Алгоритмы детекции объектов

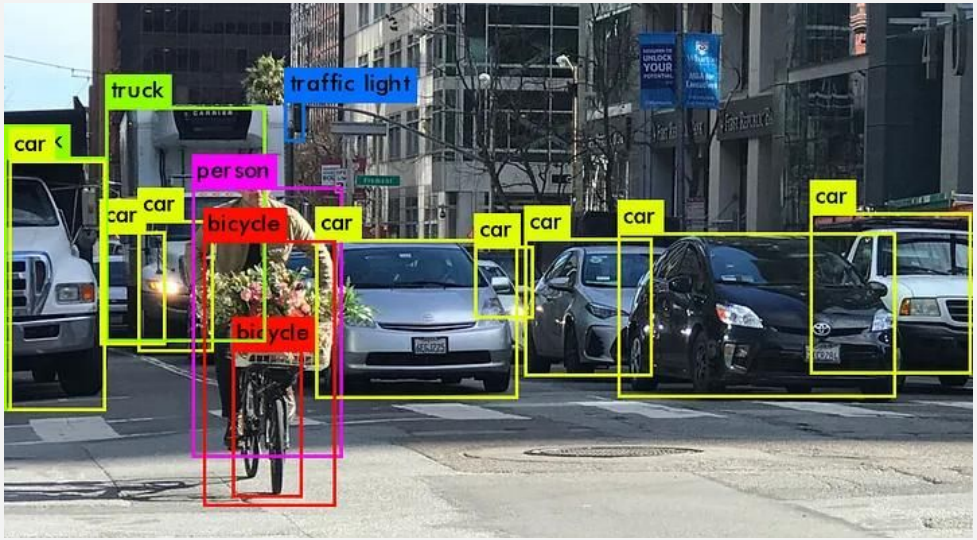
Алгоритмы детекции объектов

Детекция объектов — это процесс определения и локализации объектов на изображении или в видео.

Основная задача: **указать, где находятся объекты и к какому классу они принадлежат** (например, человек, автомобиль, животное).

Используется в различных областях:

- Видеонаблюдение
- Беспилотные автомобили
- Медицина
- Робототехника



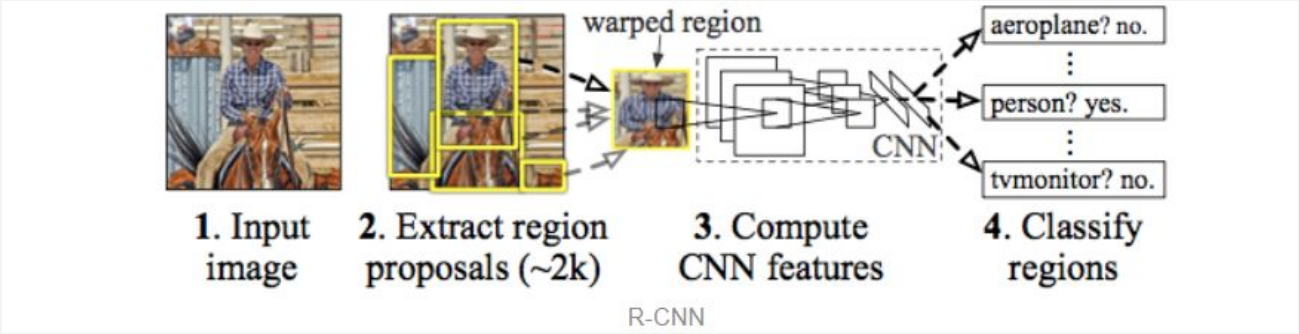
Основные этапы детекции объектов

1. ***Предобработка данных***
2. ***Извлечение признаков (feature extraction)***
3. ***Генерация кандидатов (Region Proposals)***
4. ***Классификация и локализация***
5. ***Фильтрация результатов (Post-processing)***
6. ***Вывод результата***

Общая архитектура систем детекции объектов

Вход

- Изображение передаётся на вход модели
- Размер изображения может быть адаптирован



R-CNN

Предобработка изображения

Подготовить изображение к подаче в модель для повышения точности и стабильности работы.

🛠️ **Основные шаги:**

1. Изменение размера (Resizing)

2. Нормализация пикселей (Normalization)

3. Аугментация (Data Augmentation)

4. Преобразование формата

1. Изменение размера (Resizing)

- Большинство моделей требуют фиксированный размер входа (например, 224×224, 300×300, 416×416).
- Обеспечивает согласованность батчей при обучении.
- Уменьшает вычислительные затраты.

