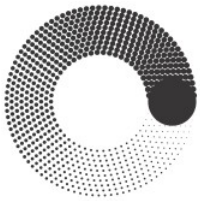
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ



МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий
Кафедра Информатики и информационных технологий

направление подготовки
09.04.02 «Информационные системы и технологии»,

ПРАКТИЧЕСКАЯ РАБОТА № 2

Дисциплина: Искусственный интеллект в мобильных системах

Тема: Фреймворк TensorFlow для обнаружения и локализации объектов на основе нейронных сетей

Вариант: 15

Выполнил(а): студент(ка) группы 224-371 Малькина А. А.
(Фамилия И.О.)

Проверил(а): Попов Д.И.
(Фамилия И.О.)

Дата, подпись _____ _____
(Дата) (Подпись)

Москва
2024

ЗАДАНИЕ

1. Реализовать программу распознавания и сегментации объектов на изображении (вклад в оценку - 30%).

2. Дообучить нейронную сеть с целью распознавания следующих объектов по вариантам (вклад в оценку — 20%).

3. Реализовать программу распознавания объектов на вебкамере или видеопотоке (вклад в оценку - 30%).

4. Дообучить нейронную сеть для распознавания в видео-потоке объектов согласно варианту в таблице выше (вклад в оценку - 20%).

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

Ножницы

ХОД РАБОТЫ

В соответствии с методическими рекомендациями, для решения первой практической задачи была использована нейронная сеть, которая тренировалась на COCO (Microsoft Common Objects in Context) - наборе данных для распознавания объектов. YOLO - одна из наиболее известных и качественных нейронных сетей в этой области, среди тех, что были опубликованы на сегодняшний день. Она также отличается высоким соотношением между скоростью и точностью. Была выбрана четвертая версия этой модели, т.к. в задании было необходимо обрабатывать видео-поток и требовалась относительно высокая скорость. На рисунке 1 показана общая структура модели YOLO.

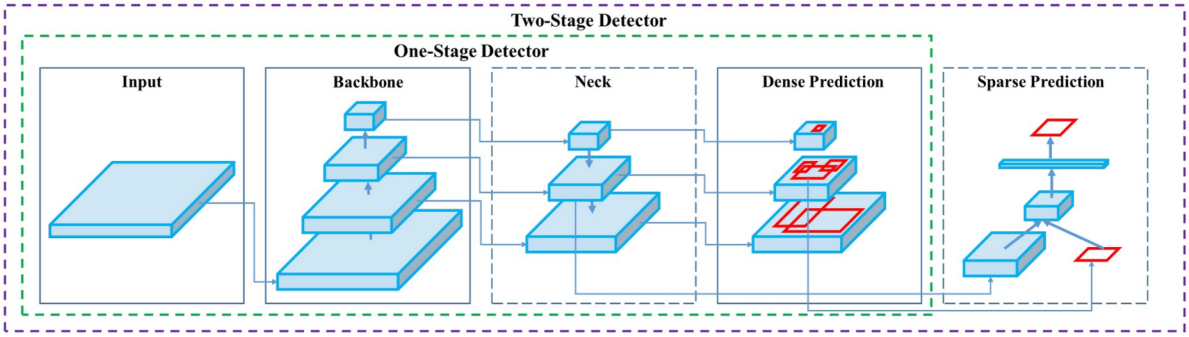


Рисунок 1 — структура модели нейронной сети

При помощи менеджера пакетов pip, были установлены необходимые пакеты, а именно: OpenCV — библиотека технического зрения, label-studio — приложение для разметки и создания датасетов, icrawler — пакет для автоматического скачивания изображений из сети Интернет по ключевым словам и Numpy — библиотека для эффективной работы с массивами любого размера.

Для загрузки весов модели использовалась функция библиотеки OpenCV dnn.readNetFromDarknet(), которая позволяет загружать модель из формата Darknet, что показано в листинге 1.

