
An Unexpected Journey to Computer Vision and Object Detection, or There and Back Again

Обо мне

- 8+ лет опыта в разработке мобильных приложений под Android.
- 3+ лет опыта в работе с AOSP.
- 3+ лет в разработке на C++.
- 2+ лет в разработке бэкенд на Java.
- Опыт в разработке под iOS с использованием Swift.
- Опыт в мультиплатформенной разработке с использованием Flutter, React Native и других технологий и KMP.

Обо мне

Agenda

О Проекте

Deep learning фреймворки

Компьютерное зрение and Convolutional Neural Network(CNN)

Самый современный и быстрый подход к компьютерному зрению

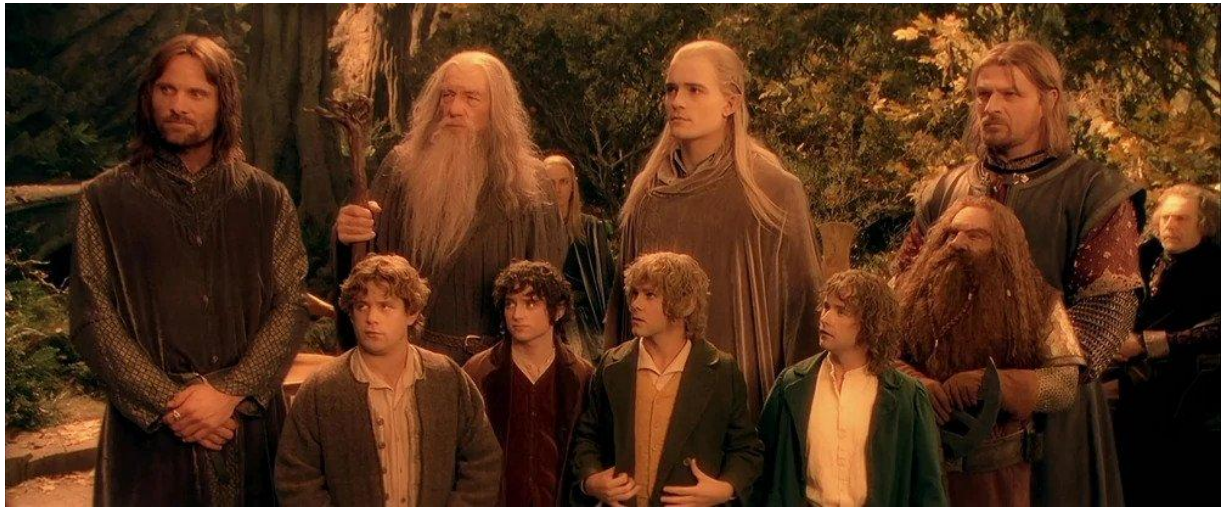
Адаптация моделей на мобильных устройствах


About the project

Dashcams

Приложение

- **Команда SDK 📱:**
  - **2 разработчика под Android**
  - **1 разработчик под iOS**
  - **2 разработчика под iOS и Android**



Основные обязанности команды 🔗:

- Установление соединения с видеорегистратором с использованием Bluetooth, BLE и Wi-Fi Direct.
- Управление видеорегистратором, передача и прием событий от приложения к камере и от камеры к приложению.
- Обработка и оптимизация данных с видеорегистратора.
- Обеспечение синхронизации данных и бесшовной репликации с сервером.

My team to dive into ML:

Team Lead

iOS developers

Android developer

Documentation guy

Задача - Разработать модель для обнаружения объектов

- **Выбор модели**
  - MVP.
  - Создание или tuning.
- **Возможности детекции объектов:**
  - Дорожные знаки.
  - Преграды.
  - Аварийные ситуации.
  - Парковки.
- **Пользовательский опыт**:
  - Realtime карта.
- **Управление данными:**
  - Отправка метаданных на сервер.

Есть ли у нас документация?

Есть ли разработчик, который работал над предыдущим MVP?

Есть ли разработчик моделей детекции объектов?

Let the unexpected journey begin!

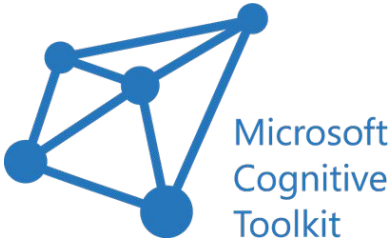

Deep learning
frameworks

- **Advantages:**
  - Динамичность: Легкая замена и адаптация моделей.
  - Дружелюбие к пользователю: Более простая разработка кода.
  - Поддержка исследований: Популярность в академических проектах.

- **Disadvantages:**
  - Внедрение: Труднее использовать в реальных приложениях.

- **Advantages:**
  - Многопроцессорность: Хорошо работает с несколькими процессорами.
  - Поддержка языков: Кодирование на различных языках программирования.
  - Эффективное использование памяти: Использует меньше оперативной памяти компьютера.

- **Disadvantages:**
  - Кривая обучения: Очень сложная.



- **Advantages:**
  - Производительность: Оптимизирована для высокой эффективности в многопроцессорных средах с несколькими GPU.
  - Универсальность: Поддерживает широкий спектр типов нейронных сетей.
  - Интеграция: Легко интегрируется с Microsoft Azure.

- **Disadvantages:**
  - Сложность: Сложен и менее интуитивен.
  - Сообщество: Имеет более небольшое сообщество и меньше ресурсов.

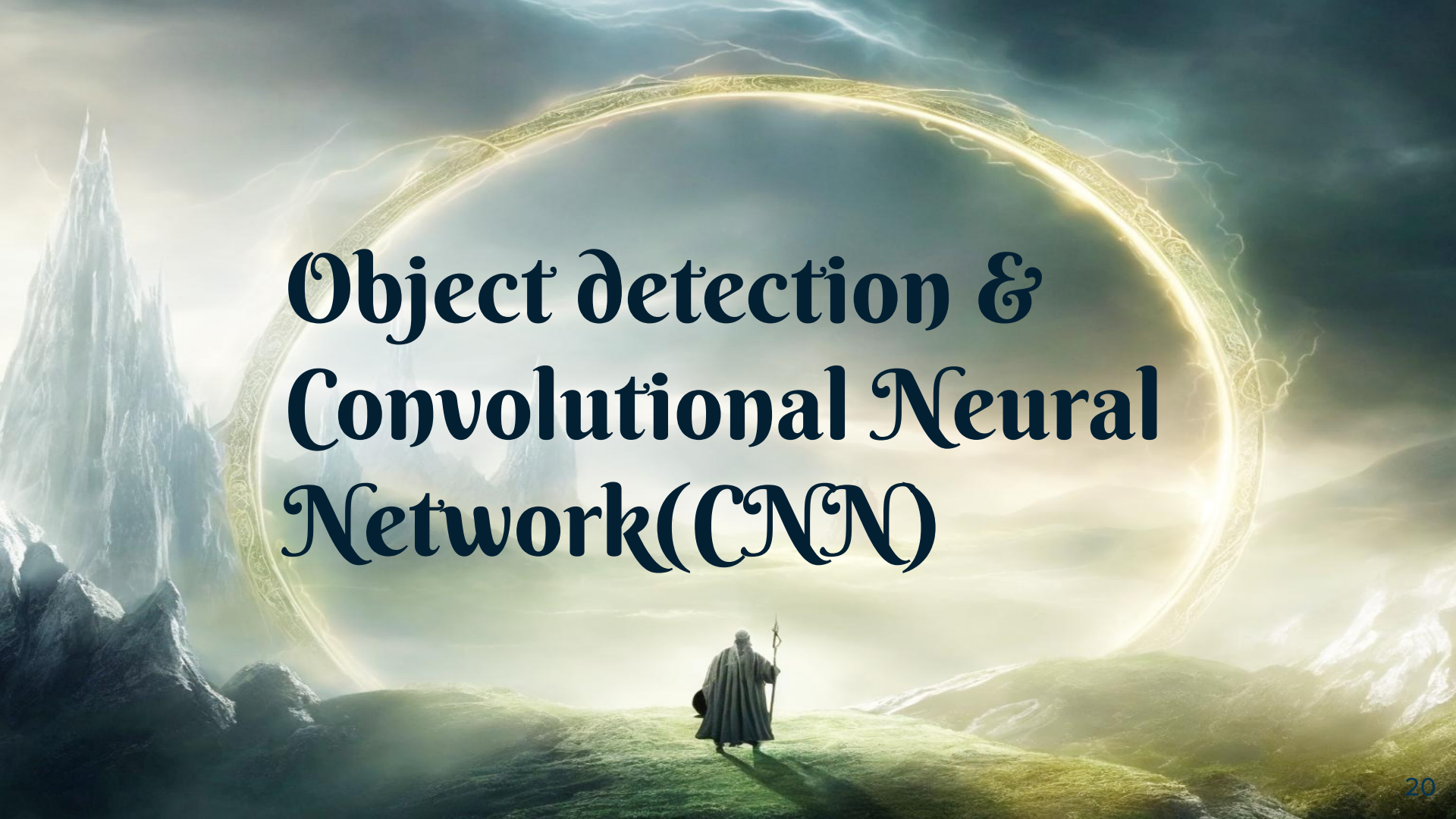
- **Advantages:**
  - Универсальность: Подходит для множества типов проектов.
  - Сообщество: Множество онлайн-помощи и ресурсов.
  - Мобильная совместимость: С TensorFlow Lite удобнее использовать модели на мобильных устройствах.