🧠 **Как это делается:**

- Используются алгоритмы интерполяции (билинейная, бикубическая и др.).
- Сохраняется пропорция или добавляются отступы (padding), если это критично.

2. Нормализация пикселей (Normalization)

Пиксели исходного изображения имеют значения от 0 до 255

Мы преобразуем их в диапазон от **0 до 1** или **от -1 до 1**

Формула: `x_normalized = x / 255` или `(x - среднее) / стандартное_откл`

💡 **Зачем это нужно:**

- Упрощает обработку данных нейросетями
- Снижает влияние масштабов и ускоряет сходимость градиента
- Предотвращает числовую нестабильность

3. Аугментация (Data Augmentation)

Искусственно увеличить объём тренировочных данных и улучшить обобщающую способность модели.

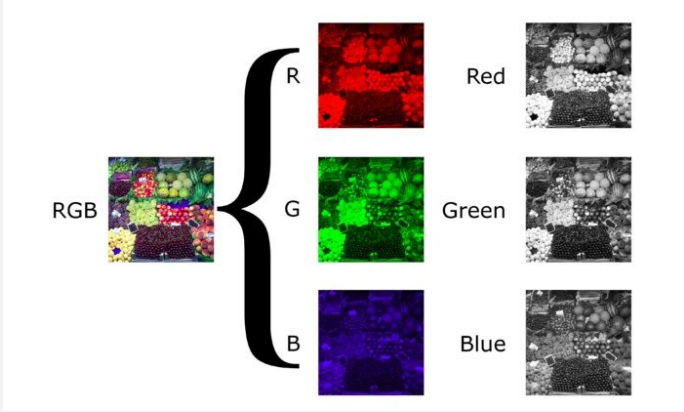
🎯 **Зачем использовать:**

- Повышает устойчивость к переобучению
- Позволяет модели лучше распознавать объекты в различных положениях
- Особенно важно при небольших датасетах

3. Аугментация. Основные техники:

- **Повороты (Rotation) -** изображения поворачиваются на случайные углы.
- **Отражения (Flip) -** горизонтальное или вертикальное отражение.
- **Изменение масштаба (Zoom) -** увеличение или уменьшение без потери содержимого.
- **Сдвиги (Shifts) -** сдвиг по ширине и высоте.
- **Шумы, размытие, яркость -** добавление случайных искажений.

4. Преобразование формата



- **Переименование и переиндексация**

Имена файлов и классов унифицируются (например, `image_001.jpg`)

- **Конвертация аннотаций -** из JSON, XML или CSV в формат, поддерживаемый фреймворком (YOLO, COCO, Pascal VOC etc.)
- **Изменение структуры данных -** разделение по папкам: `images/train`, `images/val`, `labels/train`, `labels/val`
- **Изменение каналов изображения -** RGB → Grayscale или наоборот, если модель требует

Выделение признаков (Feature Extraction)

Преобразовать изображение в числовые представления, содержащие важную информацию о формах, цветах, текстурах и границах объектов.

Как это работает:

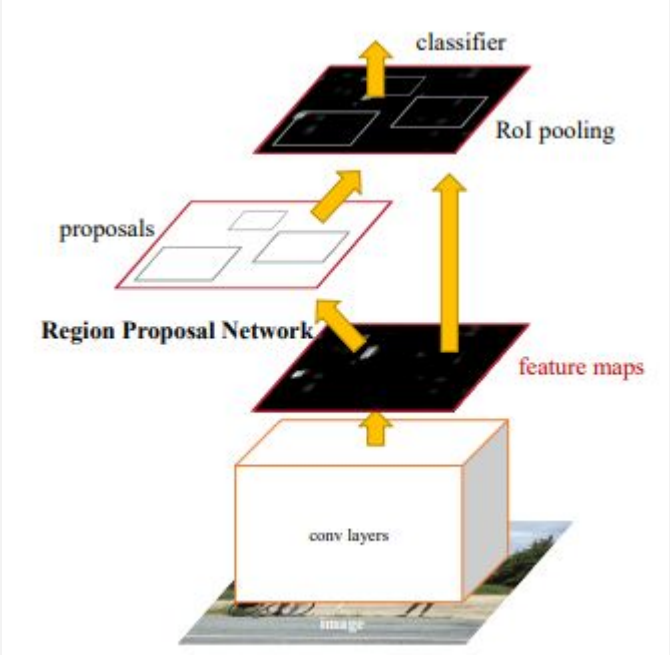
- Низкие уровни CNN: простые признаки (линии, цвета).
- Средние уровни: формы, структуры.
- Высокие уровни: сложные паттерны, части объектов.

