

Deep
Learning
School

Разбираем детектор по
частям

План

1. Разбиваем детектор на части.
2. Как детектор делает предсказания?
3. Non-Maximum Suppression (NMS), IoU.
4. Метрики качества для задачи детекции.
5. Как детектор обучается?
6. Методы аугментации для детекции.

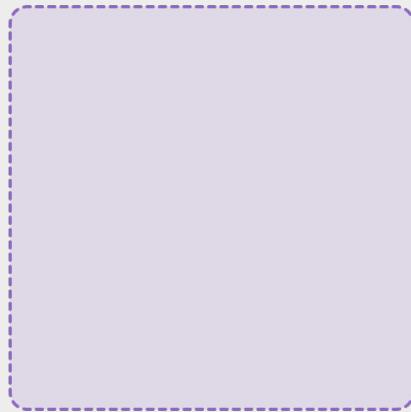


Разбираем детектор на части



Общий вид архитектуры детектора (One-stage)

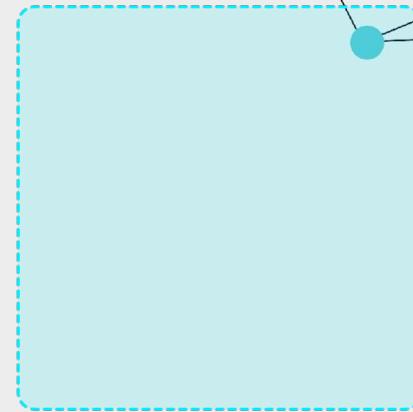
*Backbone or
Feature extractor*



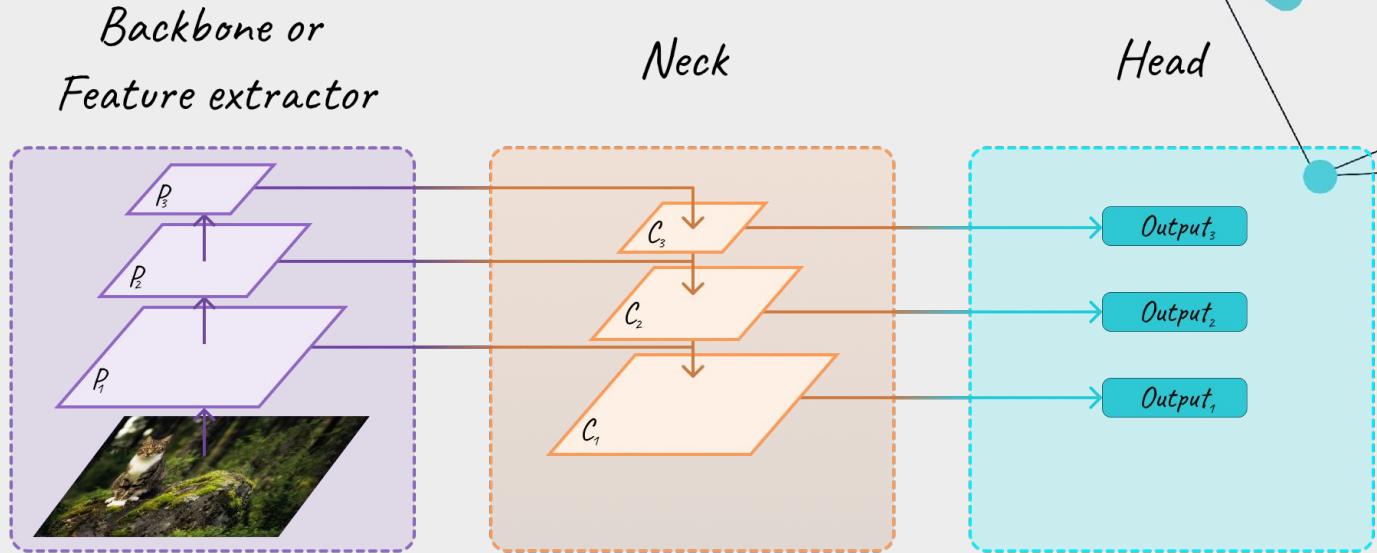
Neck



Head

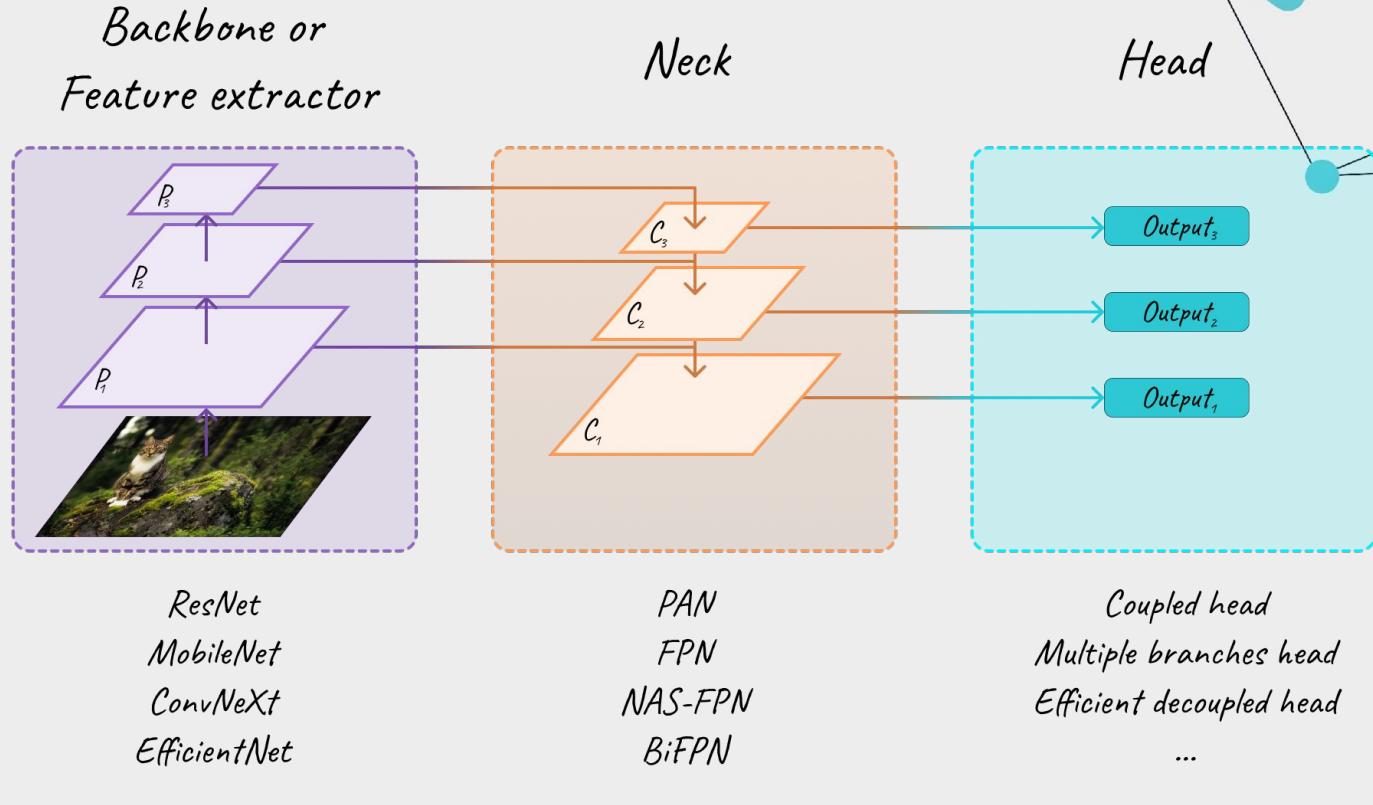


Общий вид архитектуры детектора (One-stage)



