изпълнява алгоритмите за обработка и детекция, като свързва действията на потребителя с модела и изгледа.

Model-View-Controller (MVC) в контекста на системата



Подготовка на данни чрез програмата Labelling



Съхранение и интеграция на обучен модел (псевдокод)

Algorithm save_trained_model is:

Вход: обучен модел (final_model), път за запазване (path)

Изход: файл с обучен модел (.h5 формат)

Стъпка 1: Дефинираме път за запазване:

path = '/obj_detect_file/simple_obj_detect/working/simple_object_detection.h5'

Стъпка 2: Извикваме save(path) върху final_model:

- сериализираме структурата на невронната мрежа
- запазваме обучените тегла
- запазваме настройките за компилация (optimizer, loss, metrics)

Стъпка 3: Създаваме файл .h5 на зададеното място

Стъпка 4: Връщаме готов модел

Въвеждане на данни (псевдокод)

Algorithm image_upload is:

Вход: избор на източник на изображение (image_source)

Изход: качено изображение (source_image)

Стъпка 1: Проверяваме дали image_source е равно на "Upload an image".

Стъпка 2: Ако условието е вярно, стартираме компонент за качване на изображение (file_uploader).

Стъпка 3: Ограничаваме типовете файлове до jpg, png, jpeg, bmp и webp.

Стъпка 4: Указваме уникален ключ „file_uploader“.

Стъпка 5: Ако потребителят качи изображение, запазваме файла в променливата source_image.

Стъпка 6: Връщаме стойността на source_image.