Выделение признаков. Ключевые методы

- **Ручные признаки (до эпохи DL):** SIFT, HOG, SURF — выделяют края, углы и структуры вручную.
- **Автоматические признаки (с DL):** сверточные нейронные сети (CNN) автоматически извлекают признаки с разных уровней абстракции.

Генерация кандидатов (Region Proposals)

Определить потенциальные области изображения, где могут находиться объекты — до точной классификации и уточнения границ.

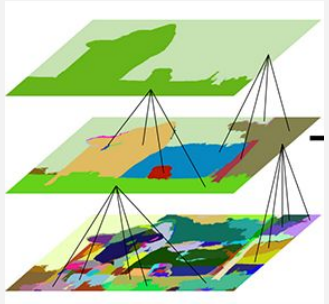


Генерация кандидатов. Подходы:

Селективный поиск (Selective Search): классический метод - объединение похожих регионов по цвету, текстуре, размеру

Edge Boxes: оценивает вероятность наличия объекта по количеству и расположению краев в области

RPN (Region Proposal Network): глубокая нейросеть, встроенная в архитектуру Faster R-CNN, генерирует регионы кандидаты на основе признаков.



Классификация и локализация

Определить, что именно находится в предложенной области (класс объекта), и уточнить координаты ограничивающей рамки (bounding box).

- **Классификация:** для каждой области-кандидата предсказывается один из известных классов (например: собака, машина, человек).
- **Локализация:** определяются координаты рамки, которые наиболее точно охватывают объект внутри области.

Фильтрация результатов (Post-processing)

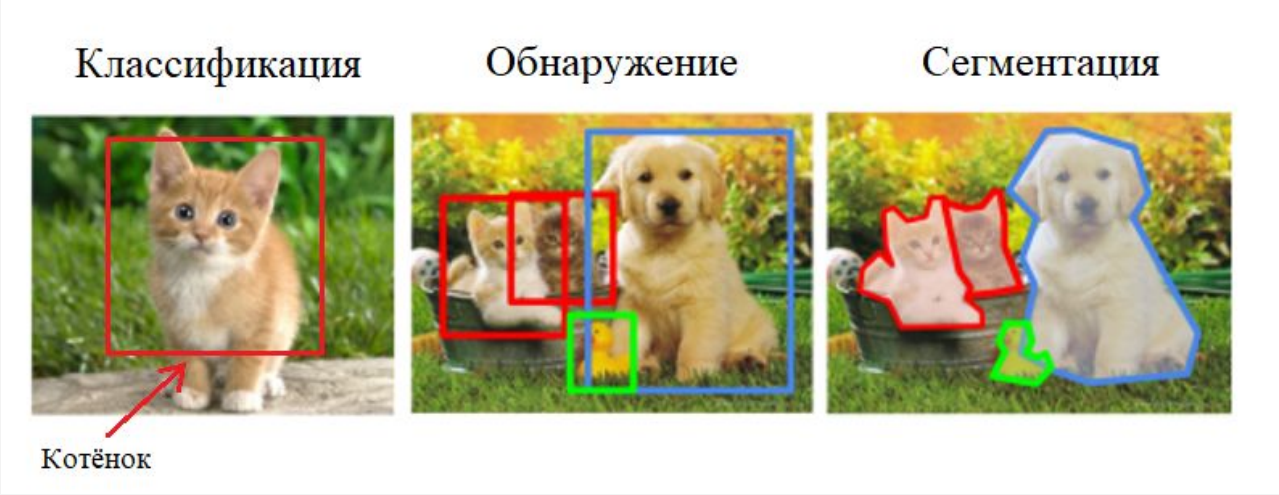
Удалить лишние или ошибочные предсказания и оставить только наиболее точные.

Основные методы:

- **Non-Maximum Suppression (NMS):** удаляет перекрывающиеся рамки, оставляя только рамку с наибольшей вероятностью
- **Порог вероятности (Thresholding):** удаляет объекты, вероятность которых ниже заданного значения (например, < 0.5)
- **Калибровка рамок (Bounding Box Refinement):** уточняет координаты рамки для повышения точности

Вывод результата

Показать пользователю финальные предсказания модели в понятной форме.