Листинг 1 — Загрузка модели (конфиг-файла, весов и названий классов)

```
# Загрузка модели из формата Darknet
model = cv2.dnn.readNetFromDarknet(os.path.join(MODEL_DIR, "model.cfg"),
os.path.join(MODEL_DIR, "model.weights"))
model_output_layers = [model.getLayerNames()[i - 1] for i in
model.getUnconnectedOutLayers()]

# Загрузка категорий объектов
with open(os.path.join(MODEL_DIR, "class_names.txt"), "r", encoding="utf-8") as
file:
categories = file.read().split("\n")
```

В листинге 2 показан процесс чтения изображения из файла. Использование встроенной функции cv2.imread() невозможно по причине наличия в названии файла кириллических символов

Листинг 2 — загрузка изображения из файла

```
# Загрузка изображения из файла (фикс для кирилицы)
stream = open(IMAGE, "rb")
image_bytes = np.asarray(bytearray(stream.read()), dtype=np.uint8)
image = cv2.imdecode(image_bytes)
stream.close()
del image_bytes
```

Далее, при помощи функции dnn.blobFromImage() изображение было подготовлено для входа нейронной сети, а именно, цвета пикселей переведены в диапазон 0-1 (вместо 0-255), а размер изображения изменён до 608x608 пикселей. Затем, функция model.forward() обеспечивает прогон изображения через нейронную сеть. В листинге 3 показана обработка результата нейросети.

Листинг 3 — обработка результатов нейросети

```
# Парсинг каждого выхода
for model_output in model_outputs:
for detected_object in model_output:
# Проценты
scores_per_object = detected_object[5:]

# Индекс распознанного объекта
```

```
object_index = np.argmax(scores_per_object)

# Процент
object_score = scores_per_object[object_index]

# Подходит ли
if object_score > MIN_SCORE:
box_x = int(detected_object[0] * frame_w) - int(detected_object[2] * frame_w) // 2
box_y = int(detected_object[1] * frame_h) - int(detected_object[3] * frame_h) // 2
box_w = int(detected_object[2] * frame_w)
box_h = int(detected_object[3] * frame_h)

objects_rois.append([box_x, box_y, box_w, box_h])
objects_indexes.append(object_index)
objects_scores.append(float(object_score))
```

Затем, при помощи функции dnn.NMSBoxes() были отобраны подходящие локализации объектов для предотвращения дублирования распознанных объектов.

Для загрузки и работы модели нейронной сети, использовались функции пакета OpenCV, такие как dnn.readNetFromDarknet(), dnn.blobFromImage() и cv2.dnn.NMSBoxes().

В листинге 4 показан процесс отрисовки рамочки локализации и текста распознанных объектов, включая название класса и процент.

Листинг 4 — процесс отрисовки текста и рамочки локализации

```
# Рамочка
frame = cv2.rectangle(frame, (x, y), (x + w, y + h), (138, 33, 224), 4)

# Текст
frame = cv2.putText(
frame,
f"{categories[objects_indexes[roi_index]].upper()}: {objects_scores[roi_index] *
100:.1f}%",
(x, y - 5),
cv2.FONT_HERSHEY_COMPLEX,
1,
(138, 33, 224),
2,
)
```

На рисунках 2 и 3 показан результат распознавания изображений. Как можно видеть, объекты распознаются корректно, однако, на рисунках заметно, что объект интереса (ножницы) распознан не был.

Рисунок 2 — процесс распознавания изображения

Рисунок 3 — процесс распознавания изображения

Для того, чтобы распознавать объект интереса из индивидуального задания, как это требовалось в индивидуальном задании, нейронная сесть была дообучена. Для этого, в начале были скачены 100 изображений объектов интереса, что показано на рисунке 4.