Модул за вход

User Authentication

Login

Register

Forgot Password

Forgot Username

Login

Username

Password



Login

Промяна на парола

Change Password

Current Password



New Password



Confirm New Password




Change Password

Промяна на имейл

Change Email

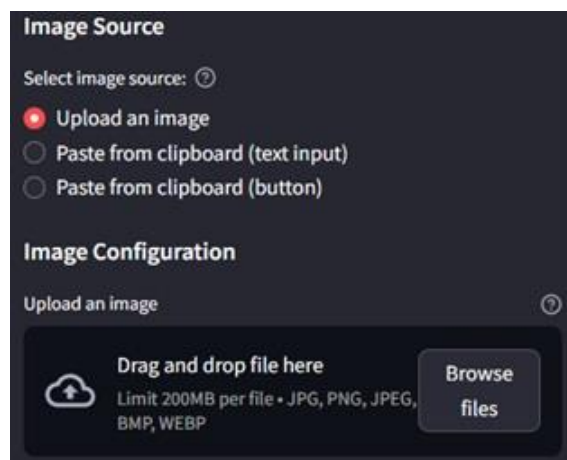
Password



New Email

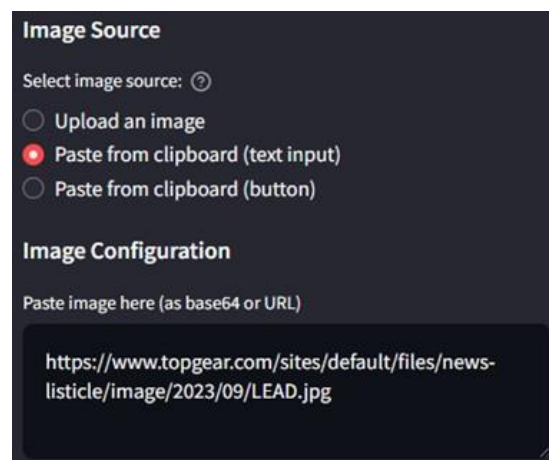
Change Email

Модул за работа с изображения



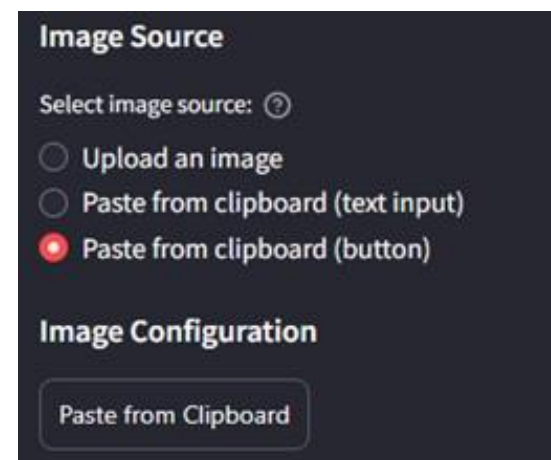
The screenshot shows the 'Image Source' section with three radio buttons: 'Upload an image' (selected), 'Paste from clipboard (text input)', and 'Paste from clipboard (button)'. Below this is the 'Image Configuration' section with the label 'Upload an image' and a question mark icon. It features a cloud icon with an upward arrow, the text 'Drag and drop file here', a file size limit 'Limit 200MB per file • JPG, PNG, JPEG, BMP, WEBP', and a 'Browse files' button.

*Качване на изображение
чрез upload*



The screenshot shows the 'Image Source' section with three radio buttons: 'Upload an image', 'Paste from clipboard (text input)' (selected), and 'Paste from clipboard (button)'. Below this is the 'Image Configuration' section with the label 'Paste image here (as base64 or URL)'. It contains a text input field with the URL 'https://www.topgear.com/sites/default/files/news-listicle/image/2023/09/LEAD.jpg'.

*Поставяне на изображение
чрез копираня адрес*



The screenshot shows the 'Image Source' section with three radio buttons: 'Upload an image', 'Paste from clipboard (text input)', and 'Paste from clipboard (button)' (selected). Below this is the 'Image Configuration' section with a 'Paste from Clipboard' button.

*Поставяне на изображение
чрез clipboard бутон*

Детекция и сегментация

Визуализация на разпознатите обекти в изображения

Deploy

Your account

Home

Change Password

Change Email

Logout

Model Configuration

Select Task Type: ?

YOLO General Detection

YOLO General Segmentation

YOLO Model (Specific)

TensorFlow Model (Specific)

Confidence Threshold

040100

Image Source

Select image source: ?

Upload an image

Paste from clipboard (text input)

Paste from clipboard (button)

Image Configuration

Upload an image


Image Detection

Statistics

Detection History

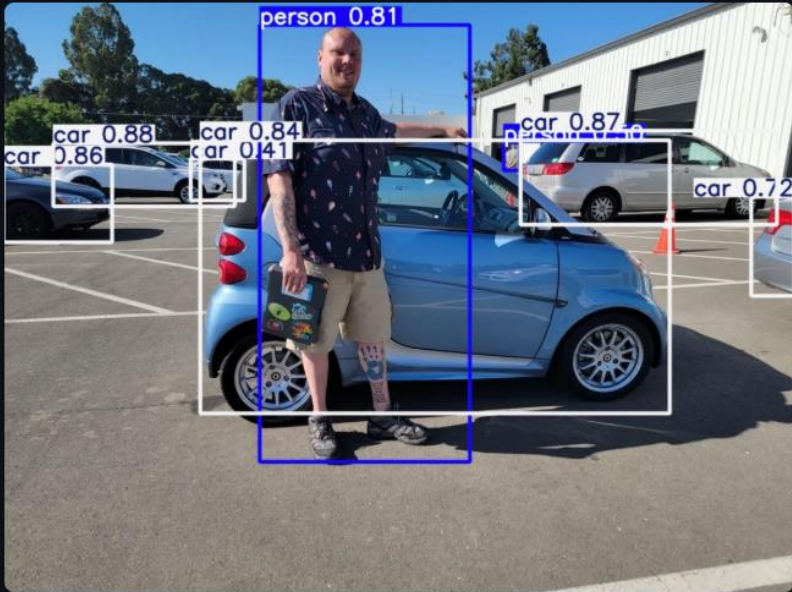
Overall Statistics

Input Image



Default Image - Upload or paste your own image to see detection results

Detection Results



Object	Confidence Score
person	0.81
car	0.88
car	0.86
car	0.84
car	0.41
car	0.87
car	0.72

Sample Detection - Upload or paste your own image to see live results

Структура на данни

В приложението структурата на данните е организирана така, че да осигурява функционалност за работа с потребители, изображения и резултати от детекция.