- **Disadvantages:**
  - Сложность обучения: Может быть не просто.
  - Сложность кодирования: Требует больше строк кода.

Почему мы выбрали TensorFlow?

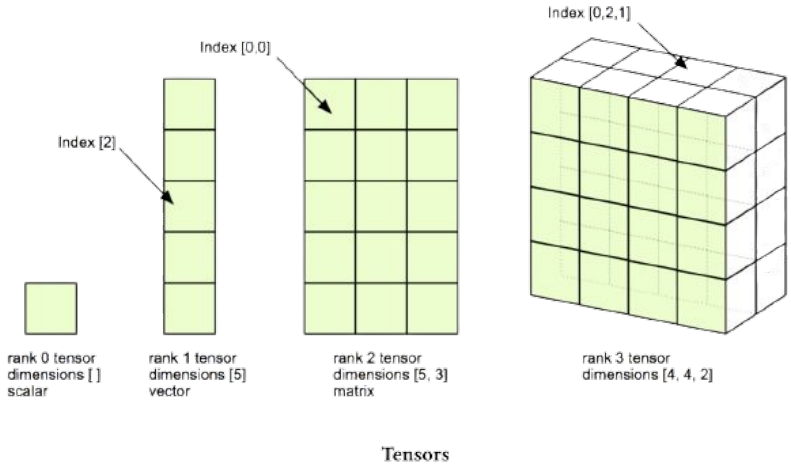
- TensorFlow Lite
- Производительность
- Готовые модели
- Гибкие архитектуры
- Аппаратное ускорение
- Активное сообщество
- Кросс-платформенность

TensorFlow


Object detection &
Convolutional Neural
Network(CNN)

Что такое Tensor?

Тензор - это многомерное числовое представление данных (также называемое n-мерным, где n может быть любым числом).



Что такое Tensor?

- Вы можете преобразовать изображение в тензоры с shape (224, 224, 3), где:
- 224, 224 (первые 2 измерения) - это высота и ширина изображения в пикселях.
- 3 - количество цветовых каналов изображения (красный, зеленый, синий).

```
[[[ 80,  32,  18],
  [ 77,  29,  15],
  [ 75,  27,  13],
       ...,
  [189, 163, 130],
  [196, 170, 137],
  [192, 166, 133]],

 [[ 79,  31,  17],
  [ 76,  28,  14],
  [ 74,  26,  12],
       ...,
  [155, 129,  96],
  [177, 151, 118],
  [192, 166, 133]],
       ...
```

Алгоритмы глубокого обучения(Deep Learning Algorithms)

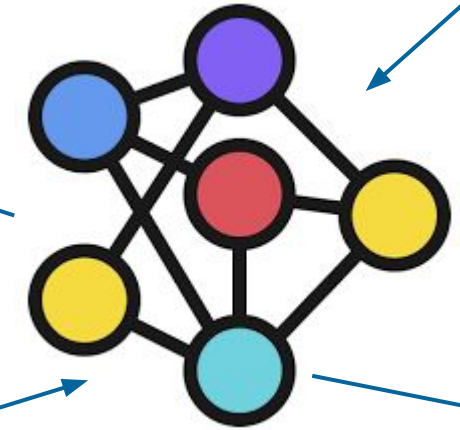
- Neural Networks (NN)
- Convolutional Neural Networks (CNNs)
- Recurrent Neural Networks (RNNs)
- Generative Adversarial Networks (GANs)

Convolutional Neural Network(CNN)

1. Инициализация со случайными весами (только в начале)

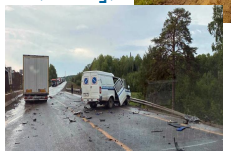
[0.092, 0.210, 0,415],
[0.078, 0.929, 0.030],
[0.019, 0.182, 0.555],
....,

2. Примеры



3. Больше примеров

[116, 78, 15],
[117, 43, 96],
[125, 87, 23],
...



4. Обновление выходных данных

[0.092, 0.210, 0,415],
[0.078, 0.929, 0.030],
[0.019, 0.182, 0.555],

5. Результат
Дорожный знак

Создание модели сверточной нейронной сети (CNN).

```
model = tf.keras.models.Sequential([
  tf.keras.layers.Conv2D(filters=10,
                         kernel_size=3,
                         activation="relu",
                         input_shape=(224, 224, 3))
  tf.keras.layers.Conv2D(10, 3, activation="relu"),
  tf.keras.layers.MaxPool2D(pool_size=2,
                            padding="valid"),
  tf.keras.layers.Conv2D(10, 3, activation="relu"),
  tf.keras.layers.Conv2D(10, 3, activation="relu"),
  tf.keras.layers.MaxPool2D(2),
  tf.keras.layers.Flatten(),
  Dense(10, activation='softmax')
])
```

Создание модели сверточной нейронной сети (CNN).

```
model = tf.keras.models.Sequential()
```

- **Что это такое?**

  Это линейный стек слоев в TensorFlow

- **Зачем использовать?**

- Простота
- Beginner-Friendly

Создание модели сверточной нейронной сети (CNN).

```
tf.keras.layers.Conv2D(filters=10,
                       kernel_size=3,
                       activation="relu",
                       input_shape=(224, 224, 3))
```

- **Тип:** Слой свертки.

- **Основные параметры:**
- **filters=10:** Использует 10 различных фильтров.

- **kernel_size=3:** Каждый фильтр имеет размер 3x3 пикселя.

- **activation="relu":** Функция Rectified Linear Unit, f(x)=max(0,x).

- **input_shape=(224, 224, 3):** Ожидает изображения размером 224x224 пикселя с 3 цветовыми каналами (RGB).

Создание модели сверточной нейронной сети (CNN).

```
tf.keras.layers.Conv2D(10, 3, activation="relu")
```

- **Тип:** Ещё один сверточный слой.
- **Что делает:** Дополнительно уточняет обнаружение признаков от предыдущего слоя.
- **Основные параметры:** Использует 10 фильтров размером 3x3 пикселя.