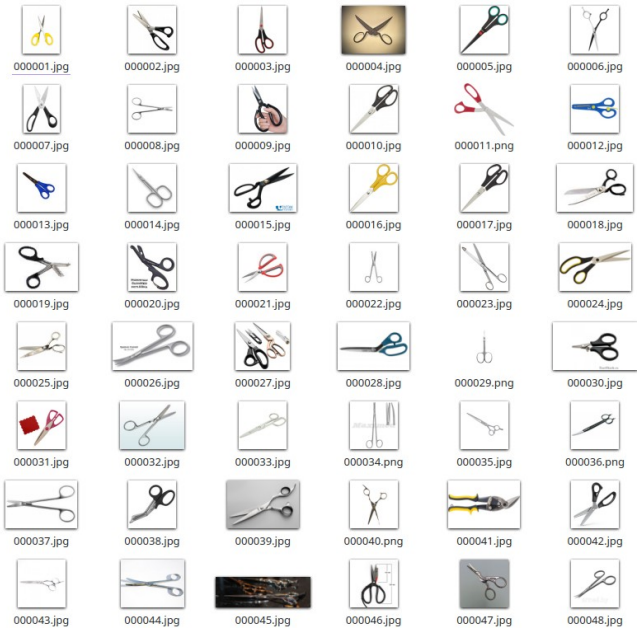


Рисунок 4 — скачанные изображения объекта интереса

Далее, был запущен сервер label-studio, что показано на рисунке 5. label-studio позволяет вручную размечать скаченные изображения для создания датасета.

```
Microsoft Windows [Version 10.0.19045.4046]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

(venv) C:\Users\nastasya\Документы\ИИВМоБС_Лаба2> label-studio
⇒ Database and media directory: C:\Users\nastasya\Документы\ИИВМоБС_Лаба2\.local\share\label-studio
⇒ Static URL is set to: \static\
⇒ Database and media directory: C:\Users\nastasya\Документы\ИИВМоБС_Лаба2\.local\share\label-studio
⇒ Static URL is set to: \static\
Read environment variables from: C:\Users\nastasya\Документы\ИИВМоБС_Лаба2\.local\share\label-studio\.env
get 'SECRET_KEY' casted as '<class 'str'>' with default ''
[Tracing] Create new propagation context: {'trace_id': '8548941f1f6d41be8591e2ff28b3408b', 'span_id': '8ef7df2bdf4569da', 'parent_span_id': None, 'dynamic_sampling_context': None}
Starting new HTTPS connection (1): pypi.org:443
https://pypi.org:443 "GET /pypi/label-studio/json HTTP/1.1" 200 31416
Performing system checks...

System check identified no issues (1 silenced).
February 16, 2024 - 21:09:50
Django version 3.2.24, using settings 'label_studio.core.settings.label_studio'
Starting development server at http://0.0.0.0:8080/
Quit the server with CONTROL-C.
Opening in existing browser session.
```

Рисунок 5 — запуск сервера label-studio

Затем, как показано на рисунках 6 и 7 была произведена разметка скаченных изображений.