В основата си приложението използва MongoDB като база от данни, в която се съхраняват както регистрационните данни на потребителите, така и резултатите от извършените анализи на изображения.

Структура на документа при потребителски данни

При регистрация, данните на потребителите се съхранява в базата от данни под колекцията “users” , като документ съдържа следната информация:

- `_id` – уникален идентификатор на документа
- `user_id` - уникален идентификатор на потребителя
- `username` - потребителско име
- `email` - имейл
- `password` - криптирана парола

```
_id: ObjectId('68822eb5ea86712aee44f7c6')
user_id: "c95a2c46-4403-4c22-8af1-e774a0b303e2"
username: "user1"
email: "random1@abv.bg"
password: Binary.createFromBase64('JDJiJDEyJGRBdHJPQ0VGUGRPb2gzMU9nYjk5VXU1UjgvNnNB0C50c0d1a1hnSDVBN3I4Yll5ekI1Z1ll', 0)
```

Структура на документа при детекция

Всеки резултат от детекция се запазва като отделен документ. Документът съдържа метайнформация за изображението и конкретните открити обекти. Основните полета са:

- `_id` - уникален идентификатор на сесията
- `timestamp` - времето на извършване на детекцията
- `model_type` - използваният тип модел
- `confidence-threshold` - прагът на увереност
- `source` - източникът на изображението
- `image_name` - име на изображението, ако е качено от файл
- `image_resolution` - ширина и височина на изображението
- `object_count` - броят обекти от всеки клас
- `Objects` - списък с подробни данни за всяка детекция
- `user_id` - уникален идентификатор на потребителя
- `image_bytes` - изображението с нанесени маркировки, съхранено като base64 формат

Пример на структура на документа от MongoDB Compass

MongoDB Compass - object_detection/object_detection.person

Connections Edit View Collection Help

Compass

My Queries

CONNECTIONS (2)

Search connections

object_detection

- admin
- config
- local
- object_detection
 - Multiclass_objects
 - Water_Gun
 - airplane
 - bear
 - bicycle
 - bird
 - bus
 - car
 - carrot
 - dog
 - motorcycle
 - no_detections

users object_detection person

object_detection > object_detection > person

Documents (2) Aggregations Schema Indexes (1) Validation

Type a query: { field: 'value' } or [Generate query](#)

Explain Reset Find Options

ADD DATA EXPORT DATA UPDATE DELETE

25 1 - 2 of 2

```
{
  "_id": ObjectId('68bc6db3f0cd189e0624f374'),
  "timestamp": "2025-09-06T17:21:55.137+00:00",
  "model_type": "Detection",
  "confidence_threshold": 0.4,
  "source": "Paste from clipboard (button)",
  "image_name": "pasted_image",
  "image_resolution": {
    "width": 3000,
    "height": 4000
  },
  "object_counts": {
    "person": 1
  },
  "objects": [
    {
      "object_id": 0,
      "class": "person",
      "confidence": 0.4463236331939697,
      "box_xywh": [
        1500.185302734375,
        2017.769287109375,
        2987.7451171875,
        3964.46142578125
      ]
    }
  ],
  "user_id": "c95a2c46-4403-4c22-8af1-e774a0b303e2",
  "image_bytes": "/9j/4AAQSkZJRgABAQAAQABAAQ/2wBDAAgBGgcGBQgHBwcJCQgKDBQNDAsLDBkSEw8UHR..."
}
```

Структура на колекциите в базата от данни

За по-добра организация и по-бързо търсене документите се разпределят в различни колекции в зависимост от класа на откритите обекти.

Ако е засечен само един вид клас, например „person“ или „car“, документът се запазва в колекция съответстващ със името на обекта.

При наличие на повече от един клас, детекцията се записва в колекцията **Multiclass_objects**.

Ако не са засечени обекти, данните се запазват в колекция **no_detections**.