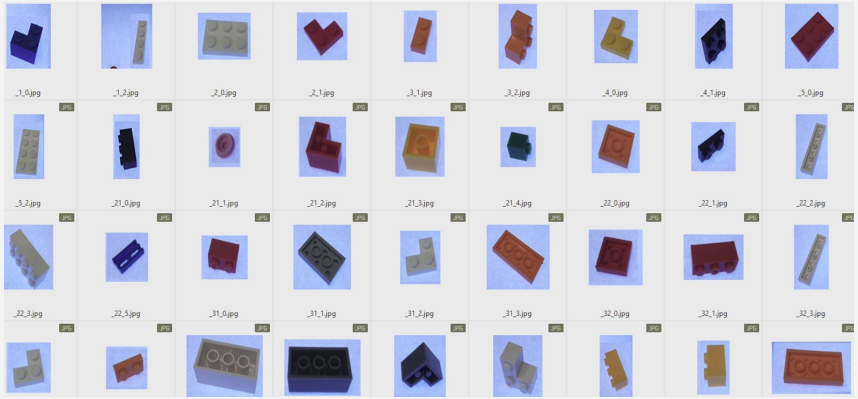
Вывод результата. Основные методы:

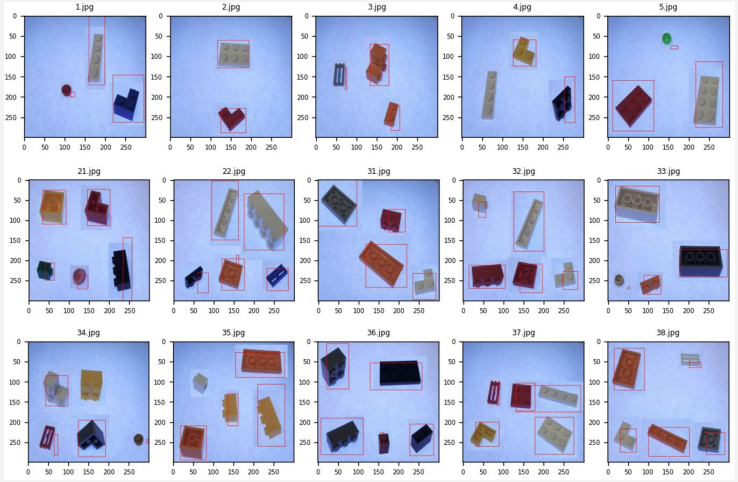
- **Non-Maximum Suppression (NMS):** удаляет перекрывающиеся рамки, оставляя только рамку с наибольшей вероятностью
- **Порог вероятности (Thresholding):** удаляет объекты, вероятность которых ниже заданного значения (например, < 0.5)
- **Калибровка рамок (Bounding Box Refinement):** уточняет координаты рамки для повышения точности

Методы детекции объектов

Методы детекции объектов

1. **Классические (традиционные) методы**
2. **Глубокое обучение (нейросетевые методы)**
3. **Single-stage vs Two-stage**



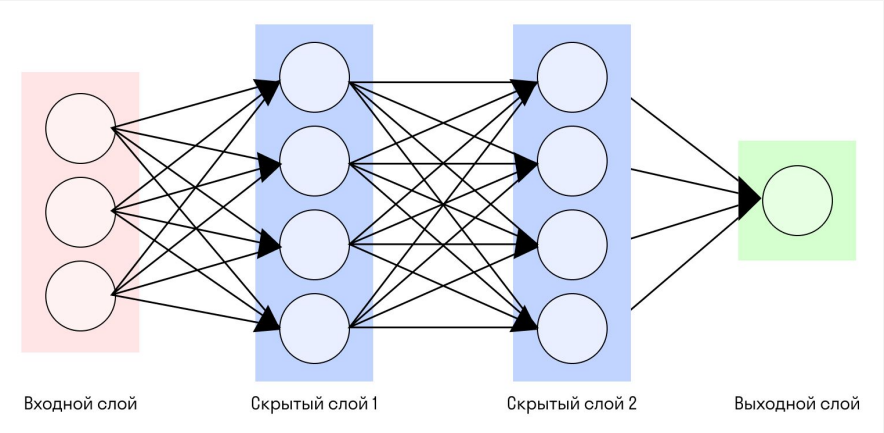


Классические (традиционные) методы

- ➔ Основаны на ручном выделении признаков
- ➔ Быстро работают на слабом железе
- ➔ Не используют нейросети. Используются алгоритмы, как:
  - ***Haar Cascades*** – используется в OpenCV, эффективен для распознавания лиц.
  - ***HOG + SVM*** – гистограмма направленных градиентов + метод опорных векторов.
  - ***Edge Boxes*** – генерация регионов на основе границ.