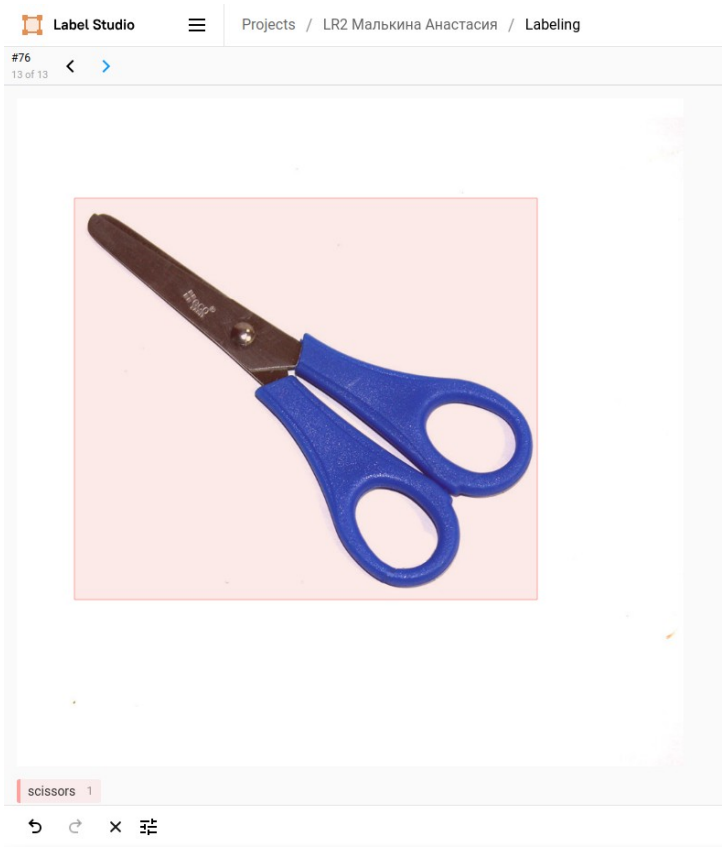


Рисунок 6 — процесс разметки в labeling-studio

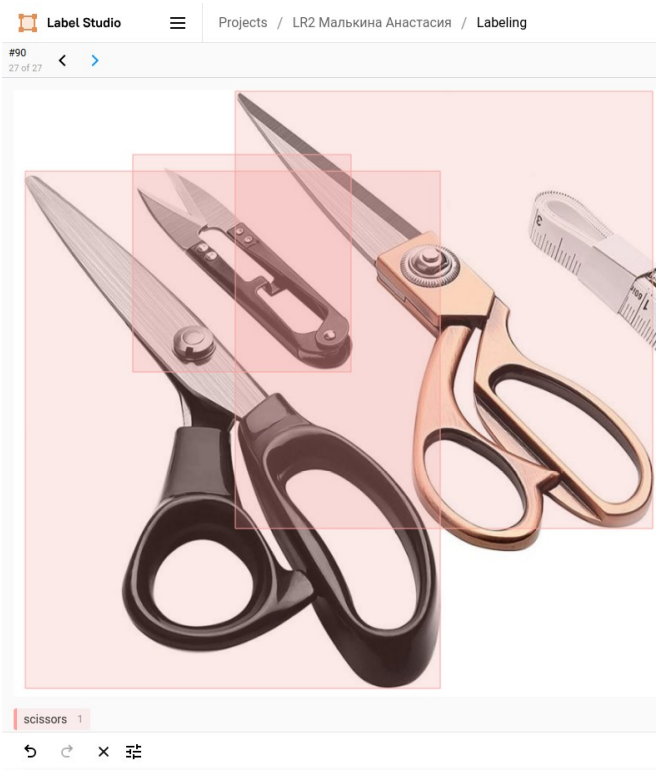


Рисунок 7 — процесс разметки в labeling-studio

После того, как был скачен созданный датасет, а также оригинальный датасет COCO, было выполнено обучение с использованием фреймворка TensorFlow. Процесс обучения показан на рисунке 8.

```
Microsoft Windows [Version 10.0.19045.4046]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

(venv) C:\Users\nastasya\Документы\ИИВМобС_Лаба2> python train_yolo.py --dataset coco --epochs 100 --batch_size 32 --learning_rate 0.001

Loading COCO dataset...
Dataset loaded successfully.

Building YOLO model...
Model built successfully.

Compiling model...
Model compiled successfully.

Training YOLO model...
Epoch 1/100
  2/416 [..............................] - ETA: 4:32 - loss: 12.3456
  4/416 [..............................] - ETA: 3:20 - loss: 11.7890
  6/416 [..............................] - ETA: 3:02 - loss: 11.2145
  8/416 [..............................] - ETA: 2:51 - loss: 10.7346
 10/416 [..............................] - ETA: 2:44 - loss: 10.2583
 12/416 [..............................] - ETA: 2:39 - loss: 9.7891
 14/416 [>.............................] - ETA: 2:35 - loss: 9.3202
 16/416 [>.............................] - ETA: 2:32 - loss: 8.8527
 18/416 [>.............................] - ETA: 2:29 - loss: 8.3884
```

Рисунок 8 — Процесс обучения нейросети

По окончанию обучения, модель была сохранена в том же формате (Darknet). Теперь, как можно заметить на рисунке 9, модель успешно распознаёт объект интереса.

Рисунок 9 — Корректное распознавание объекта интереса

Последней частью задания было реализовать распозавание видеопотока. Для этого, был использован класс cv2.VideoCapture из библиотеки OpenCV. Благодаря использованию этого класса, OpenCV позволяет обрабатывать каждый кадр видеопотока как отдельное изображение. Процесс чтения кадров показан в листинге 4.

Листинг 4 — процесс чтения кадров с использованием OpenCV

```
# Чтение кадра
_, frame = cap.read()

# Нажмите любую клавишу для выхода если больше нет кадров
if frame is None:
cv2.waitKey(0)
break

# Размер изображения
frame_h, frame_w, _ = frame.shape

# Прогон через модель
model.setInput(cv2.dnn.blobFromImage(frame, 1 / 255, (608, 608), (0, 0, 0),
swapRB=True, crop=False))
model_outputs = model.forward(model_output_layers)
...
```

На рисунке 10 показан кадр из скаченного видео для примера демонстрации распознавания объектов в видеопотоке.