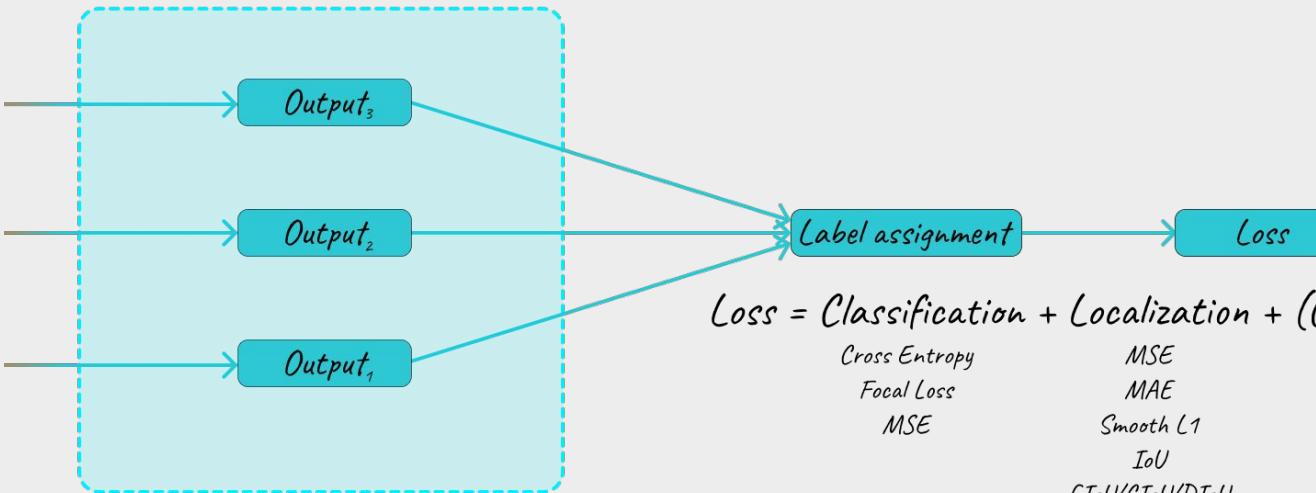
Например, вот так выглядит модель Feature Pyramid Network (FPN)

Общий вид архитектуры детектора (One-stage)



Общий вид архитектуры детектора (One-stage)

Head



$$\text{Loss} = \text{Classification} + \text{Localization} + (\text{Confidence})$$

Cross Entropy

Focal Loss

MSE

MSE

MAE

Smooth L1

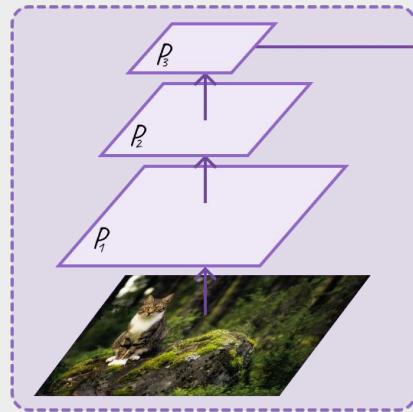
Cross Entropy

IoU
 GIoU/CIoU/DIoU

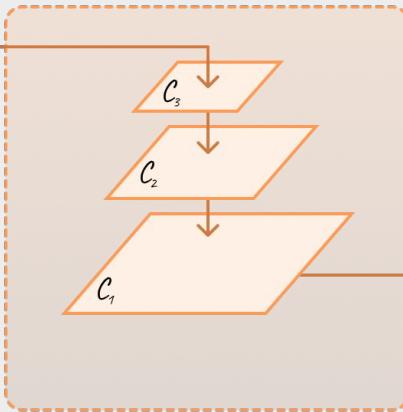


Общий вид архитектуры детектора (One-stage)

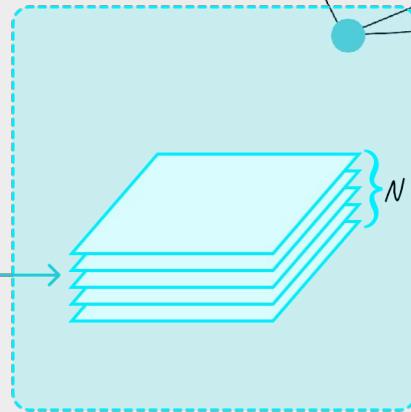
*Backbone or
Feature extractor*



Neck



Head



Выходной вектор размером N :

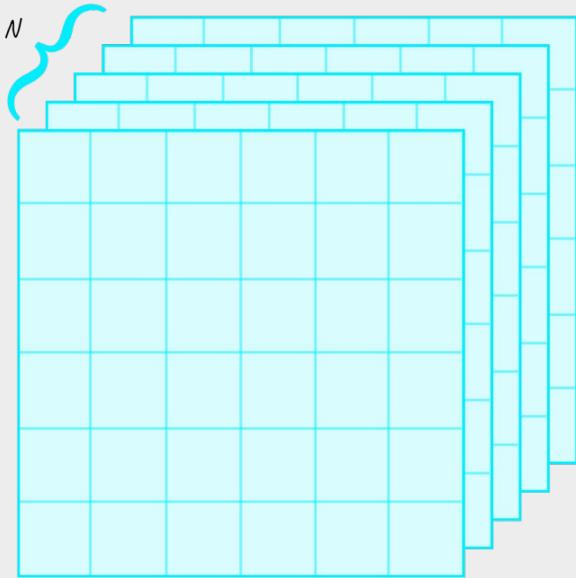
1. 4 координаты блока (x, y, w, h)
2. Confidence score
3. Probability for each class

Как детектор делает предсказания?



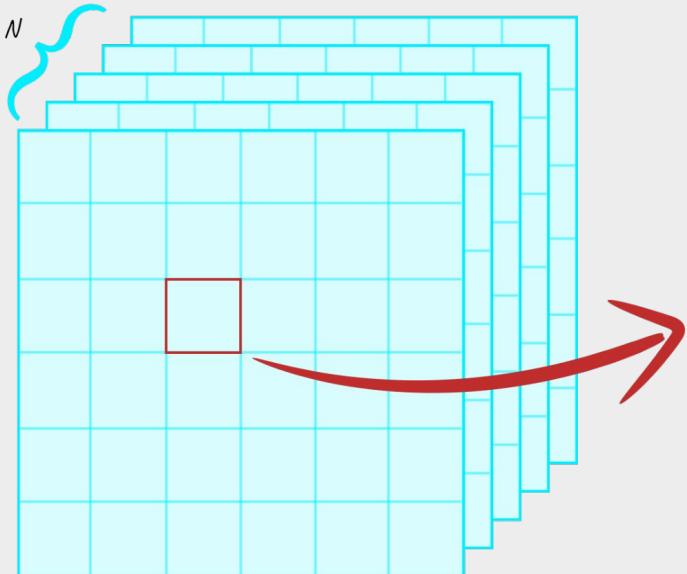
Как детектор делает предсказания?

Output feature map



Как детектор делает предсказания?

Output feature map



Каждый “пиксель” представляет собой вектор, длиной N:

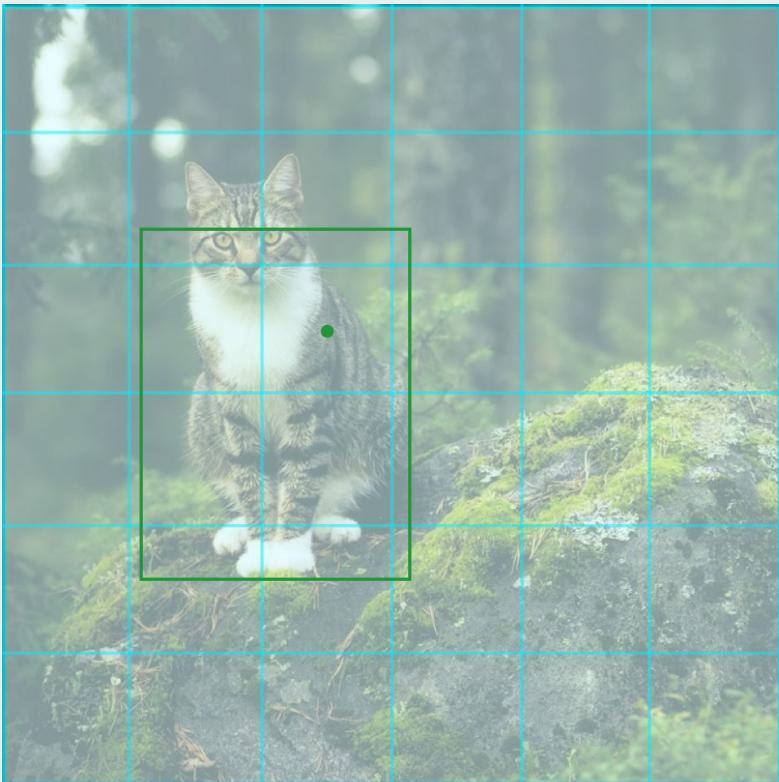
$$[x, y, w, h, c, P_1, \dots, P_n]$$

Вектор содержит информацию:

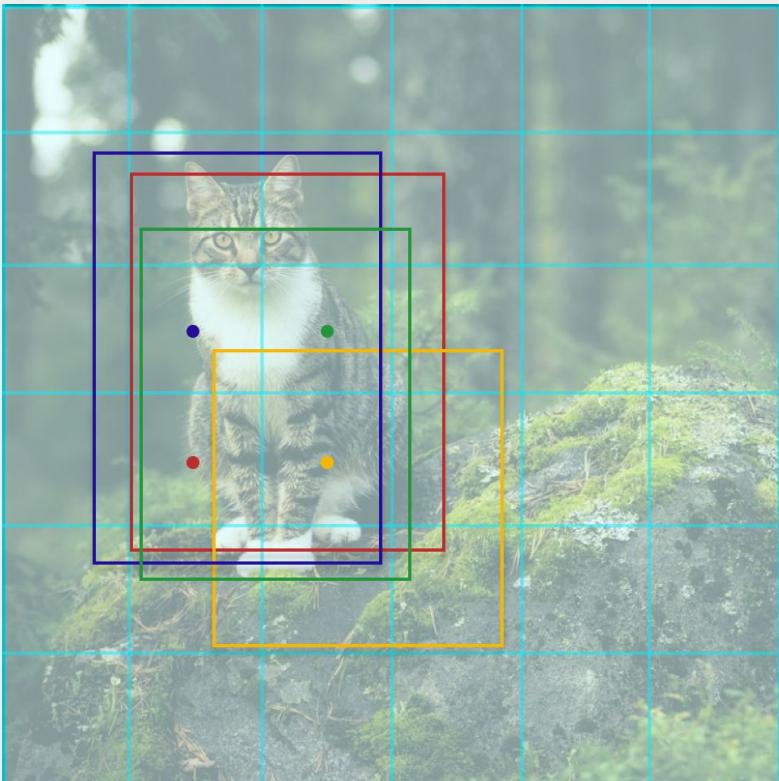
- О размерах предсказанного бокса,
- Confidence score или вероятность присутствия объекта в этом боксе,
- Вероятности принадлежности к каждому из n классов (число классов зависит от датасета).



Как детектор делает предсказания?



Как детектор делает предсказания?

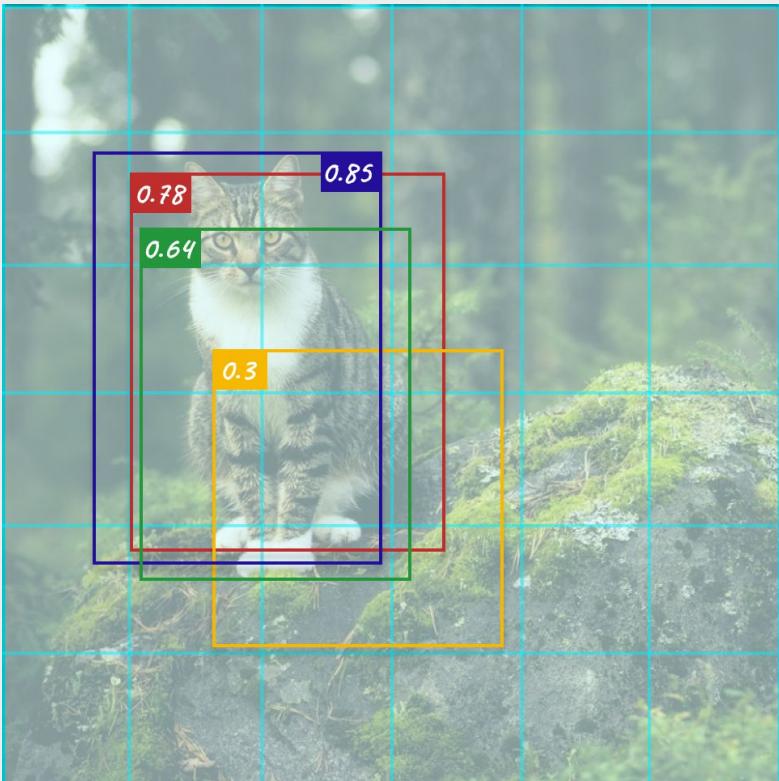


Рассмотрим 4 предсказания:

`Bbox = [x, y, w, h, c, "cat"]`



Как детектор делает предсказания?

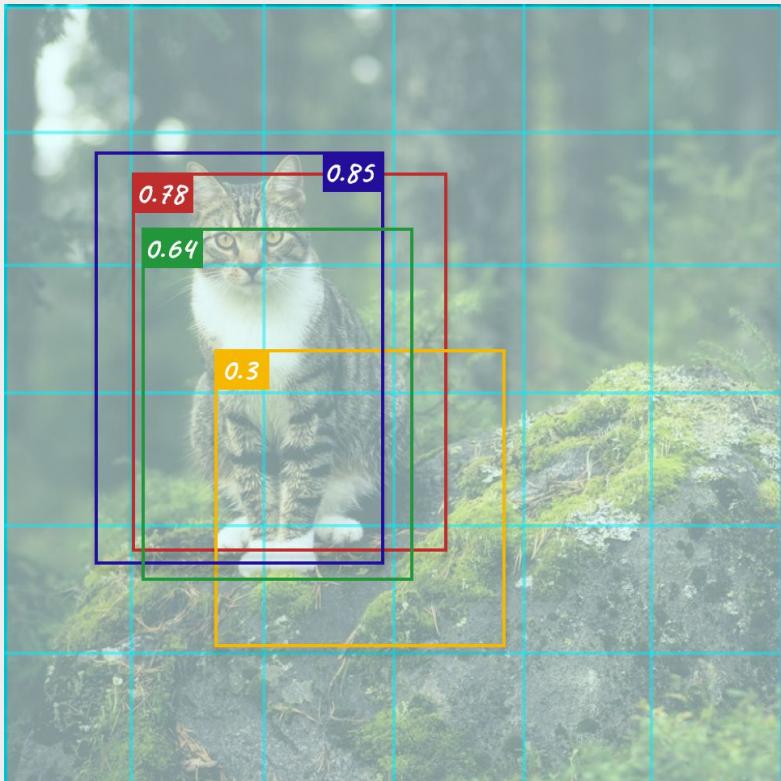


Допустим, предсказанные ббоксы имеют следующие confidence scores.

Как выбрать из них самый лучший?



Как детектор делает предсказания?



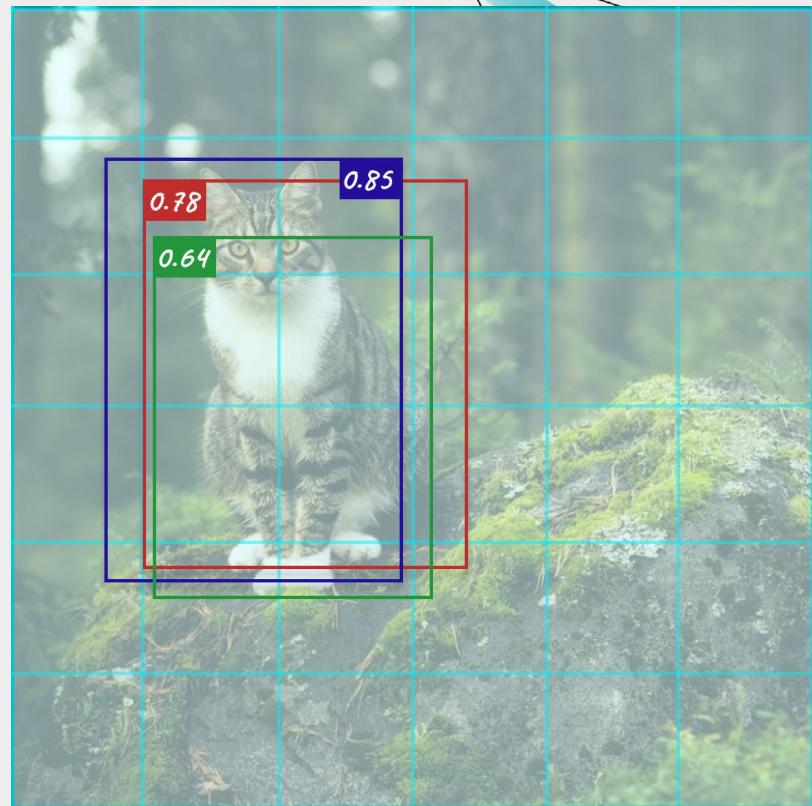
Допустим, предсказанные боксы имеют следующие confidence scores.