Создание модели сверточной нейронной сети (CNN).

```
tf.keras.layers.MaxPool2D(pool_size=2,
                          padding="valid")
```

- **Тип:** Слой максимальной пулинга.

- **Что делает:** Играет важную роль в уменьшении размерности данных и извлечении ключевых признаков из изображений.

- **Основные параметры:**
- **pool_size=2:** Рассматривает области 2x2 и выбирает наибольшее значение.

- **padding="valid":** Не добавляет дополнительное заполнение к feature map.

Создание модели сверточной нейронной сети (CNN).

```
tf.keras.layers.Conv2D(10, 3, activation="relu"),
tf.keras.layers.Conv2D(10, 3, activation="relu")
```

- **Тип:** Еще больше сверточных слоев.
- **Что делает:** Продолжает уточнять обнаружение признаков.
- **Основные параметры:** Использует 10 фильтров размером 3x3 пикселя.

Создание модели сверточной нейронной сети (CNN).

```
tf.keras.layers.MaxPool2D(2)
```

- **Тип:** Еще один слой максимальной пулинга.
- **Что делает:** Дополнительно уменьшает размер карт признаков.
- **Основные параметры:** Рассматривает области 2x2 и выбирает наибольшее значение.

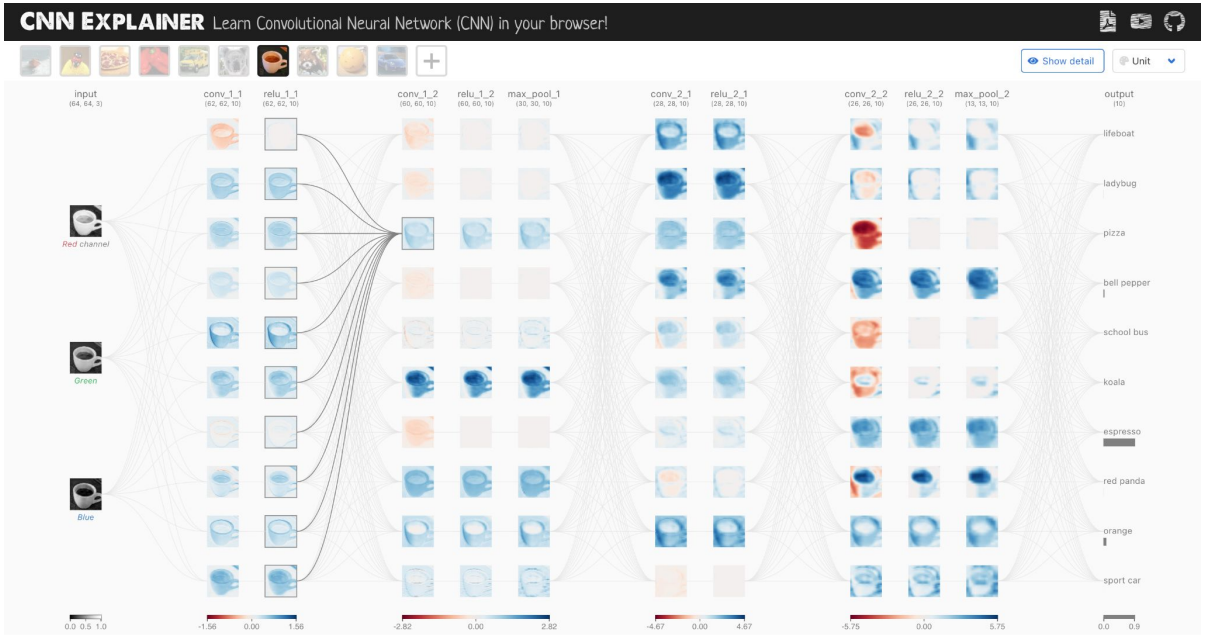
Создание модели сверточной нейронной сети (CNN).

```
tf.keras.layers.Flatten()
```

- **Тип:** Слой выравнивания (Flattening).
- **Что делает:** Преобразует двумерные карты признаков в одномерный вектор, что делает их подходящими для финальных слоев.

Создание модели сверточной нейронной сети (CNN).

https://poloclub.github.io/cnn-explainer/



Настройка модель сверточной нейронной сети (CNN)

```
model.compile(loss="binary_crossentropy",
              optimizer=tf.keras.optimizers.Adam(),
              metrics=["accuracy"])
```

- Функция потерь - "categorical_crossentropy":
- Оптимизатор - Adam
- Метрика - "accuracy"

Обучение модели сверточной нейронной сети (CNN)

Обучение модели сверточной нейронной сети (CNN)

```
model.fit(train_data,
          epochs=5,
          steps_per_epoch=len(train_data),
          validation_data=valid_data,
          validation_steps=len(valid_data))
```

- **Данные обучения - train_data:**
- **Эпохи - epochs=5:**
- **Шаги в каждой эпохе - steps_per_epoch=len(train_data)**
- **Данные проверки - validation_data=valid_data**
- **Шаги проверки - validation_steps=len(valid_data)**

