Введение в нейронные сети

Урок 7. Детектирование объектов

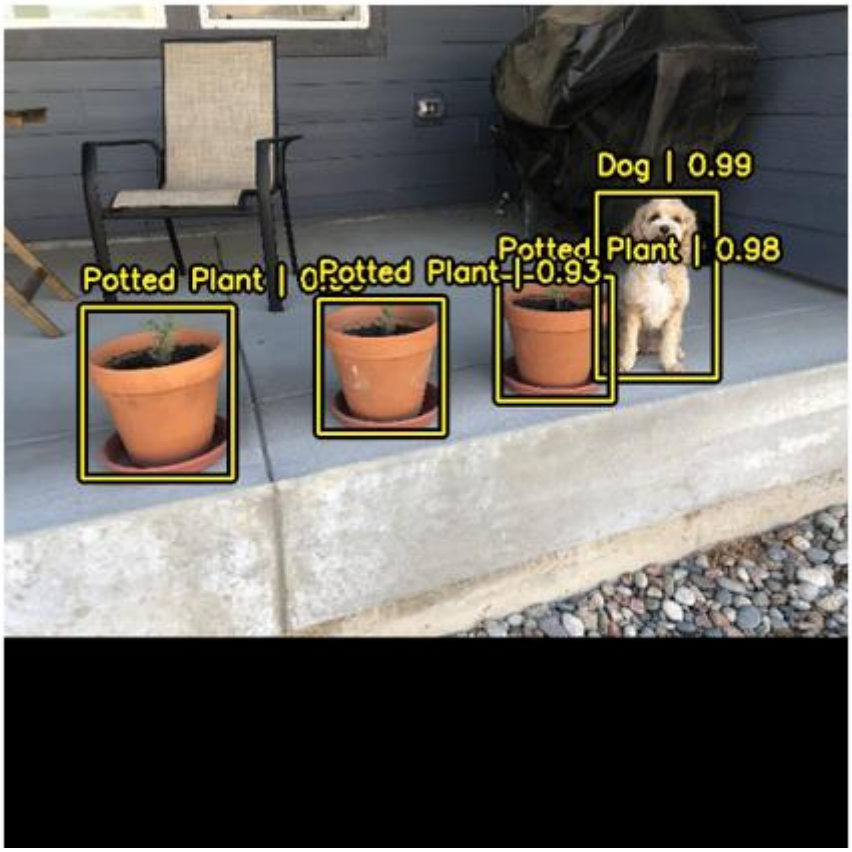
Либо 1 пункт либо пункт 2.

1. Сделайте краткий обзор любой научной работы, посвящённой алгоритму для object detection, не рассматривавшемуся на уроке. Проведите анализ: чем отличается выбранная вами архитектура нейронной сети от других? В чём плюсы и минусы данной архитектуры? Какие могут возникнуть трудности при применении этой архитектуры на практике?

2. Поиграться с кодом урока в разделе Advanced, поменять код и написать свои выводы.

Задания и ответы:

1.



2.