Глубокое обучение (нейросетевые методы)

- Используют сверточные нейронные сети (CNN)
- Обучаются на больших датасетах
- Автоматически извлекают признаки
- Обеспечивают высокую точность при высокой вычислительной нагрузке



Глубокое обучение. Примеры архитектур

- **R-CNN, Fast R-CNN, Faster R-CNN** – двухэтапные методы.
- **YOLO (You Only Look Once)** – одноэтапный метод, очень быстрый.
- **SSD (Single Shot Detector)** – баланс между скоростью и точностью.
- **RetinaNet** – устойчив к дисбалансу классов.

Single-stage vs Two-stage

1. Two-Stage (двухэтапные) методы

- Сначала **генерируются регионы-кандидаты** (Region Proposals)
- Затем проводится **классификация и уточнение координат**

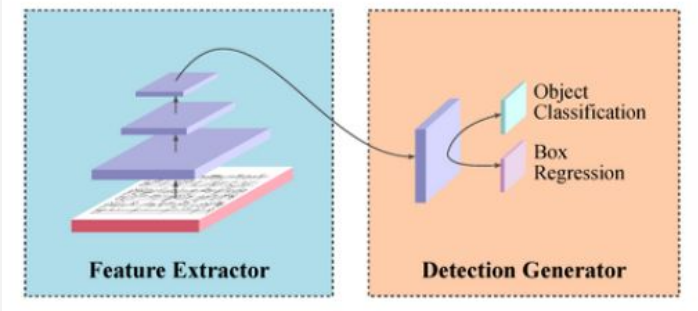
2. One-Stage (одноэтапные) методы

- Модель сразу предсказывает **локации объектов и их классы**
- Быстрее, но может уступать в точности

Single-stage vs Two-stage

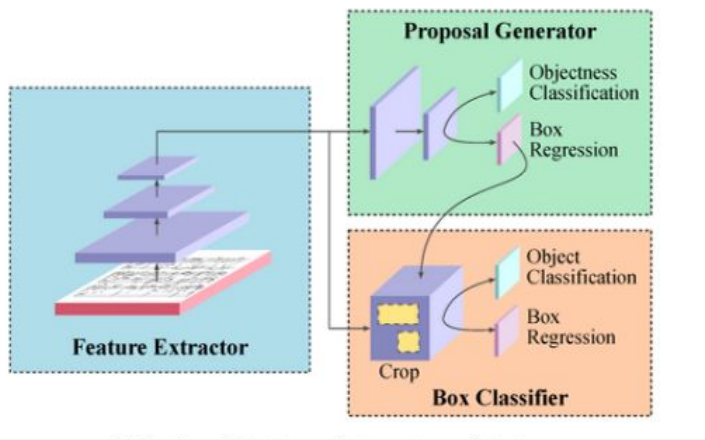
One-Stage методы:

- YOLO (You Only Look Once)
- SSD (Single Shot MultiBox Detector)



Two-Stage методы:

- R-CNN
- Fast R-CNN
- Faster R-CNN



Характеристика	Single-stage	Two-stage
Этапы обработки	Один этап (детекция и классификация)	Два этапа (предсказание + классификация)
Скорость	Очень высокая (реальное время)	Ниже, особенно на CPU
Точность	Ниже, чем у two-stage	Выше, особенно при мелких объектах
Примеры	YOLO, SSD	R-CNN, Faster R-CNN
Сложность модели	Проще	Сложнее
Применение	Мобильные и встраиваемые устройства	Системы с высокой точностью

Примеры

YOLOv5 (You Only Look Once v5)

- Делит изображение на сетку, предсказывает классы и координаты объектов за один проход.
- Работает в реальном времени.