Как выбрать из них самый лучший?

Нам поможет алгоритм Non-Maximum Suppression (NMS)

Non-Maximum Suppression

1. Удаляем все bbox у которых
confidence score < conf_threshold
(Обычно 0.3 - 0.5),

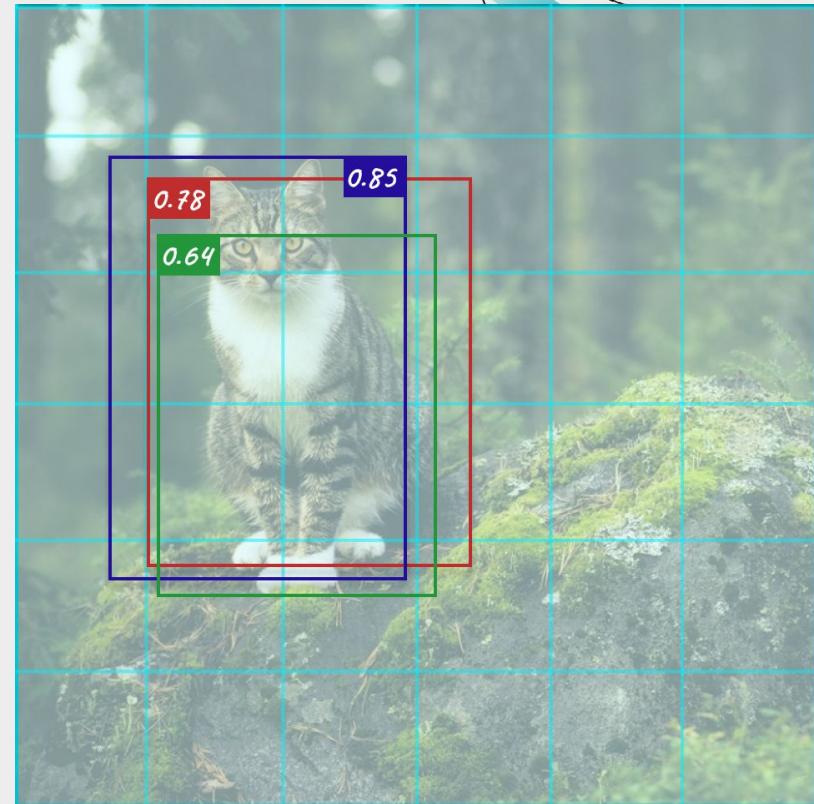


Non-Maximum Suppression

1. Удаляем все bbox у которых confidence score < conf_threshold (Обычно 0.3 - 0.5),
2. Сортируем оставшиеся bbox по убыванию confidence score,

Predicts:

```
Bbox = [x, y, w, h, 0.85, "cat"]
Bbox = [x, y, w, h, 0.78, "cat"]
Bbox = [x, y, w, h, 0.64, "cat"]
```



Non-Maximum Suppression

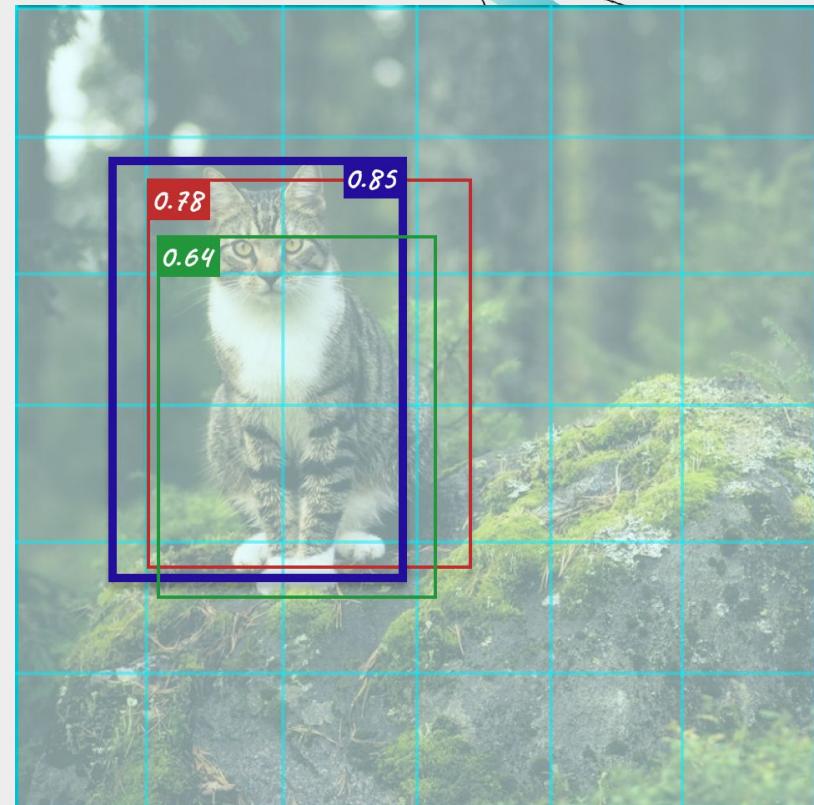
1. Удаляем все bbox у которых confidence score < conf_threshold (Обычно 0.3 - 0.5),
2. Сортируем оставшиеся bbox по убыванию confidence score,
3. Выбираем bbox с максимальным значением confidence score в качестве "опорного".

Predicts:

Bbox = [x, y, w, h, 0.85, "cat"]

Bbox = [x, y, w, h, 0.78, "cat"]

Bbox = [x, y, w, h, 0.64, "cat"]



Non-Maximum Suppression

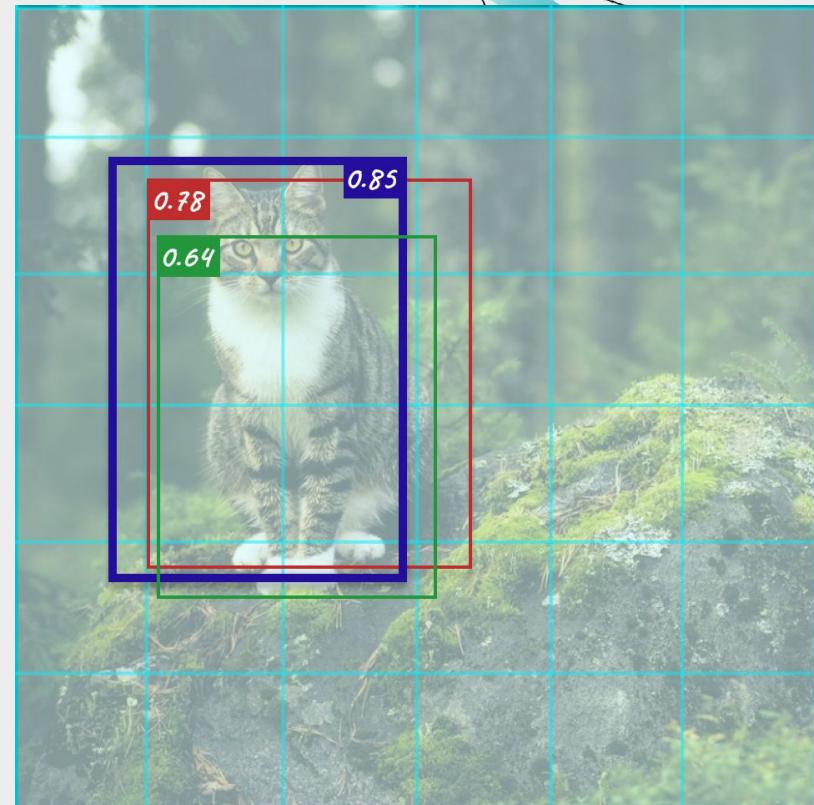
1. Удаляем все bbox у которых confidence score < conf_threshold (Обычно 0.3 - 0.5),
2. Сортируем оставшиеся bbox по убыванию confidence score,
3. Выбираем bbox с максимальным значением confidence score в качестве "опорного".
4. Считаем метрику IoU между bbox и всеми остальными bbox.