История на предишни детекции и статистики

Визуализация на изображения от минали детекции

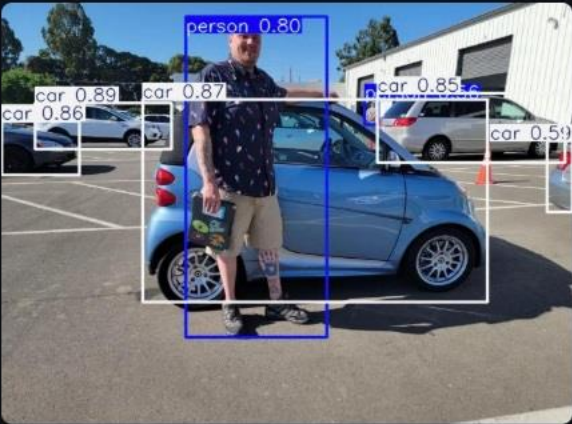
[Image Detection](#) [Statistics](#) [Detection History](#) [Overall Statistics](#)

Detection History

Filter by class name (optional)

Filter by source (optional)

1. `pasted_image`



Detected Image

Download

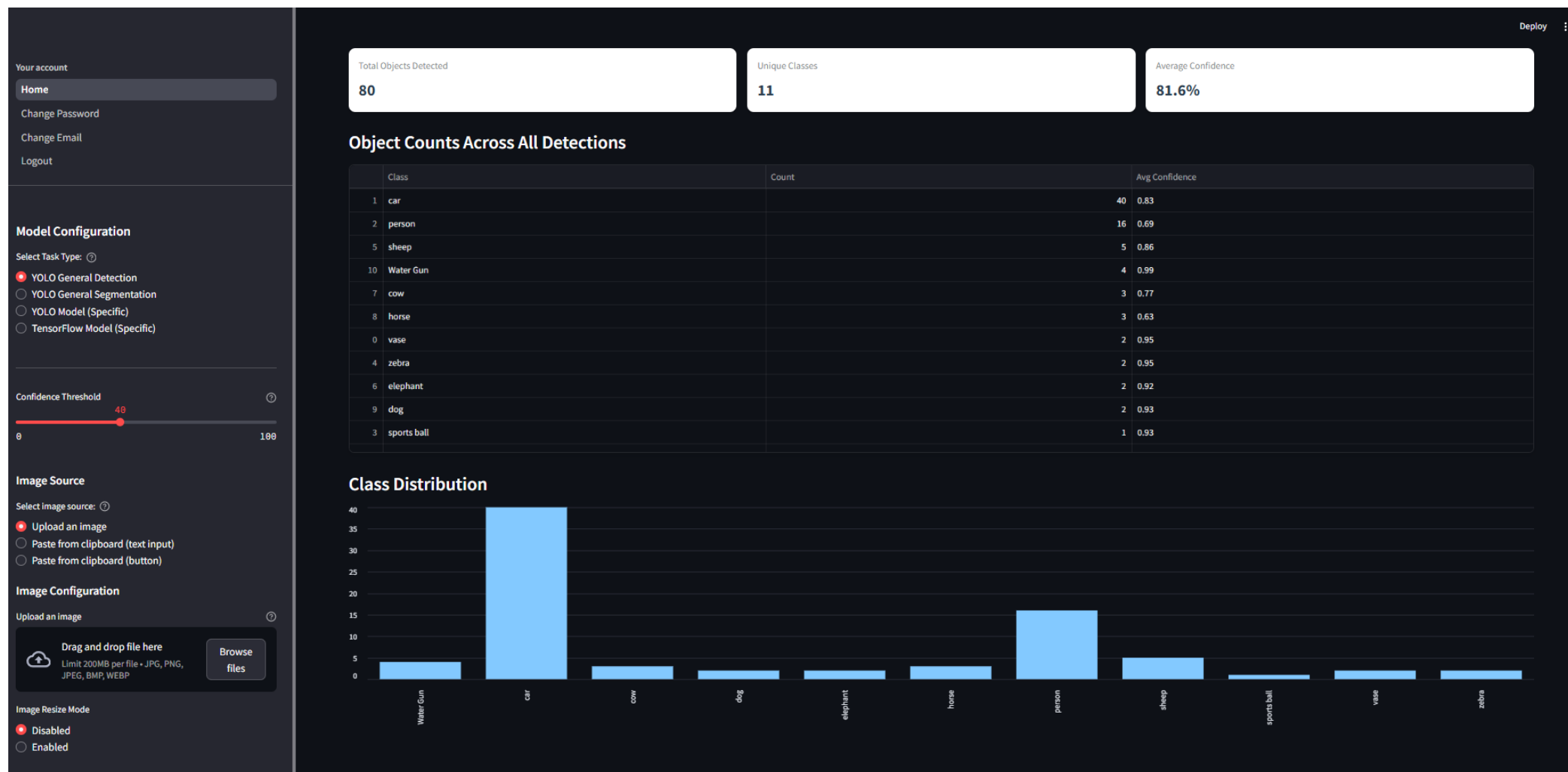
Information:

- **Date:** 2025-09-12 10:51:58 EEST
- **Model:** YOLO General Detection
- **Source:** Paste from clipboard (button)
- **Confidence Threshold:** 0.4

Detected Classes:

- car: 5
- person: 2

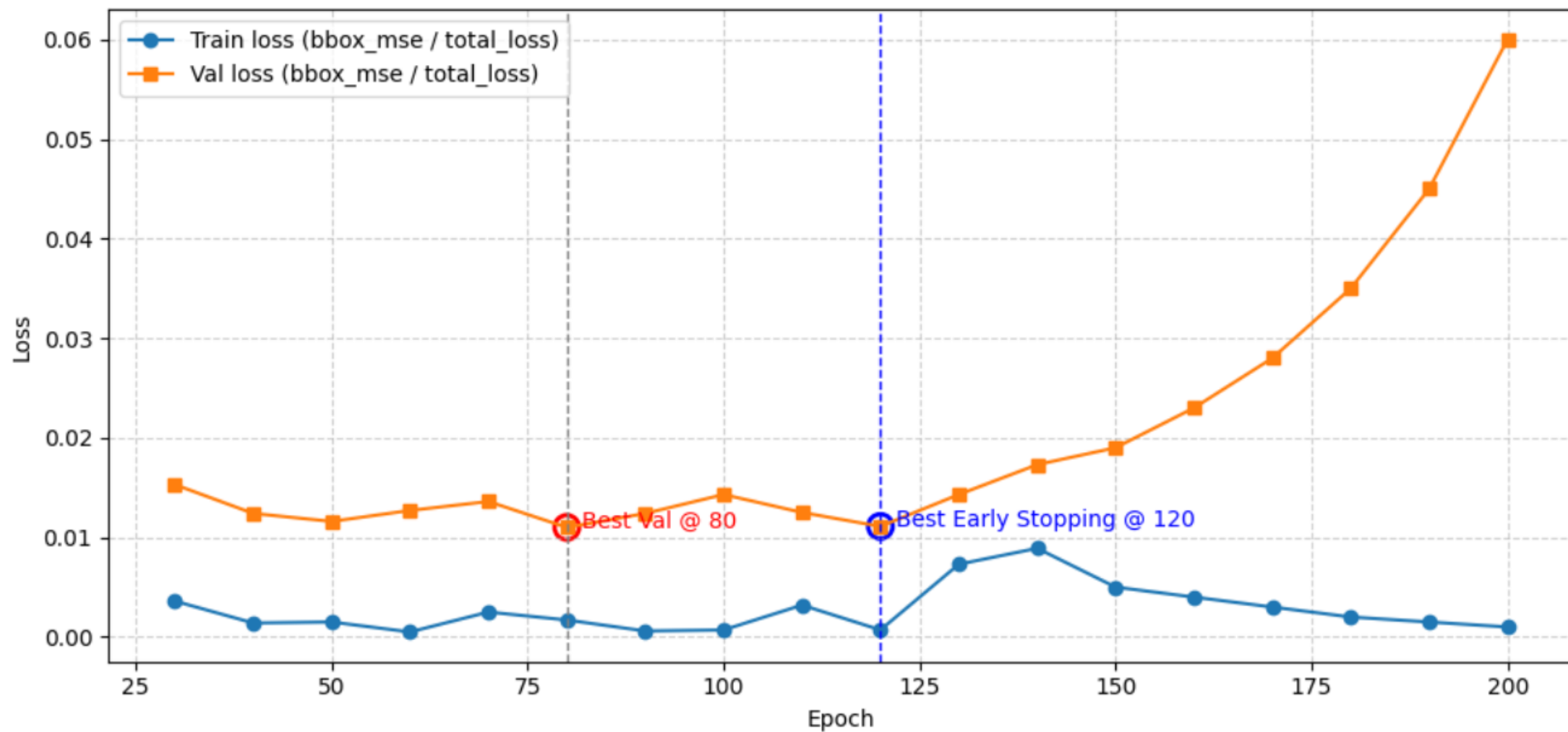
Визуализация на таблица с информация за всички извършени налични записи