Прогноз с использованием модели CNN

```
model.predict(img)
```

Image shape: (512, 512, 3)
Road blockage

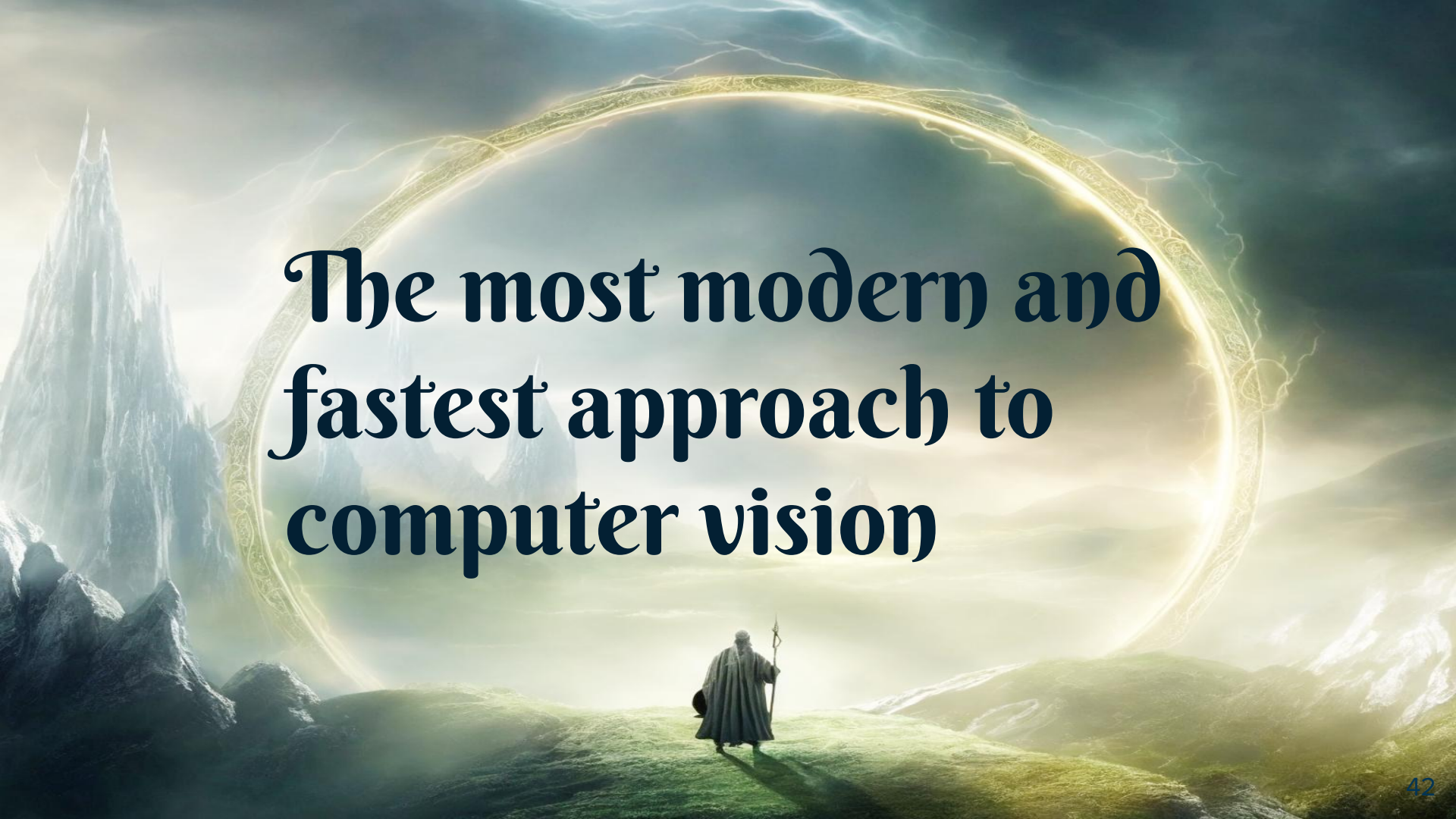
Image shape: (512, 512, 3)
Road sign

Ошибки прогнозирования

Image shape: (512, 512, 3)
Free parking space

Ошибки прогнозирования

Image shape: (512, 512, 3)
Car collision


The most modern and fastest approach to computer vision

YOLO, как самый современный и быстрый подход к компьютерному зрению.

YOLO является аббревиатурой, которая расшифровывается как **You Only Look Once**

YOLO, как самый современный и быстрый подход к компьютерному зрению.

Почему мы решили использовать YOLO вместо собственной модели?

- **Скорость:**
  - Обрабатывает изображения в режиме реального времени.

- **Точность:**
  - Несмотря на свою скорость, YOLO поддерживает конкурентоспособную точность по сравнению с другими лучшими методами обнаружения.

- **Унифицированная архитектура:**
  - Обнаружение объектов в один проход

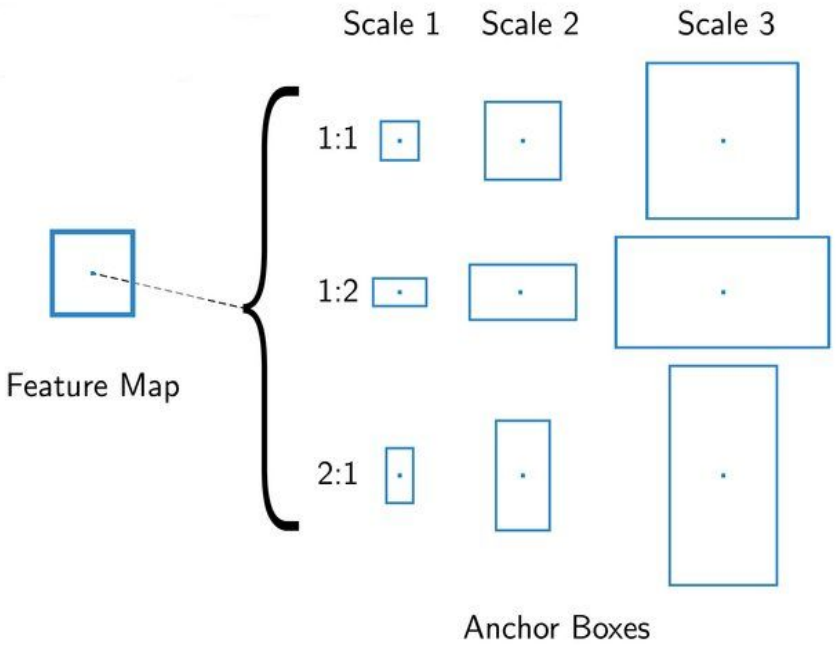
- **Предварительно обученные модели**

- **Масштабируемость:**
  - Tiny-YOLO для ограниченных устройств

Концепция якорных рамок (Anchor Boxes):

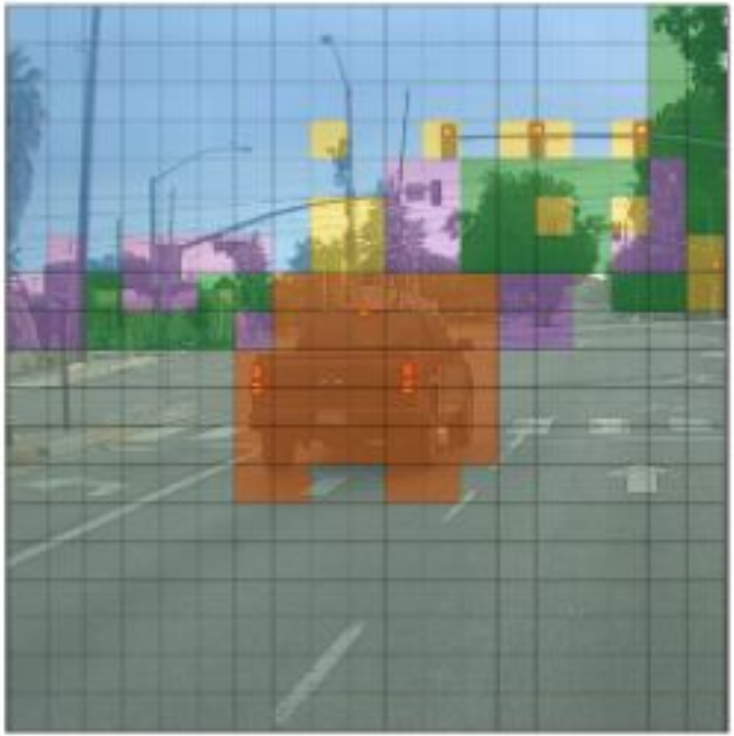
- **Что такое якорные рамки (Anchor Boxes)?**
  - Якорные рамки - это предварительно определенные формы ограничивающих рамок, которые используются для более точного предсказания форм объектов в алгоритме YOLO (You Only Look Once).

- **Зачем они нужны?**
  - Несколько объектов в одной ячейке сетки
  - Разнообразие форм



Что такое детекция на основе сетки (Grid-based Detection)?

YOLO делит изображение на сетку (например, 13x13). Каждая сеточная ячейка отвечает за предсказание ограничивающих рамок и вероятностей классов объектов, центр которых находится в пределах этой ячейки.



Что такое детекция на основе сетки (Grid-based Detection)?

Для каждой ячейки в сетке алгоритм определяет вектор [p, x, y, w, h, conf[n]], где:

p — вероятность того, что в ячейке сетки содержится объект какого-либо класса.