Predicts:

Bbox = [x, y, w, h, 0.85, "cat"]

Bbox = [x, y, w, h, 0.78, "cat"]

Bbox = [x, y, w, h, 0.64, "cat"]

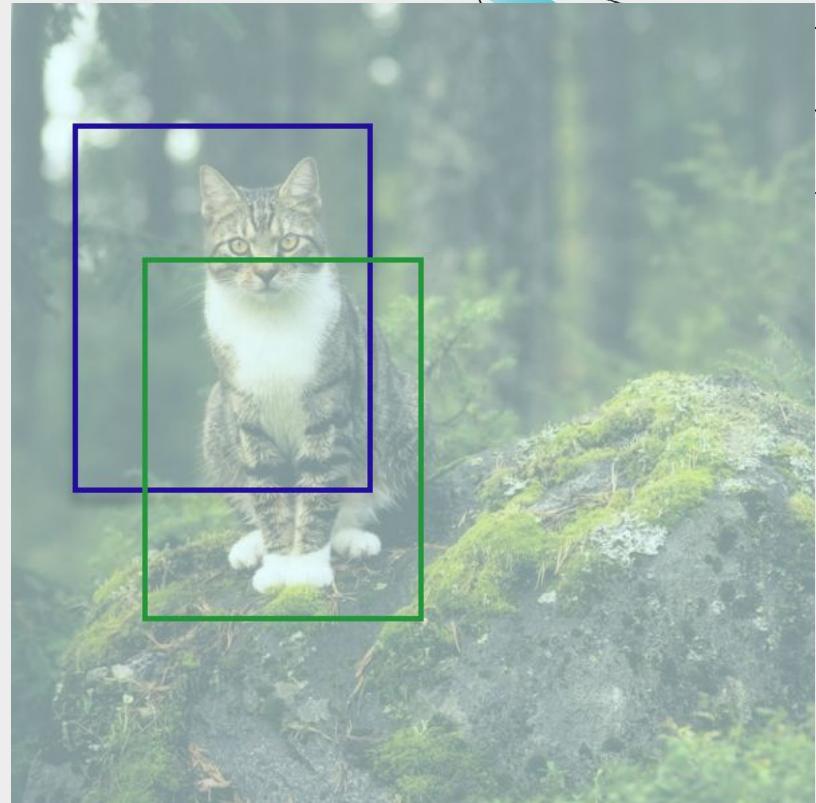


Intersection over Union (IoU)

Predicts:

Bbox = [x, y, w, h]

Bbox = [x, y, w, h]



Intersection over Union (IoU)

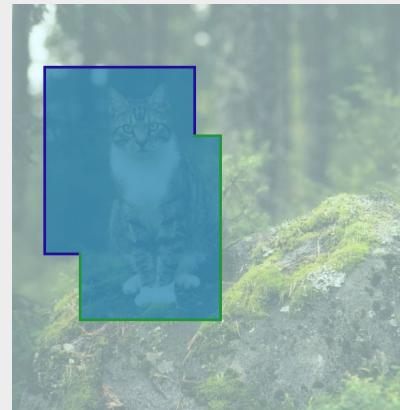
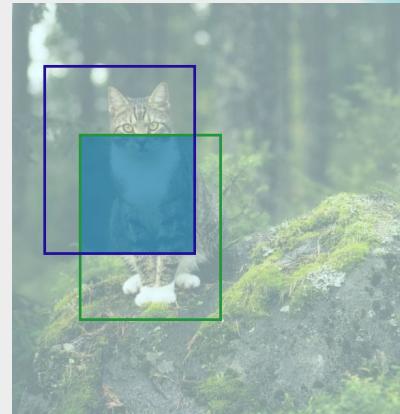


Predicts:

Bbox = [x, y, w, h]

Bbox = [x, y, w, h]

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

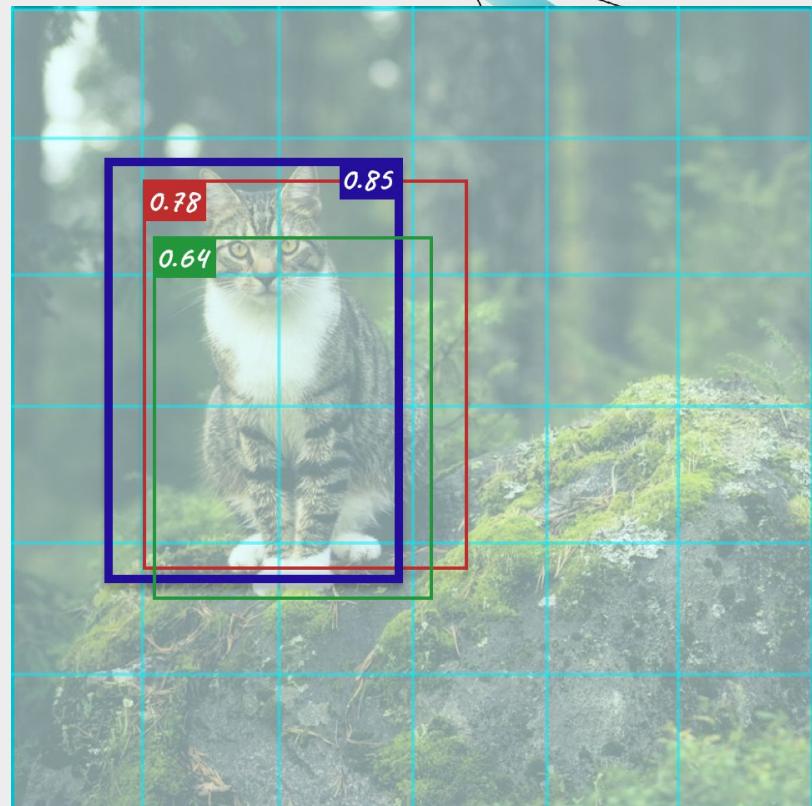


Non-Maximum Suppression

1. Удаляем все bbox у которых confidence score < conf_threshold (Обычно 0.3 - 0.5),
2. Сортируем оставшиеся bbox по убыванию confidence score,
3. Выбираем bbox с максимальным значением confidence score в качестве "опорного".
4. Считаем метрику IoU между bbox и всеми остальными bbox.

Predicts:

Bbox = [x, y, w, h, 0.85, "cat"]
Bbox = [x, y, w, h, 0.78, "cat"],
IoU = 0.7
Bbox = [x, y, w, h, 0.64, "cat"],
IoU = 0.63

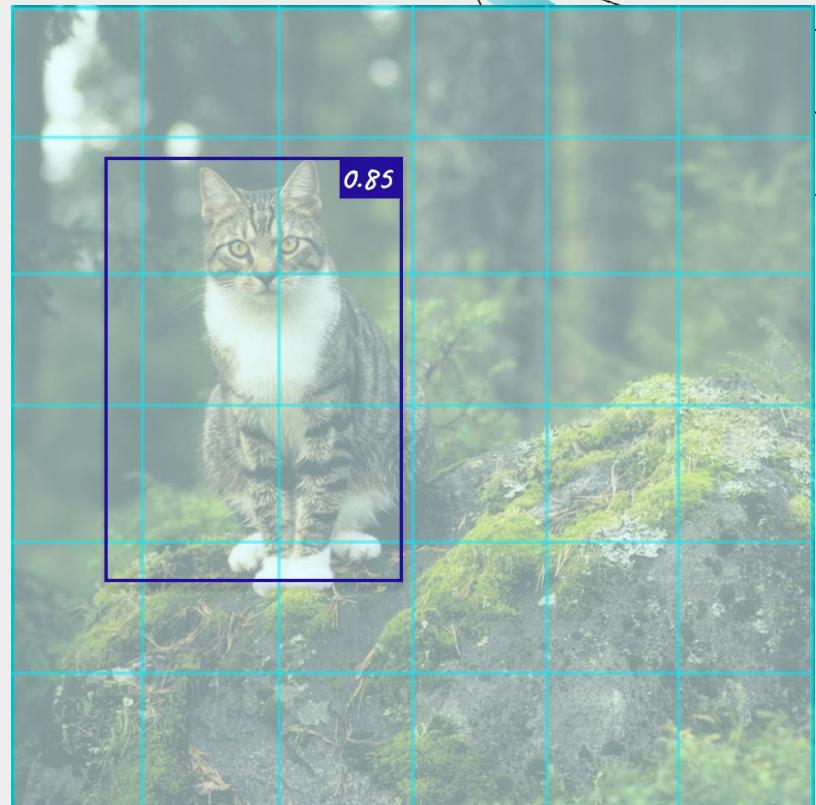


Non-Maximum Suppression

1. Удаляем все bbox у которых confidence score < conf_threshold (Обычно 0.3 - 0.5),
2. Сортируем оставшиеся bbox по убыванию confidence score,
3. Выбираем bbox с максимальным значением confidence score в качестве "опорного".
4. Считаем метрику IoU между bbox и всеми остальными bbox.
5. Удаляем все bbox, для которых IoU > iou_threshold (например 0.5)

Predicts:

Bbox = [x, y, w, h, 0.85, "cat"]



Non-Maximum Suppression



1. Удаляем все bbox у которых confidence score < conf_threshold (Обычно 0.3 - 0.5),
2. Сортируем оставшиеся bbox по убыванию confidence score,
3. Выбираем bbox с максимальным значением confidence score в качестве "опорного".
4. Считаем метрику IoU между bbox и всеми остальными bbox.
5. Удаляем все bbox, для которых IoU > iou_threshold (например 0.5)



Predicts:

Bbox = [x, y, w, h, 0.85, "cat"]

В данном случае, у нас остался только 1 bbox, он и будет итоговым предсказанием.