ADVANCED

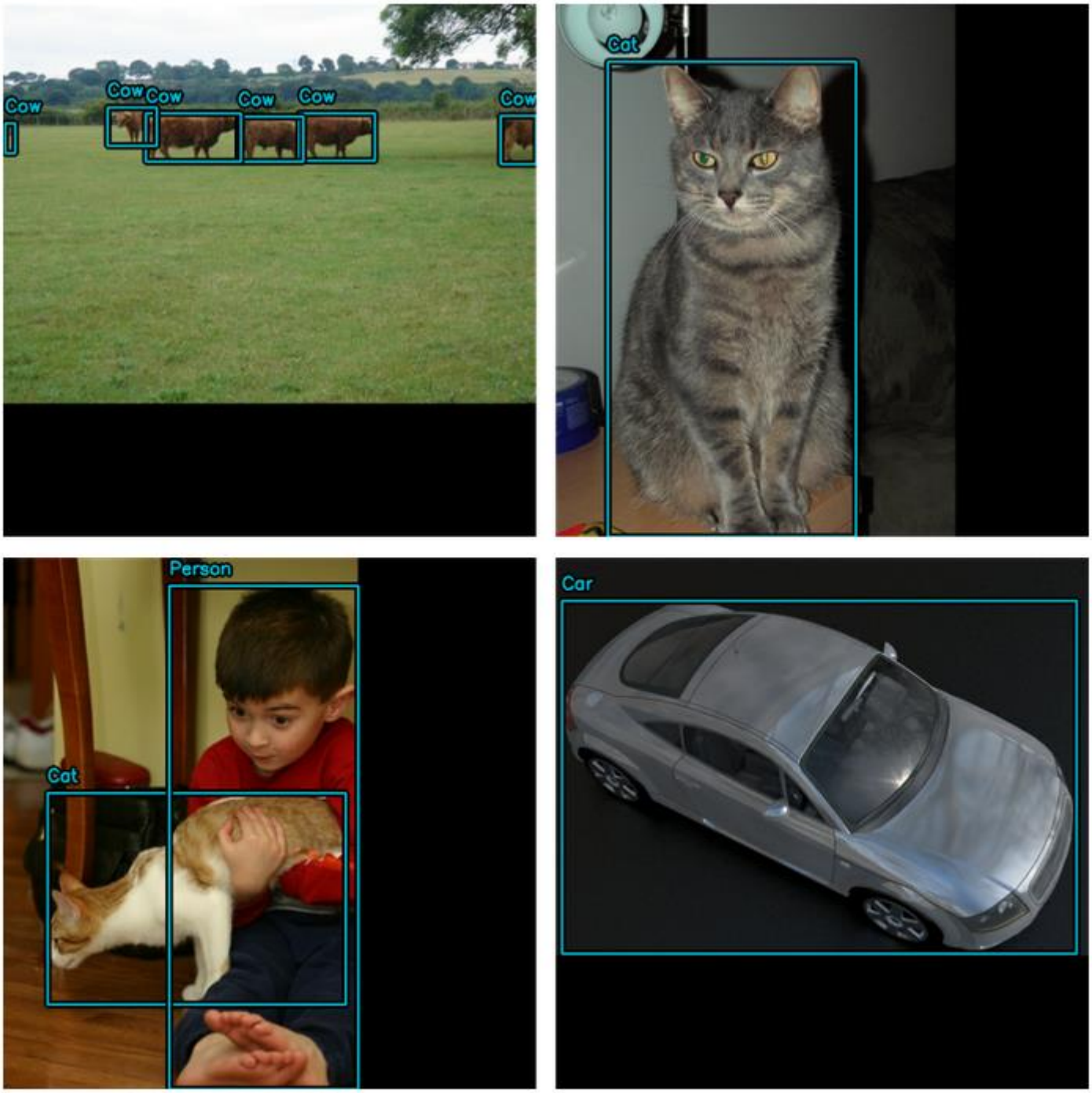
```
model = keras_cv.models.YOLOV8Detector.from_preset(
    "resnet50_imagenet",
    # For more info on supported bounding box formats, visit
    # https://keras.io/api/keras_cv/bounding_box/
    bounding_box_format="xywh",
    num_classes=20,
)
```

```
model.compile(
    classification_loss="binary_crossentropy",
    box_loss="ciou",
    optimizer=optimizer,
)
```

```
model.fit(
    train_ds.take(20),
    # Run for 10-35~ epochs to achieve good scores.
    epochs=1,
    callbacks=[coco_metrics_callback],
)
```

```
20/20 [==============================] - 486s 24s/step
creating index...
index created!
creating index...
index created!
Running per image evaluation...
Evaluate annotation type *bbox*
DONE (t=0.44s).
Accumulating evaluation results...
DONE (t=0.19s).
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.000
20/20 [==============================] - 2304s 115s/step - loss: 51.6753 - box_loss: 3.1318 - class_loss: 48.5435 - val_AP: 0.0000e+00
<keras.src.callbacks.History at 0x7801bf074730>
```

```
1/1 [==============================] - 110s 110s/step
```



Потрясающий результат!