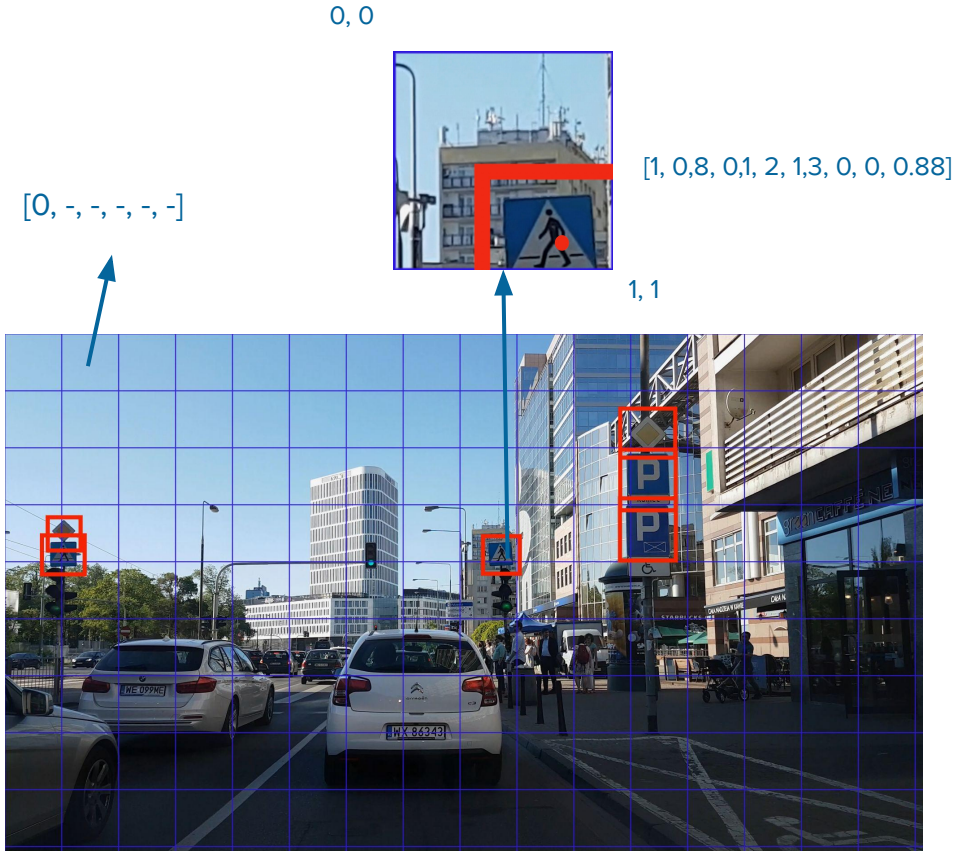
x — координата X центра обнаруженного объекта.

y — координата Y центра обнаруженного объекта.

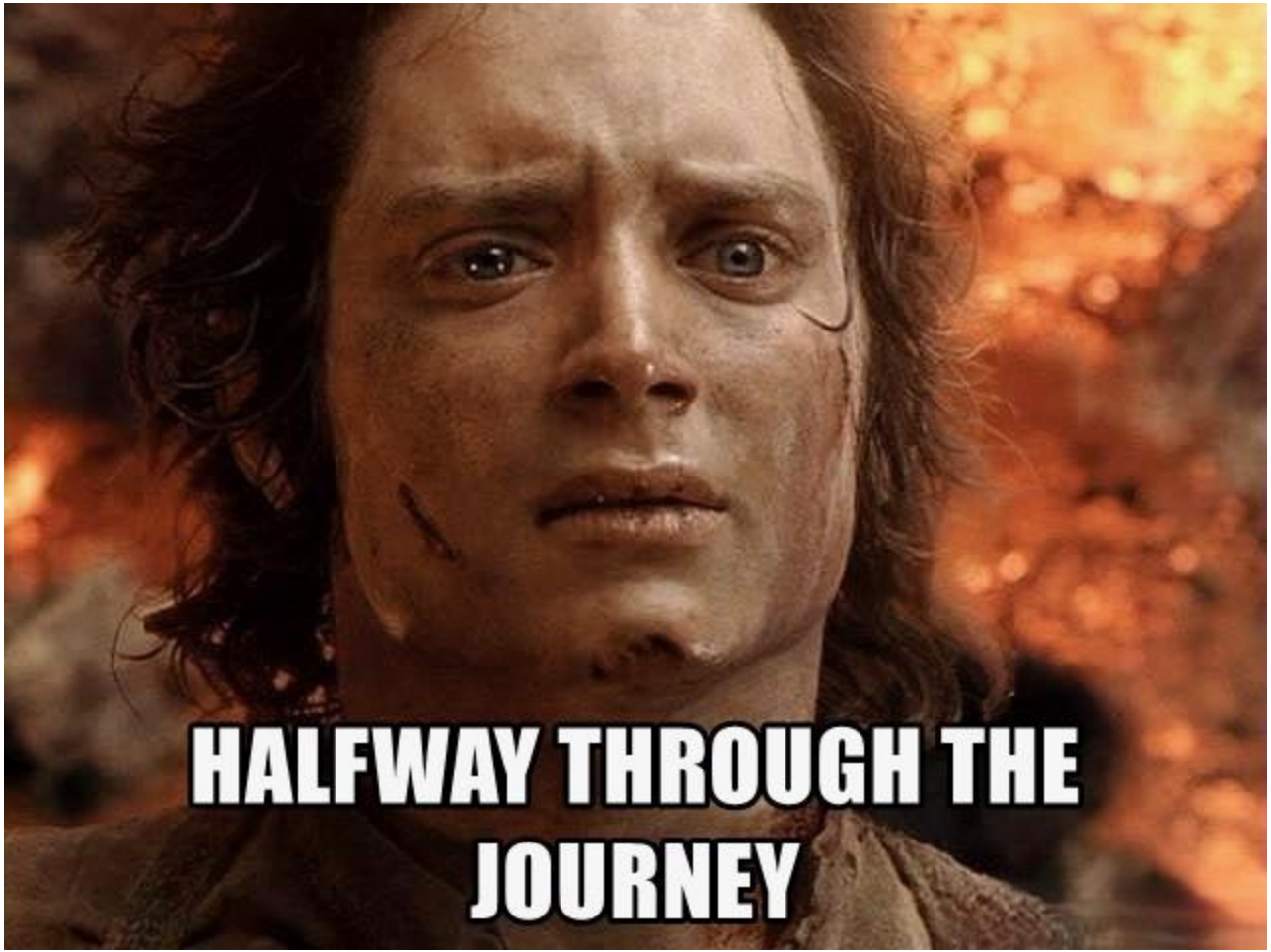
w — ширина обнаруженного объекта.

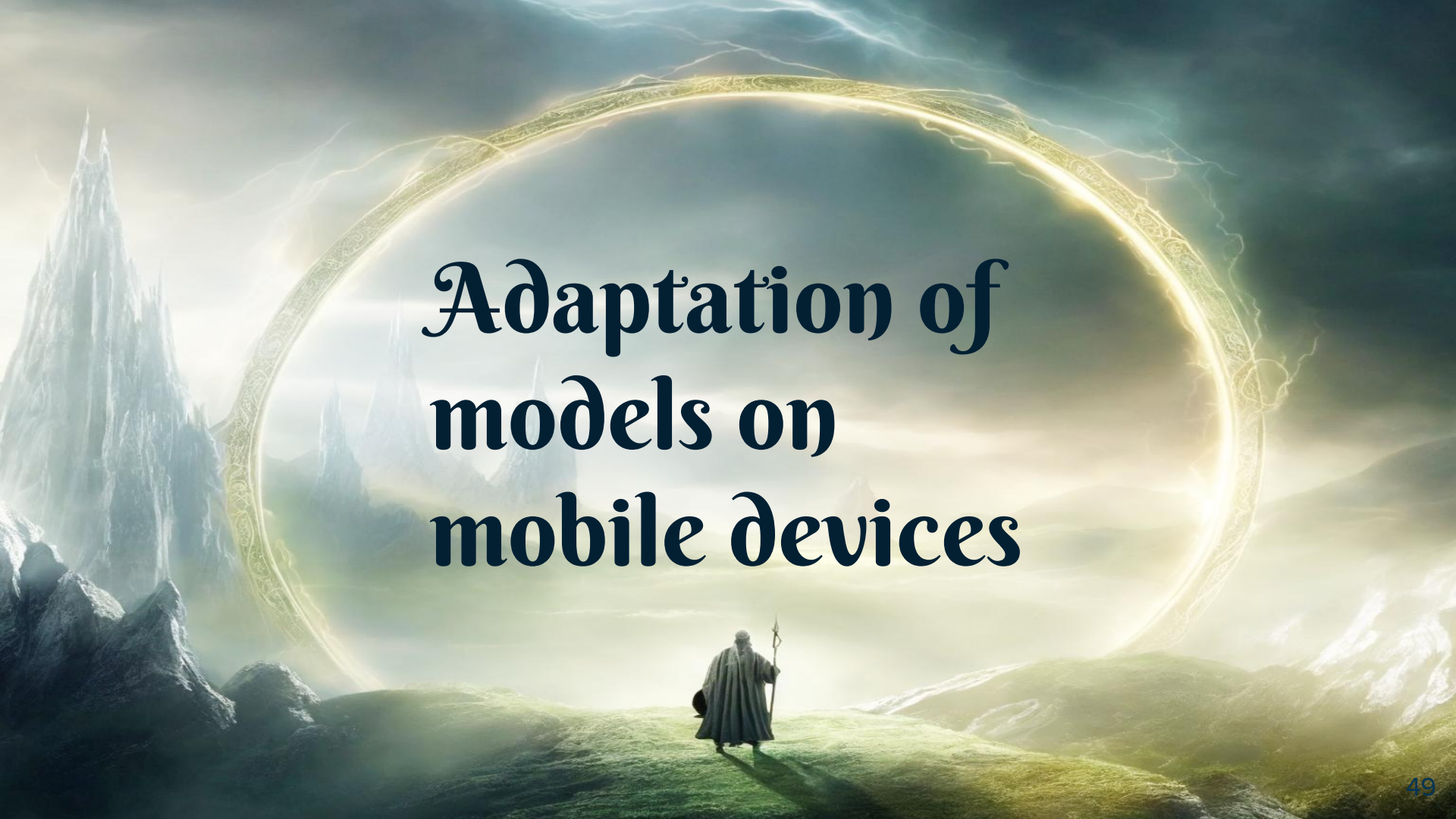
h — высота обнаруженного объекта.