Non-Maximum Suppression



1. Удаляем все bbox у которых confidence score < conf_threshold (Обычно 0.3 - 0.5),
2. Сортируем оставшиеся bbox по убыванию confidence score,
3. Выбираем bbox с максимальным значением confidence score в качестве "опорного".
4. Считаем метрику IoU между bbox и всеми остальными bbox.
5. Удаляем все bbox, для которых IoU > iou_threshold (например 0.5)



Predicts:

Bbox = [x, y, w, h, 0.85, "cat"]

В данном случае, у нас остался только 1 bbox, он и будет итоговым предсказанием.

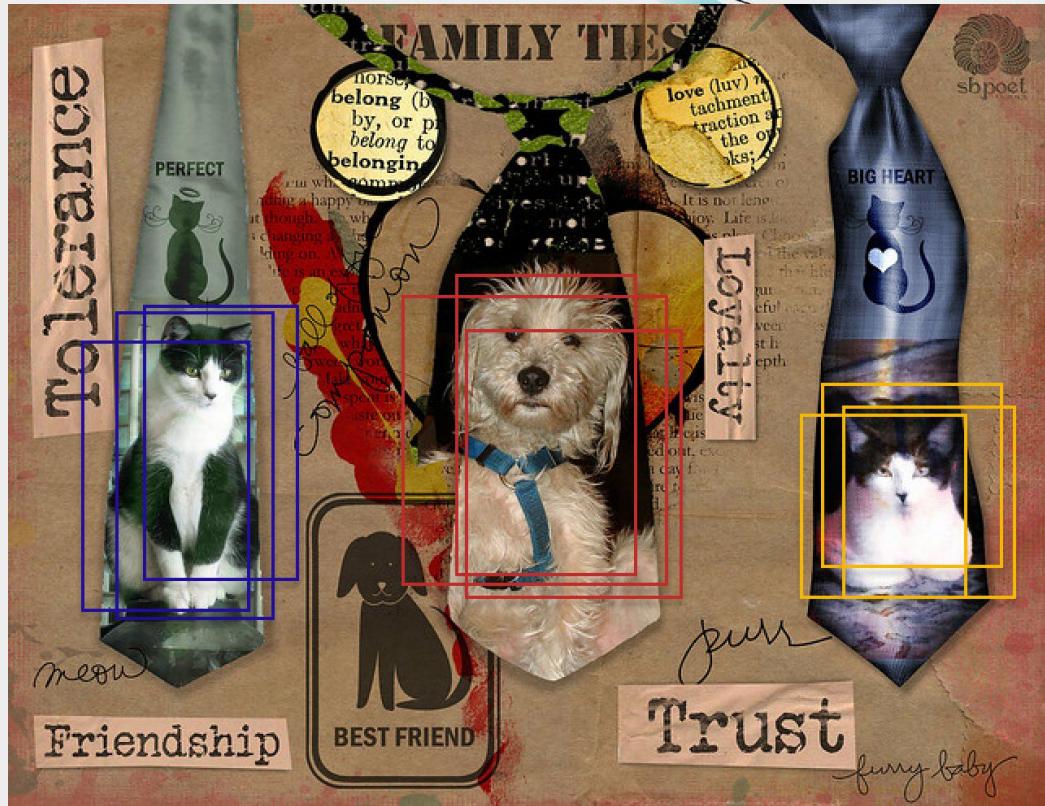
Но что делать, когда предсказаний больше? А если на картинке несколько разных объектов?

NMS. Пример 2.



NMS. Пример 2.

Визуализируем боксы, для которых
confidence score >
conf_threshold:



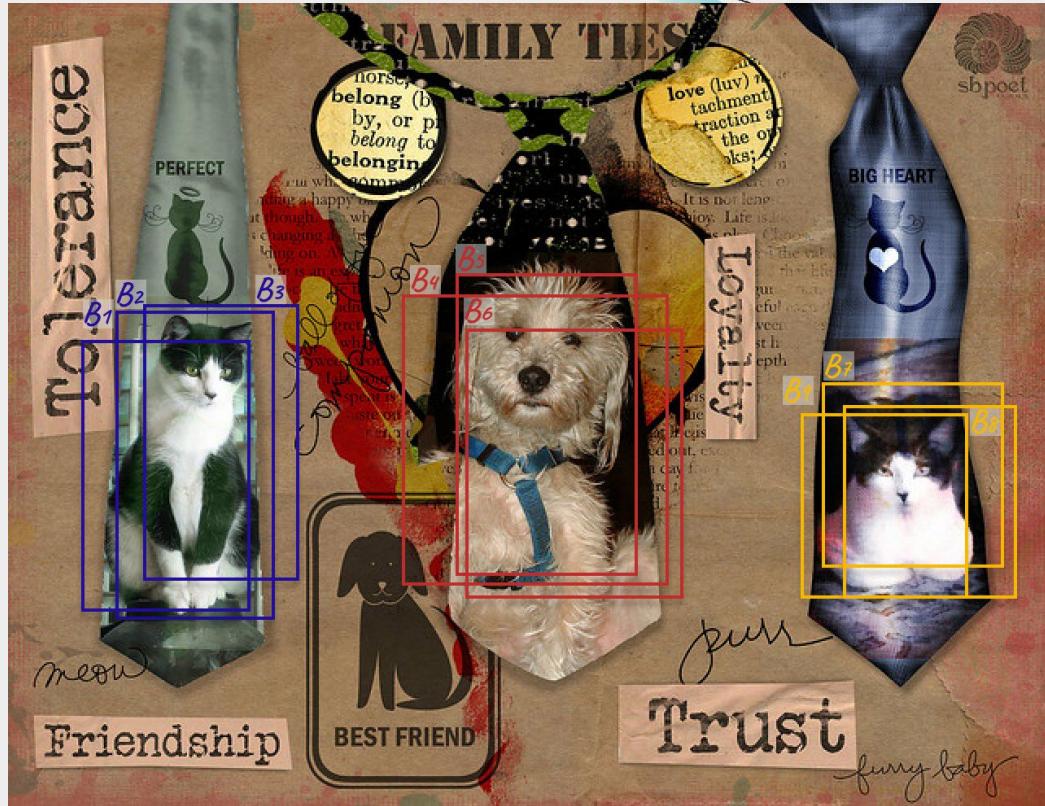
NMS. Пример 2.

В данном случае, будем рассматривать следующий набор ббоксов:

```
B1 = [coords, 0.7, "cat"]
B2 = [coords, 0.95, "cat"]
B3 = [coords, 0.8, "cat"]

B4 = [coords, 0.85, "dog"]
B5 = [coords, 0.6, "dog"]
B6 = [coords, 0.7, "dog"]

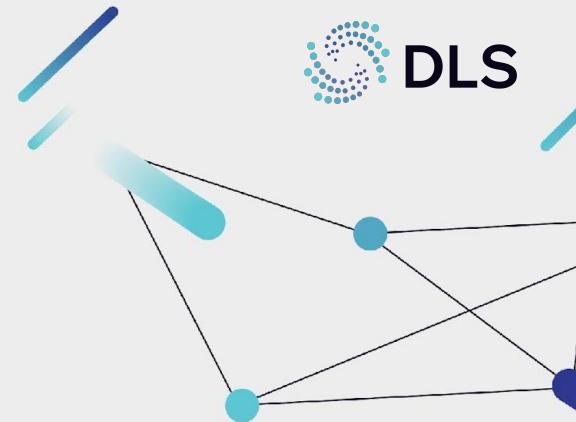
B7 = [coords, 0.8, "cat"]
B8 = [coords, 0.72, "cat"]
B9 = [coords, 0.79, "cat"]
```



NMS. Пример 2.