Установка

git clone https://github.com/ultralytics/yolov5
cd yolov5
pip install -r requirements.txt

Запуск детекции на изображении: *python detect.py --weights yolov5s.pt --img 640 --conf 0.25 --source data/images/zidane.jpg*

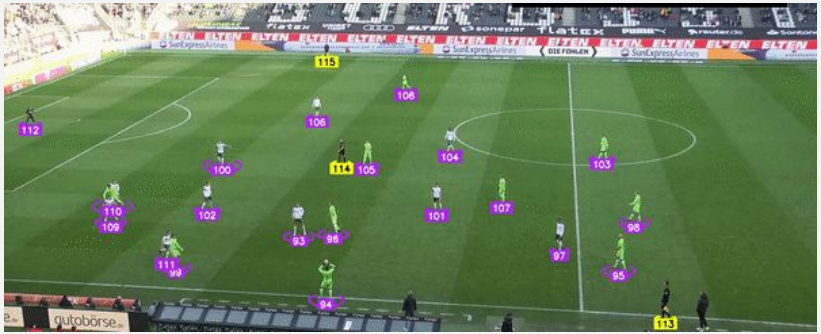
Пример кода Python

```
import torch
from PIL import Image
from matplotlib import pyplot as plt

   model     =     torch.hub.load('ultralytics/yolov5',
'yolov5s', pretrained=True)

img = Image.open('data/images/zidane.jpg')

results = model(img)

results.print()
results.show()
```

Выходной результат

Модель выведет список объектов, например:

```
person  85%

sports ball  76%
```



Haar Cascades (OpenCV)

Обнаруживает лица, глаза, силуэты с помощью обученных каскадов признаков.

```
import cv2
face_cascade = cv2.CascadeClassifier
('haarcascade_frontalface_default.xml')
faces = face_cascade.detectMultiScale(gray_img,
scaleFactor=1.3, minNeighbors=5)
```

Где применяют: базовая видеоаналитика, распознавание лиц в веб-камере.

HOG + SVM (Histogram of Oriented Gradients)

Извлекает ориентированные градиенты и классифицирует через SVM.

```
from skimage.feature import hog
features = hog(image, orientations=9, pixels_per_cell=(8, 8), cells_per_block=(2, 2))
```

Где применяют: обнаружение пешеходов на изображениях с камер наблюдения.

YOLOv5 (You Only Look Once)

Одновременно находит и классифицирует объекты на изображении.

```
# Из командной строки
python detect.py --weights yolov5s.pt --img 640 --source image.jpg
```

Где применяют: видеонаблюдение в реальном времени, автопилоты, дроны.

SSD (Single Shot MultiBox Detector)

Быстрый и точный детектор, использующий предопределённые якоря.

```
import tensorflow as tf
model = tf.saved_model.load
('ssd_mobilenet_v2_fpnlite_320x320_coco17')
detections = model(input_tensor)
```

Где применяют: мобильные устройства, AR/VR камеры.

Faster R-CNN

Сначала предлагает регионы (RPN), потом классифицирует их.

```
import torchvision
model = torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)
model.eval()
result = model([image_tensor])
```

Где применяют: медицинская визуализация, спутниковые снимки.

DETR (Detection Transformer)

Использует трансформеры для глобального внимания, без NMS.

```
   model = torch.hub.load('facebookresearch/detr',
'detr_resnet50', pretrained=True)
   outputs = model(images)
```

Где применяют: сложные сцены, беспилотники, роботы-манипуляторы.

Grad-CAM (визуализация карты активации)

Показывает, на какие области модели "смотрит" при распознавании.

```
from pytorch_grad_cam import GradCAM
cam = GradCAM(model=model,
target_layers=[model.layer4])
heatmap = cam(input_tensor, targets=[class_index])
```

Где применяют: медицина, аудит ИИ-моделей, образовательные проекты.

Tiny YOLO (например, YOLOv4-Tiny)

Упрощённая версия YOLO для слабых устройств.

```
   # Использование с OpenCV DNN
   net = cv2.dnn.readNet("yolov4-tiny.weights",
"yolov4-tiny.cfg")
   blob = cv2.dnn.blobFromImage(img, 1/255, (416,
416), swapRB=True)
   net.setInput(blob)

   outputs =
net.forward(net.getUnconnectedOutLayersNames())
```

Где применяют: Raspberry Pi, видеокамеры, системы умного дома

MobileNet-SSD

Легкий детектор на базе MobileNet + SSD, хорош для Android/Edge.

```
   net = cv2.dnn.readNetFromCaffe("deploy.prototxt",
"mobilenet_iter_73000.caffemodel")
   blob = cv2.dnn.blobFromImage(image, 0.007843,
(300, 300), 127.5)
   net.setInput(blob)
   detections = net.forward()
```

Где применяют: мобильные приложения, AR-очки, носимая электроника.