# **Модуль генерации данных**

Для реализации модуля генерации данных я создала 2 класса: *SetArea* и *Figures*. Решила представить плоскость 256х256 как множество кубов (в блокноте реализована версия 128х128 и 64х64). Для каждого экземпляра класса *SetArea* (a1, a2, a3, c1, c2 и т.д.) прописала свойства:

* x0, y0 - левый верхний угол куба
* shift\_x, shift\_y – смещения относительно осей.

Таким образом, в начале работы я знала координаты bboxes. Это помогло в дальнейшем при решении задачи детекции.

В классе *Figures* реализовала функции создания геометрических фигур: ромб, квадрат, круг, шестиугольник, треугольник. В функции *get\_image* прописала списки всевозможных комбинации кубов 64х64 и 128х128, учитывая, что фигуры не должны пересекаться и выходить за пределы области.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Примеры работы модуля | | |

В *get\_image* создаётся изначальный “холст” (нулевая матрица размером 256х256х3) с помощью функции *get\_blank.* Затем, в зависимости от числа фигур (которое генерируется функцией *count\_figures*), последовательным вызовом функции *get\_image*, каждая из фигур постепенно наносится на матрицу, где уже нанесена предыдущая фигура (и, которая является одним из возвращаемых аргументов функции *get\_figure*).

# **Детекция примитивных геометрических фигур на изображении**

Для рализации выбрала one-stage детектор YOLOv5, так как это хорошая модель в соотношении качество/скорость.

|  |
| --- |
|  |
| Устройство YOLO |

В основе любого детектора лежит свёрточная нейронная сеть. У YOLOv5 CSP-Darknet53 - сверточная сеть Darknet53, используемая в качестве основы для YOLOv3, к которой авторы применили стратегию перекрестной частичной сети (CSP). Сеть использует остаточные и плотные блоки, чтобы обеспечить поток информации в самые глубокие слои и преодолеть проблему затухающего градиента.

Блок SPP выполняет агрегирование информации, поступающей с входных данных, и возвращает выходные данные фиксированной длины. Таким образом, преимущество заключается в значительном увеличении поля восприятия и выделении наиболее релевантных контекстных функций без снижения скорости сети.

Завершающая часть сети (Head) состоит из трех слоев свертки, которые предсказывают расположение ограничивающих рамок (x, y, высота, ширина), оценки и классы объектов.

# **Обучение**

Обучила модель для 100 и 1000 изображений в течение 25 эпох.

|  |  |
| --- | --- |
| **Фигура** | **Идентификатор класса** |
| круг | 0 |
| квадрат | 1 |
| ромб | 2 |
| шестиугольник | 3 |
| треугольник | 4 |

|  |
| --- |
| C:\Users\yavor\Desktop\result_1_60_20_20_without_augm\train_batch0.jpg |
| C:\Users\yavor\Desktop\result_1_60_20_20_without_augm\train_batch1.jpg |
| Примеры размеченных батчей с аугментацией |

Метрики после обучения на 100 изображениях

|  |
| --- |
| C:\Users\yavor\Desktop\result_1_60_20_20_without_augm\P_curve.png |
| C:\Users\yavor\Desktop\result_1_60_20_20_without_augm\R_curve.png |
| C:\Users\yavor\Desktop\result_1_60_20_20_without_augm\PR_curve.png |
| C:\Users\yavor\Desktop\result_1_60_20_20_without_augm\confusion_matrix.png |
| C:\Users\yavor\Desktop\result_1_60_20_20_without_augm\labels.jpg |

Метрики после обучения на 1000 изображений

|  |
| --- |
| C:\Users\yavor\Desktop\result_2_600_200_200_without_aug\P_curve.png |
| C:\Users\yavor\Desktop\result_2_600_200_200_without_aug\R_curve.png |
| C:\Users\yavor\Desktop\result_2_600_200_200_without_aug\confusion_matrix.png |
| C:\Users\yavor\Desktop\result_2_600_200_200_without_aug\F1_curve.png |
| C:\Users\yavor\Desktop\result_2_600_200_200_without_aug\labels.jpg |

# **Выводы**

Логично, что точность предсказаний растёт с увеличений выборки и количества эпох обучения. Например, разница в True Positive ответах модели 100 и 1000 хорошо видна по confusion matrix.

В первом случае при предсказании любого класса модель больше ошибалась, чем давала правильный ответ (всего в 30% случаев, где-то и 0). Модель 1000 хорошо различает все 5 фигур, но вот, например, при определении отсутствия или же присутствия квадрата на изображении возникают небольшие сложности. Я думаю, это объяснимо тем, что на некоторых изображениях обучающей выборки квадраты вписаны на всю площадь bbox. Поэтому, возможно, модель немного не понимает фигура ли перед ней или просто фон.

Дисбаланс классов для модели 100 объясним небольшим количеством сгенерированных изображений, что подтверждает график распределения классов для модели 1000.

Precision score для модели 100 “застрял” в районе 0.5. Можно сказать, что модель почти не обучилась и её ответы скорее были случайными. А вот точность модели 1000 стремится к 1. Это говорит о том, что в своих предсказаниях модель почти не ошибается (TP/TP+FP). Причём круги, шестиугольники и ромбы она определяет увереннее, чем квадраты и треугольники. Это также видно по Recall score (TP/TP+FN)).

Это подтверждают предсказания.

|  |
| --- |
| C:\Users\yavor\Desktop\result_1_60_20_20_without_augm\val_batch0_pred.jpg |
| Предсказания модели 100 на валидационном наборе |
| C:\Users\yavor\Desktop\result_2_600_200_200_without_aug\val_batch2_pred.jpg |
| Предсказания модели 1000 на валидационном наборе |

Невозможность выделения беспроигрышных паттернов для треугольников объяснима высокой вариативностью изображений класса, поэтому для повышения точности классификации треугольников и квадратов нужно больше эпох и примеров.