Дополним наш алгоритм:

1. Удаляем все bbox у которых
 $\text{confidence score} < \text{conf_threshold}$,



NMS. Пример 2.

Дополним наш алгоритм:

1. Удаляем все bbox у которых
confidence score < conf_threshold,
Для bbox с разными классами, независимо:

Bboxes for "cat":

```
B1 = [coords, 0.7, "cat"]
B2 = [coords, 0.95, "cat"]
B3 = [coords, 0.8, "cat"]
B7 = [coords, 0.8, "cat"]
B8 = [coords, 0.72, "cat"]
B9 = [coords, 0.79, "cat"]
```

Bboxes for "dog":

```
B4 = [coords, 0.85, "dog"]
B5 = [coords, 0.6, "dog"]
B6 = [coords, 0.7, "dog"]
```



NMS. Пример 2.

Дополним наш алгоритм:

1. Удаляем все bbox у которых
confidence score < conf_threshold,
для bbox с разными классами, независимо:
2. Сортируем оставшиеся bbox по
убыванию confidence score,

Bboxes for "cat":

```
B2 = [coords, 0.95, "cat"]
B3 = [coords, 0.8, "cat"]
B7 = [coords, 0.8, "cat"]
B9 = [coords, 0.79, "cat"]
B8 = [coords, 0.72, "cat"]
B1 = [coords, 0.7, "cat"]
```



NMS. Пример 2.

Дополним наш алгоритм:

1. Удаляем все bbox у которых confidence score < conf_threshold,

Для bbox с разными классами, независимо:

2. Сортируем оставшиеся bbox по убыванию confidence score,
3. Выбираем bbox с максимальным значением confidence score в качестве "опорного",

Bboxes for "cat":

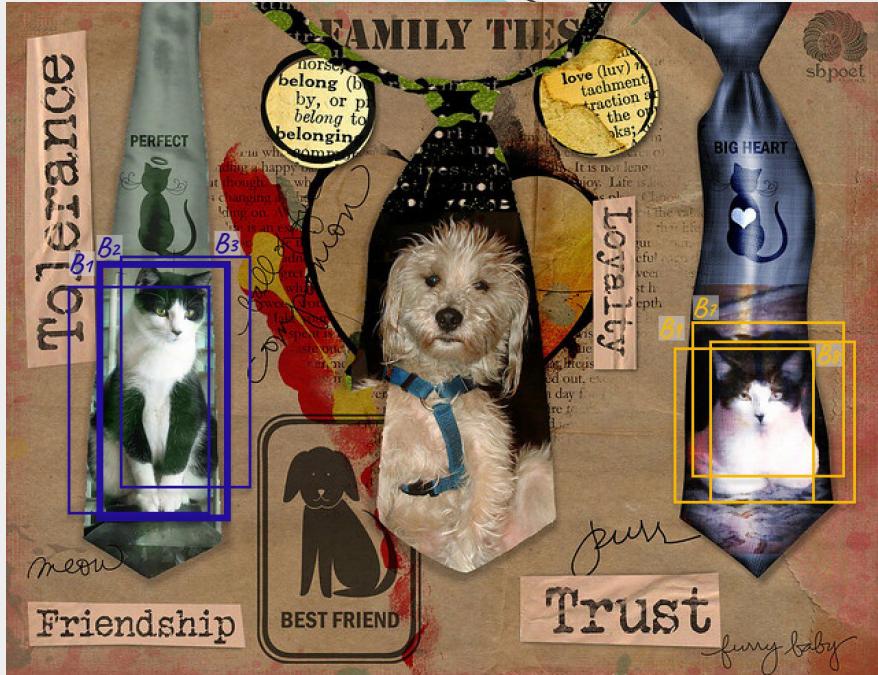
```
B2 = [coords, 0.95, "cat"]  
B3 = [coords, 0.8, "cat"]  
B7 = [coords, 0.8, "cat"]  
B9 = [coords, 0.79, "cat"]  
B8 = [coords, 0.72, "cat"]  
B1 = [coords, 0.7, "cat"]
```



NMS. Пример 2.

Дополним наш алгоритм:

1. Удаляем все bbox у которых confidence score < conf_threshold,
- Для bbox с разными классами, независимо:**
2. Сортируем оставшиеся bbox по убыванию confidence score,
 3. Выбираем bbox с максимальным значением confidence score в качестве "опорного",



NMS. Пример 2.

Дополним наш алгоритм:

1. Удаляем все bbox у которых confidence score < conf_threshold,
2. Для bbox с разными классами, независимо:
Сортируем оставшиеся bbox по убыванию confidence score,
3. Выбираем bbox с максимальным значением confidence score в качестве "опорного",
4. Считаем метрику IoU между опорным bbox и всеми остальными bbox.

Bboxes for "cat":

B2 = [coords, 0.95, "cat"]

B3 = [coords, 0.8, "cat"],
 $\text{IoU}(B2, B3) = 0.62$

B7 = [coords, 0.8, "cat"],
 $\text{IoU}(B2, B7) = 0$

B9 = [coords, 0.79, "cat"],
 $\text{IoU}(B2, B9) = 0$

B8 = [coords, 0.72, "cat"],
 $\text{IoU}(B2, B8) = 0$

B1 = [coords, 0.7, "cat"],
 $\text{IoU}(B2, B1) = 0.71$



NMS. Пример 2.

Дополним наш алгоритм:

1. Удаляем все bbox у которых confidence score < conf_threshold,
- Для bbox с разными классами, независимо:**
2. Сортируем оставшиеся bbox по убыванию confidence score,
3. Выбираем bbox с максимальным значением confidence score в качестве "опорного",
4. Считаем метрику IoU между опорным bbox и всеми остальными bbox.
5. Удаляем все bbox, для которых IoU > iou_threshold (например 0.5). Отмечаем, что опорный элемент - наше предсказание.
6. Повторяем пункты (2-5), пока список ббоксов не пустой.

Bboxes for "cat":

B7 = [coords, 0.8, "cat"]
B9 = [coords, 0.79, "cat"]
B8 = [coords, 0.72, "cat"]

Предсказанные ббоксы:

B2



NMS. Пример 2.

Дополним наш алгоритм:

1. Удаляем все bbox у которых confidence score < conf_threshold,
2. Для bbox с разными классами, независимо:
2. Сортируем оставшиеся bbox по убыванию confidence score,
3. Выбираем bbox с максимальным значением confidence score в качестве "опорного",
4. Считаем метрику IoU между опорным bbox и всеми остальными bbox.
5. Удаляем все bbox, для которых IoU > iou_threshold (например 0.5). Отмечаем, что опорный элемент - наше предсказание.
6. Повторяем пункты (2-5), пока список ббоксов не пустой.

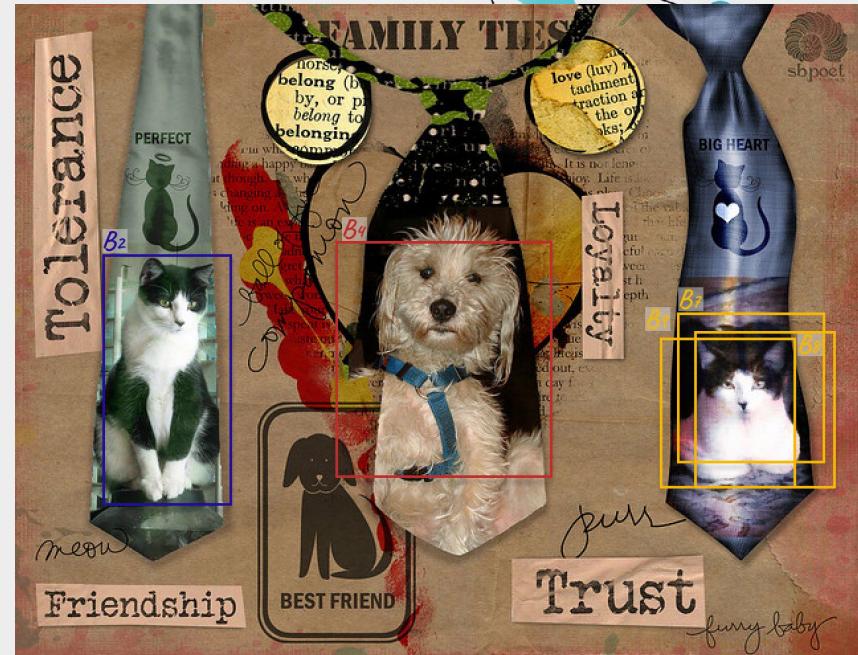


Предсказанные ббоксы:
B2

NMS. Пример 2.