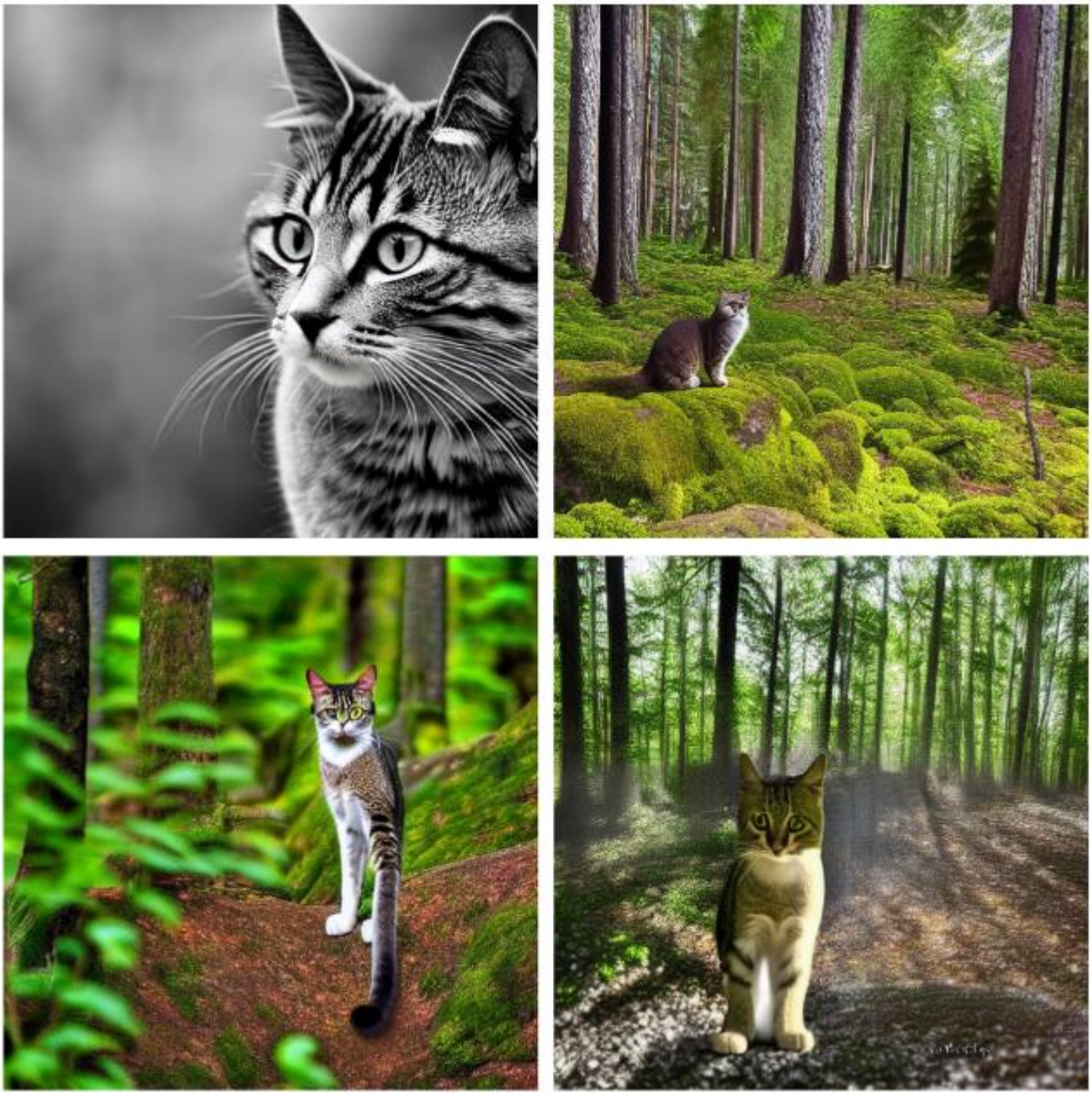
Выводы о KerasCV:

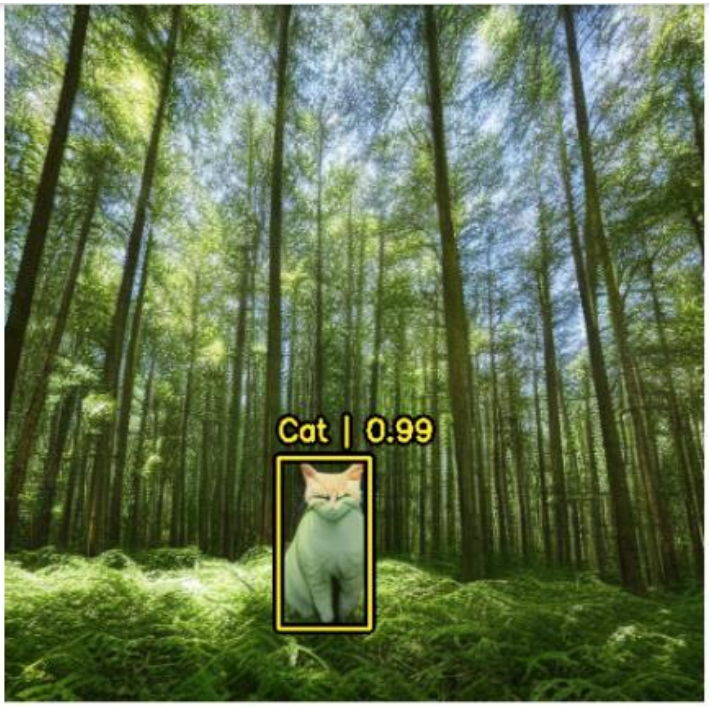
- KerasCV позволяет легко создавать современные конвейеры обнаружения объектов.
  Мы начали с написания загрузчика данных с использованием KerasCV, спецификации ограничивающей рамки.
  После этого мы собрали конвейер увеличения данных промышленного уровня, используя Уровни предварительной обработки KerasCV в <50 строках кода.
- Компоненты обнаружения объектов KerasCV могут использоваться независимо, но они также имеют глубокую интеграцию друг с другом.
- KerasCV обеспечивает увеличение ограничивающей рамки пользовательского уровня, обучение моделей, визуализацию, а метрическая оценка достаточно проста.

Ещё немного выводов о проблемах и итогах:

1. Есть проблема в совместимости версий TensorFlow и Keras.
   В первый раз блокнот запустился. Но потом закончилось время работы в Colab, и пришлось начинать всё сначала. К сожалению, вновь блокнот не запустился...
   Пришлось решать проблему, которая оказалась в совместимости версий.
   См. Ячейка №1 и Ячейка №2.
   В итоге проблема решена, но на это ушло достаточно много времени.
2. Очень долго идёт обучение, заканчивается бесплатное использовоние GPU в процессе обучение.
   Есть возможность использования локального GPU достаточно мощной видео-карты. Но здесь тоже не так всё просто. Буду разбираться.
   Как вариант - ипользование возможностей kaggle.
3. Одностадийная модель YOLO достаточно точно провела детекцию, но обучается очень долго...
   Если не решить проблему п.2, то двухстадийные модели похоже на ресурсах Colab не потянуть.
4. **В итоге обучение завершилось успешно!**
   И результат очень порадовал.
   См. картинки выше, раздел "Результаты".

И в завершении забавный результат фрагмента кода, демонстрирующий возможности API KerasCV!




Cat | 0.99


Cat | 0.97


Cat | 0.99


Cat | 0.91