conf[n] — уверенность (вероятность) принадлежности объекта к каждому классу n.



YOLO, как самый современный и быстрый подход к компьютерному зрению.




Adaptation of
models on
mobile devices

TensorFlow Lite

YOLO, как самый современный и быстрый подход к компьютерному зрению.

TensorFlow Lite

Современный и Быстрый

Давайте кодить

```
dependencies {
...
    // Tensorflow Lite dependencies
    implementation 'org.tensorflow:tensorflow-lite-task-vision-play-services:0.4.2'
    implementation 'com.google.android.gms:play-services-tflite-gpu:16.1.0'
...
}
```

Инициализация TensorFlow Lite с поддержкой GPU

```
import com.google.android.gms.tflite.gpu.support.TfLiteGpu
import com.google.android.gms.tflite.client.TfLiteInitializationOptions


TfLiteGpu.isGpuDelegateAvailable(context).onSuccessTask { gpuAvailable ->
    val options = TfLiteInitializationOptions.builder().apply{
        if (gpuAvailable) {
            setEnableGpuDelegateSupport(true)
        }
    }.build()

    TfLiteVision.initialize(context, options)
}.addOnSuccessListener {
    // Handle success
}.addOnFailureListener {
    // Handle failure
}
```

Инициализация TensorFlow Lite с поддержкой GPU

Преимущества поддержки GPU в прогнозировании TensorFlow:

- **Скорость:**
  - Быстрые матричные операции

- **Эффективность:**
  - GPU способны обрабатывать несколько задач одновременно

- **Сложные модели:**
  - GPU лучше справляются сложными нейронными сетями

- **Обработка в реальном времени**

Конфигурация параметров детектора объектов в TensorFlow

```
import org.tensorflow.lite.task.gms.vision.detector.ObjectDetector

val optionsBuilder =
    ObjectDetector.ObjectDetectorOptions.builder()
        .setScoreThreshold(threshold)
        .setMaxResults(maxResults)
```

`setScoreThreshold(threshold)`: Будут рассматриваться только объекты, обнаруженные с оценкой уверенности выше этого порога.

`setMaxResults(maxResults)`: Ограничивает количество обнаруженных объектов до указанного значения maxResults.

Конфигурация параметров детектора объектов в TensorFlow

```
import org.tensorflow.lite.task.core.BaseOptions

val baseOptionsBuilder =
    BaseOptions.builder()
        .setNumThreads(numThreads)
        .useGpu() // optional
        .useNnapi() // optional
```

`setNumThreads(numThreads)`: Настраивает количество потоков ЦПУ, используемых для операций.

`useGpu()`: Активирует поддержку GPU для ускорения вычислений.

`useNnapi()`: Использует Android Neural Networks API для аппаратного ускорения вывода.

Android Neural Networks API

- **Что такое NNAPI?**
  - NNAPI расшифровывается как Neural Networks API.

- **Зачем использовать NNAPI?**
  - Аппаратное ускорение: NNAPI предназначен для предоставления абстракции над различными аппаратными ускорителями (такими как GPU, DSP и NPU), доступными на устройствах Android.

- **Как работает .useNnapi()?**
  - Когда вы вызываете .useNnapi() в TensorFlow Lite, вы указываете фреймворку делегировать совместимые операции в NNAPI

Конфигурация параметров детектора объектов в TensorFlow:

```
import org.tensorflow.lite.task.gms.vision.detector.ObjectDetector

val objectDetector =
    ObjectDetector.createFromFileAndOptions(
        context,
        modelName,
        optionsBuilder.build()
    )
```

`modelName`:
Имя или путь к предварительно обученному файлу модели формата .tflite.

Подготовка данных для модели

```
import org.tensorflow.lite.support.image.ImageProcessor;
import org.tensorflow.lite.support.image.ops.ResizeWithCropOrPadOp;
import org.tensorflow.lite.support.image.ops.NormalizeOp;
import org.tensorflow.lite.support.image.ops.QuantizeOp;
import org.tensorflow.lite.support.image.ops.CastOp;

val imageProcessor = ImageProcessor.Builder()
    .add(ResizeCenterFit(modelTargetSize.width, modelTargetSize.height))
    .add(NormalizeOp(floatArrayOf(IMAGE_MEAN), floatArrayOf(IMAGE_STD)))
    .add(CastOp(DataType.FLOAT32))
    .build()
    .process(this)
```

Подготовка данных для модели

ResizeCenterFit:

- **Зачем нужно:**
  - Эта операция используется для изменения размера изображений и, при необходимости, обрезания или дополнения их до заданного размера.

- **Применение:**

1724 × 970

640 × 640

Подготовка данных для модели

```
import android.graphics.Bitmap
import android.graphics.Canvas
import android.graphics.Color
import android.graphics.PointF
import kotlin.math.max
import org.tensorflow.lite.support.image.ImageOperator
import org.tensorflow.lite.support.image.TensorImage

class ResizeCenterFit(height: Int, width: Int) : ImageOperator {
    private val maxResolution = max(height, width)
    override fun apply(image: TensorImage): TensorImage =
        TensorImage.fromBitmap(
            image.bitmap.resizeBitmapImageForFitSquare(maxResolution)
        )
    …
}
```

Подготовка данных для модели

```
fun Bitmap.resizeBitmapImageForFitSquare(maxResolution: Int): Bitmap {
    val ratio =
        if (width >= height) {
            maxResolution.toFloat() / width
        } else {
            maxResolution.toFloat() / height
        }

    val finalWidth = (width.toFloat() * ratio).toInt()
    val finalHeight = (height.toFloat() * ratio).toInt()

    val scaledImage = Bitmap.createScaledBitmap(this, finalWidth, finalHeight, true)

    return if (scaledImage.width == scaledImage.height) {
        scaledImage
    } else {
        val left = if (scaledImage.width != maxResolution) {
            (maxResolution - scaledImage.width) / 2
        } else {
            0
        }

        val top = if (scaledImage.height != maxResolution) {
            (maxResolution - scaledImage.height) / 2
        } else {
            0
        }

        return Bitmap.createBitmap(maxResolution, maxResolution, Bitmap.Config.ARGB_8888).also {
            Canvas(it).apply {
                drawColor(Color.WHITE)
                drawBitmap(scaledImage, left.toFloat(), top.toFloat(), null)
                save()
                restore()
            }
        }
    }
```