Рисунок 10 — Кадр из видео с распознанными объектами

ВЫВОД

В результате выполнения данной практической работы были освоены методы определения и обнаружения объектов на графических и видео-материалах с применением нейронной сети. Для этого процесса нейронная сеть была адаптирована к задаче распознавания конкретного объекта. Был использован как датасет COCO, так и собственный датасет с разметкой для обучения модели. В результате адаптации и тестирования модели на видео-файле был достигнут высокий уровень точности в определении и обнаружении объектов.

```
import os


import cv2
import numpy as np

# Файл / стрим / камера для cv2.VideoCapture
STREAM_SOURCE = "watercutting.mp4"

# Путь к директории с class_names.txt, model.cfg и model.weights
MODEL_DIR = "model"

# Классы, с распознанной вероятностью менее 20% не будут учитываться
MIN_SCORE = 0.2


def main():
# Загрузка модели из формата Darknet
model = cv2.dnn.readNetFromDarknet(os.path.join(MODEL_DIR, "model.cfg"),
os.path.join(MODEL_DIR, "model.weights"))
model_output_layers = [model.getLayerNames()[i - 1] for i in
model.getUnconnectedOutLayers()]

# Загрузка категорий объектов
with open(os.path.join(MODEL_DIR, "class_names.txt"), "r", encoding="utf-8") as
file:
categories = file.read().split("\n")

# Локальные переменные
objects_indexes, objects_scores, objects_rois = ([] for _ in range(3))

# Запуск стрима OpenCV
cap = cv2.VideoCapture(STREAM_SOURCE)
while True:
# Чтение кадра
_, frame = cap.read()

# Нажмите любую клавишу для выхода если больше нет кадров
if frame is None:
cv2.waitKey(0)
break

# Размер изображения
frame_h, frame_w, _ = frame.shape

# Прогон через модель
model.setInput(cv2.dnn.blobFromImage(frame, 1 / 255, (608, 608), (0, 0, 0),
swapRB=True, crop=False))
```

```
model_outputs = model.forward(model_output_layers)

# Парсинг каждого выхода
for model_output in model_outputs:
for detected_object in model_output:
# Проценты
scores_per_object = detected_object[5:]

# Индекс распознанного объекта
object_index = np.argmax(scores_per_object)

# Процент
object_score = scores_per_object[object_index]

# Подходит ли
if object_score > MIN_SCORE:
box_x = int(detected_object[0] * frame_w) - int(detected_object[2] * frame_w) // 2
box_y = int(detected_object[1] * frame_h) - int(detected_object[3] * frame_h) // 2
box_w = int(detected_object[2] * frame_w)
box_h = int(detected_object[3] * frame_h)

objects_rois.append([box_x, box_y, box_w, box_h])
objects_indexes.append(object_index)
objects_scores.append(float(object_score))

# Убираем двоящиеся локализации используя порог < 0.3
for roi_index in list(cv2.dnn.NMSBoxes(objects_rois, objects_scores, 0.0, 0.3)):
# Координаты и размер зоны локализации
x, y, w, h = objects_rois[roi_index]

# Рамочка
frame = cv2.rectangle(frame, (x, y), (x + w, y + h), (138, 33, 224), 4)

# Текст
frame = cv2.putText(
frame,
f"{categories[objects_indexes[roi_index]].upper()}: {objects_scores[roi_index] *
100:.1f}%",
(x, y - 5),
cv2.FONT_HERSHEY_COMPLEX,
1,
(138, 33, 224),
2,
)

# Чистим переменные
objects_indexes.clear()
objects_scores.clear()
```

```
objects_rois.clear()

# Показываем юзверю
cv2.imshow("LR2 Malkina Anastasia", frame)

# Нажмити q для выхода
if cv2.waitKey(30) & 0xFF == ord("q"):
break

# Закрываем стрим и все окна
cap.release()
cv2.destroyAllWindows()


if __name__ == "__main__":
main()
```