Експеримент за свръхобучение (Overfitting)

- За всяка конфигурация на обучение (learner) се подава на входа набор от изображения и техните анотации, а на изхода се получава обучената конфигурация.
- За оценката на резултата се използва съотношението между train loss и validation loss функциите.
- Експериментът е проведен с брой епохи в интервала от 30 до 200, като се използва стъпка от 10 епохи. По този начин са получени 18 различни конфигурации на обучение. За всяка конфигурация са записани стойностите на тренировъчната и валидационната загуба.

Резултати от експеримента



Резултати от експеримента

	epoch	train_loss	val_loss	train/val_ratio
0	30	0.003600	0.015300	0.235294
1	40	0.001400	0.012400	0.112903
2	50	0.001500	0.011600	0.129310
3	60	0.000500	0.012700	0.039370
4	70	0.002500	0.013600	0.183824
5	80	0.001700	0.011000	0.154545
6	90	0.000600	0.012400	0.048387
7	100	0.000700	0.014300	0.048951
8	110	0.003200	0.012500	0.256000
9	120	0.000700	0.011100	0.063063
10	130	0.007300	0.014300	0.510490
11	140	0.008900	0.017300	0.514451
12	150	0.005000	0.019000	0.263158
13	160	0.004000	0.023000	0.173913
14	170	0.003000	0.028000	0.107143
15	180	0.002000	0.035000	0.057143
16	190	0.001500	0.045000	0.033333
17	200	0.001000	0.060000	0.016667

Приложимост на разработената работа

Приложението може директно да се въведе като допълнителен инструмент в различни области. Например в индустриални условия може да подпомага процеси по автоматизация чрез откриване на дефекти, преброяване на готова продукция или следене на производствени линии.

Заклучение

- Като програмен език и среда за разработка е избран Python.
- За реализирането на базата данни е избран MongoDB
- Създадена е архитектура на софтуерната система базирана на MVC шаблон
- Реализиран е модул за качване и обработка на изображения
- Реализиран е модул за разпознаване на конкретен обект
- Реализиран е модул за съхраняване на разпознатите обекти
- Реализиран е модул за търсене и визуализация
- Реализиран е модул за потребителско управление
- Експериментално е изследван случай на overfitting на предварително избран dataset

Използвани софтуерни инструменти и библиотеки

- **Python и Anaconda** - version 3.1.7
- **Streamlit** - streamlit==1.43.0, streamlit-cookies-manager==0.2.0, streamlit-cropper==0.2.2, streamlit-js-eval==0.1.7, streamlit-paste-button==0.1.2, st-img-pastebutton==0.0.6
- **Pandas** - pandas==2.2.3
- **PIL (Python Imaging Library)** - pillow==11.1.0
- **OpenCV** - opencv-python==4.11.0.86
- **Numpy** - numpy==2.0.2
- **Bcrypt** - bcrypt==4.3.0
- **Tensorflow** - tensorflow==2.18.0, tensorboard==2.18.0, ml-dtypes==0.4.1
- **Keras** - keras==3.9.0
- **Ultralytics** - ultralytics==8.3.85, ultralytics-thop==2.0.14
- **MongoDB compass** – version 1.46.11
- **MongoDB/Pymongo** - pymongo==4.13.2
- **Requests** - requests==2.32.3
- **Timezones** - tzlocal==5.3.1, pytz==2025.1

Благодаря ви за вниманието!