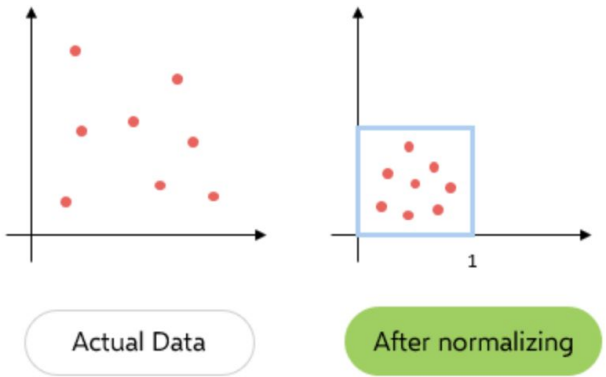
Подготовка данных для модели

NormalizeOp:

- **Зачем нужно:**
  - Нормализация входных данных

- **Применение:**

```
val mean = floatArrayOf(0.45f*255,0.40f*255, 0.35f*255)
val std = floatArrayOf(0.28f*255, 0.26f*255, 0.25f*255)
val output = (input - mean) / stddev
```



Подготовка данных для модели

CastOp:

- **Зачем нужно:** Эта операция выполняет преобразование типов данных. Например, она может быть использована для преобразования входных данных из целых чисел в числа с плавающей запятой или наоборот.

- **Применение:** Если формат ваших данных не соответствует ожиданиям модели, CastOp может быть использована для приведения данных к правильному типу.

Запуск модели

```
import org.tensorflow.lite.support.image.ImageProcessor;
import org.tensorflow.lite.support.image.TensorImage;
import org.tensorflow.lite.task.gms.vision.detector.ObjectDetector;
import org.tensorflow.lite.task.vision.detector.Detection

val tensorImage: TensorImage =
imageProcessor.process(TensorImage.fromBitmap(image))

val detections: List<Detection> = objectDetector.detect(tensorImage)

class Detection {
    fun getBoundingBox(): RectF
    fun getCategories(): List<Category>
}

class Category {
    fun getLabel(): String
    fun getScore(): Float
}
```

Non-max Suppression

TensorFlow Lite

- **Что такое Non-max Suppression (NMS)?**
  - NMS (немаксимальное подавление) - это техника постобработки, используемая в задаче обнаружения объектов, чтобы убедиться, что один и тот же объект обнаруживается только один раз, а не несколько раз с использованием нескольких перекрывающихся рамок.

- **Зачем это нужно?**
  - Перекрывающиеся предсказания
  - Оценки уверенности

multiple bounding boxes

final bounding boxes

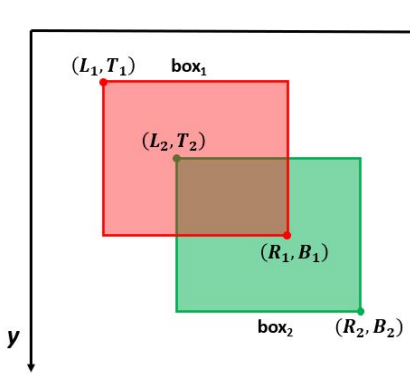
Non-max Suppression

TensorFlow Lite

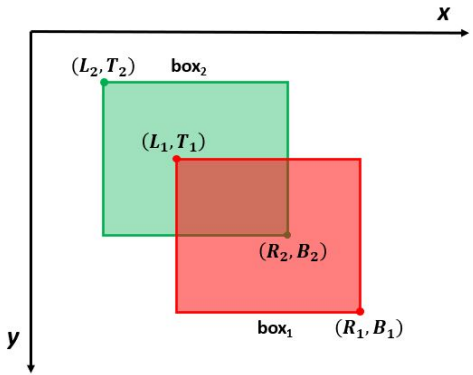
```
val detections = listOf<Detection>()
val sortedDetections = ArrayDeque(
    detections.sortedByDescending {it.getCategories().first().getScore() }
)

val results = mutableListOf<Detection>()

while (sortedDetections.isNotEmpty()) {
    val topScoreDetection = sortedDetections.removeFirst()

    list.add(topScoreDetection)

    sortedDetections.forEachIndexed { index, detection ->
        val iou = intersectionOverUnion(topScoreDetection, detection)

        if (iou > threshold) {
            sortedDetections.removeAt(index)
        }
    }
}

return results
```

TensorFlow Lite

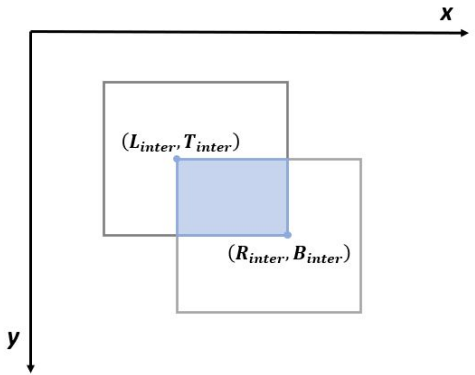
Intersection over Union (IoU)



Case 1



Case 2



Intersection rectangle

Пост-обработка

Мы должны вернуть результат к исходным значениям, а затем найти соотношение размера исходного кадра к размеру кадра, получившемуся после обработки моделью.

```
val results: List<Detection>

val adaptedResults = detectionResults.map {
    Detection.create(it.boundingBox.adapt(), it.categories)
}
```

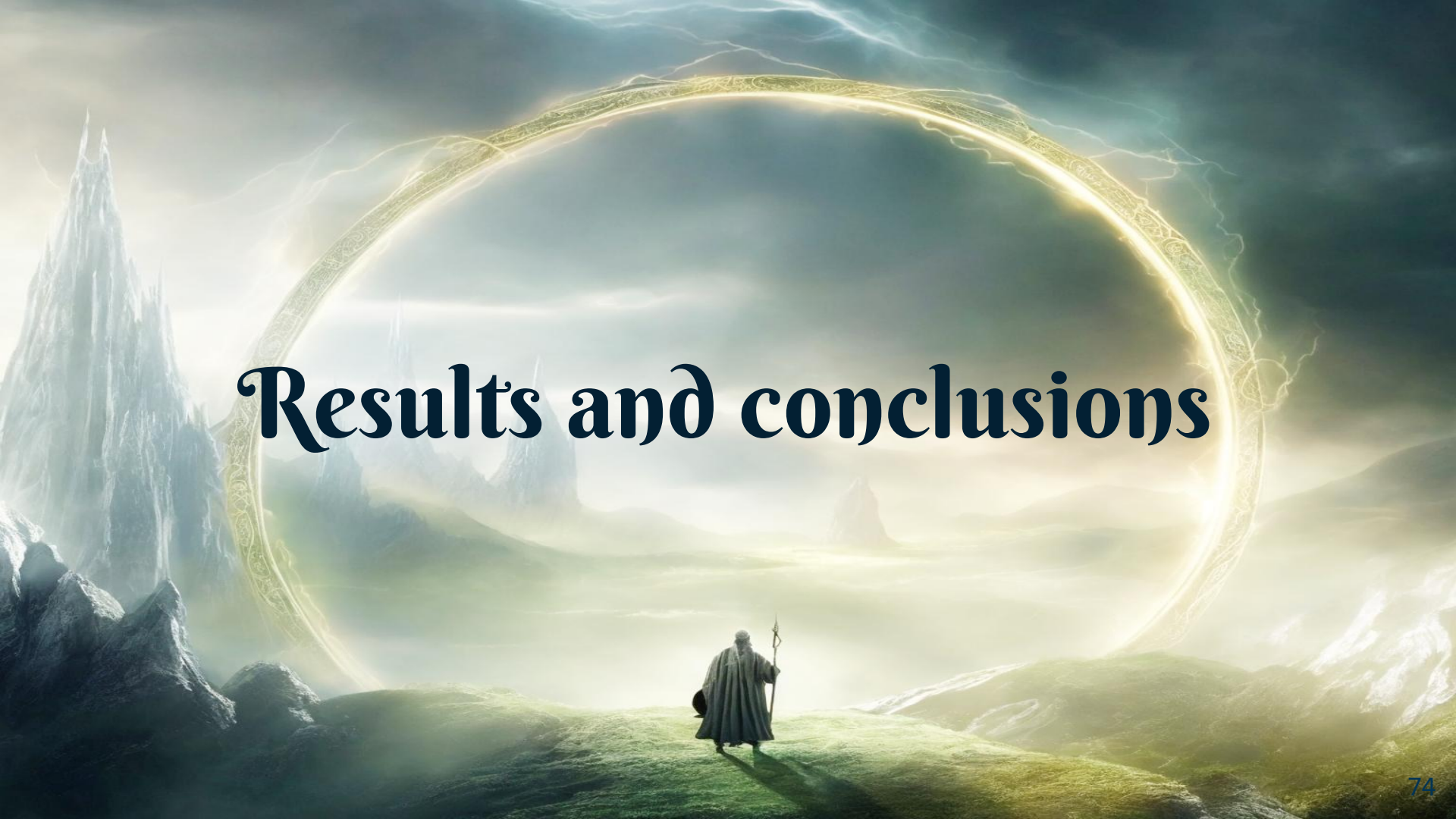
Пост-обработка


```
private fun RectF.adapt(
    modelHeight: Float,
    modelWidth: Float,
    frameHeight: Float,
    frameWidth: Float
): RectF {

    val denormalLeft = left * modelWidth
    val denormalTop = top * modelHeight
    val denormalRight = right * modelWidth
    val denormalBottom = bottom * modelHeight

    …
}
```

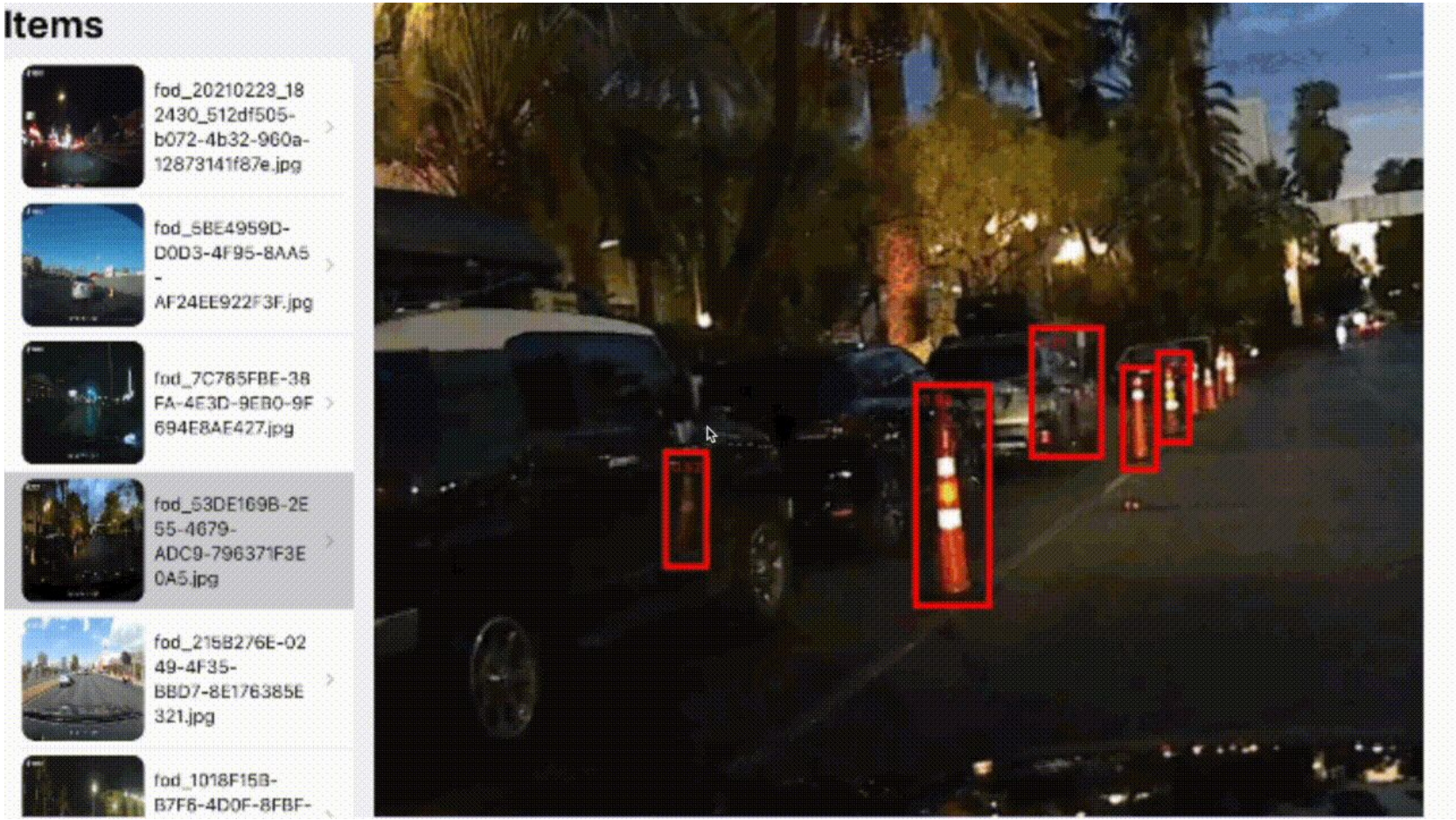
Пост-обработка


```
{
    ...
    val originalAspectRatio = frameWidth / frameHeight

    val imageFittedHeight = modelHeight / originalAspectRatio
    val imageFittedWidth = modelWidth / originalAspectRatio

    val imageOffsetHeight = modelHeight - imageFittedHeight
    val imageOffsetWidth = modelWidth - imageFittedWidth

    val widthRatio = frameWidth / imageFittedWidth
    val heightRatio = frameHeight / imageFittedHeight

    val adaptedLeft =
        max(frameHeight, denormalLeft - modelWidth / 2) * widthRatio - imageOffsetWidth
    val adaptedTop =
        max(frameHeight, denormalTop - modelHeight / 2) * heightRatio - imageOffsetHeight
    val adaptedRight =
        min(frameWidth, denormalRight + modelWidth / 2) * widthRatio - imageOffsetWidth
    val adaptedBottom =
        min(frameHeight, denormalBottom + modelHeight / 2) * heightRatio - imageOffsetHeight

    return RectF(adaptedLeft, adaptedTop, adaptedRight, adaptedBottom)
}
```

Когда пытаешься понять, за что отвечают параметры в TensorFlow


Results and conclusions

Результаты и выводы



- **Результаты:**
  - Модель доработана и обновлена после внутреннего тестирования
  - Средняя точность модели исходя из внутренней аналитики компании - 78%
  - Модуль успешно внедрена и проходит тестирование на реальных данных

- **Выводы:**
  - Базовые знания ML необходимы современным инженерам-разработчикам
  - Документируйте созданные модели и инструменты
  - Не бойтесь экспериментировать с deep learning фреймворками
  - Ознакомьтесь с сертификацией TensorFlow Developer Certificate