Дополним наш алгоритм:

1. Удаляем все bbox у которых confidence score < conf_threshold,
2. Для bbox с разными классами, независимо:
2. Сортируем оставшиеся bbox по убыванию confidence score,
3. Выбираем bbox с максимальным значением confidence score в качестве "опорного",
4. Считаем метрику IoU между опорным bbox и всеми остальными bbox.
5. Удаляем все bbox, для которых IoU > iou_threshold (например 0.5). Отмечаем, что опорный элемент - наше предсказание.
6. Повторяем пункты (2-5), пока список ббоксов не пустой.



Предсказанные ббоксы:
B2, B4

NMS. Пример 2.

Предсказанные ббоксы:
B2, B4, B7



NMS. Итоги.

Плюсы:

1. Простой алгоритм, позволяющий оставить только релевантные ббоксы,
2. Хорошо выполняет задачу выбора релевантных ббоксов в простых случаях,
3. Работает сравнительно быстро.
4. Всего 1 параметр - IoU_threshold

* conf_threshold это не параметр NMS.

Минусы:

1. Плохо работает в сложных случаях:

Ex: Если на картинке рядом стоят 2 объекта одного класса, NMS может удалить один из ббоксов, посчитав его дубликатом.

2. Сложности с multi-class.

Ex: Может остаться 2 ббокса для одного объекта, если у них разные классы.

3. IoU_threshold может требовать точного подбора.



NMS. Вариации.

1. Fast NMS

Заменяем все циклы векторизованными операциями, но можем потерять в точности в некоторых случаях.

2. Soft-NMS

Не удаляем перекрывающиеся боксы, а понижаем их скор. Помогает в случаях когда объекты перекрываются.

3. Adaptive NMS

Адаптивно подбирает порог IoU в зависимости от плотности объектов.

4. Weighted NMS

Положение предсказанного бокса считается как взвешенное среднее координат пересекающихся боксов.

5. DIoU-NMS

Используем другую оценку пересечения боксов вместо IoU.



Метрики в задаче детекции



Метрики детекции

1. Метрики качества
 - Precision, Recall, F1-score
 - Mean Average Precision (mAP)
 - Average Recall (AR)
2. Время инференса
 - Inference time
 - FPS
 - Latency
 - Throughput



Confusion matrix

Прежде чем считать метрики, нужно понять как получить confusion matrix.

А именно нас интересуют:

- True Positive (TP)
- False Positive (FP)
- False Negative (FN)

Confusion matrix

Прежде чем считать метрики, нужно понять как получить confusion matrix.

А именно нас интересуют:

- **TP** - Для бокса из разметки (Истинный бокс / Ground Truth / GT) есть предсказанный бокс того же класса,
- **FP** - Для предсказанного бокса не нашлось GT бокса,
- **FN** - Для GT бокса не нашлось предсказанного бокса.

Confusion matrix

Прежде чем считать метрики, нужно понять как получить confusion matrix.

А именно нас интересуют:

- **TP** - Для ббокса из разметки (Истинный ббокс / Ground Truth / GT) есть предсказанный ббокс того же класса,
- **FP** - Для предсказанного ббокса не нашлось GT ббокса,
- **FN** - Для GT ббокса не нашлось предсказанного ббокса.

Разберемся как их считать.

Важно: Мы оцениваем предсказания модели после применения NMS, чтобы отбросить дублирующиеся ббоксы и не занижать качество модели.

Чтобы метрики были точнее, `conf_threshold` можно взять супер маленький (около 0.01).

Confusion matrix

Алгоритм расчета похож на NMS и также делается **независимо для каждого класса**:



Confusion matrix

Алгоритм расчета похож на NMS и также делается **независимо для каждого класса**:

1. Для класса С выделяем GT_C и P_C , где:

- GT_C – множество GT боксов класса C,
- P_C – множество предсказанных боксов для которых модель предсказала класс C.



Confusion matrix

Алгоритм расчета похож на NMS и также делается **независимо для каждого класса**:

1. Для класса C выделяем GT_C и P_C , где:
 - GT_C – множество GT боксов класса C ,
 - P_C – множество предсказанных боксов для которых модель предсказала класс C .
2. Сортируем P_C по убыванию confidence score.

Confusion matrix

Алгоритм расчета похож на NMS и также делается **независимо для каждого класса**:

1. Для класса C выделяем GT_C и P_C , где:
 - GT_C – множество GT боксов класса C ,
 - P_C – множество предсказанных боксов для которых модель предсказала класс C .
2. Сортируем P_C по убыванию confidence score.
3. Отмечаем все GT_C как “не обнаруженные” и начинаем итерироваться по P_C :
 - 3.1. Вычисляем IoU между предсказанием $P_{C,i}$ и всеми g (боксы из GT_C):
 - 3.1.1. Если $\forall g \text{ IoU}(P_{C,i}, g) < \text{IoU_threshold}$, то $P_{C,i}$ - FP,
 - 3.1.2. Иначе, выбираем g с максимальным значением $\text{IoU}(P_{C,i}, g)$. Если g еще “не обнаружен” помечаем его как “обнаруженный” и считаем $P_{C,i}$ - TP, иначе, $P_{C,i}$ является дублем и помечается FP.



Confusion matrix

Алгоритм расчета похож на NMS и также делается **независимо для каждого класса**:

1. Для класса **C** выделяем **GT_c** и **P_c**, где:
 - **GT_c** – множество GT ббоксов класса **C**,
 - **P_c** – множество предсказанных ббоксов для которых модель предсказала **класс C**.
2. Сортируем **P_c** по убыванию confidence score.
3. Отмечаем все **GT_c** как “не обнаруженные” и начинаем итерироваться по **P_c**:
 - 3.1. Вычисляем IoU между предсказанием **P_{c_i}** и всеми **g** (ббоксы из **GT_c**):
 - 3.1.1. Если **Все** $\text{IoU}(\text{Pc}_i, g) < \text{IoU_threshold}$, то **Pc_i** - **FP**,
 - 3.1.2. Иначе, выбираем **g** с максимальным значением $\text{IoU}(\text{Pc}_i, g)$. Если **g** еще “не обнаружен” помечаем его как “обнаруженный” и считаем **Pc_i** - **TP**, иначе, **Pc_i** является дублем и помечается **FP**.
4. После окончания цикла по **P_c**, **GT_c**, помеченные как “не обнаруженные” считаются **FN**, тк для них не нашлось ни одного ббокса.

Метрики качества. Пример.



Метрики качества. Пример.



Допустим, модель предсказала следующие боксы:

- 1 = [coords, 0.92, "cat"]
- 2 = [coords, 0.34, "dog"]
- 3 = [coords, 0.74, "cat"]
- 4 = [coords, 0.82, "cat"]
- 5 = [coords, 0.65, "cat"]

Метрики качества. Пример.



Следуя алгоритму, посмотрим какие значения TP, FP, FN будут для класса "cat":

Nº pred	Confidence	IoU GT	Вывод
1	0.92	0.85 (A)	TP
4	0.82	0.66 (C)	TP
3	0.74	0	FP
5	0.65	0.52 (C)	FP

$$TP = 2, \quad FP = 2, \quad FN = 0$$

Метрики качества. Пример.



Следуя алгоритму, посмотрим какие значения TP, FP, FN будут для класса "dog":

№ pred	Confidence	IoU GT	Выход
2	0.34	0.04 (B)	FP

$$TP = 0, FP = 1, FN = 1$$

Precision, Recall, F1-score

Также как в классификации, для полученных предсказаний можно посчитать любые классификационные метрики:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall} = 2 * \frac{TP}{TP + FN + FP}$$

Precision, Recall, F1-score

Также как в классификации, для полученных предсказаний можно посчитать любые классификационные метрики:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall} = 2 * \frac{TP}{TP + FN + FP}$$

Несколько уточнений:

1. При решении задачи `multiclass` детекции, метрики усредняются по классам,
2. В отличии от классификации, в детекции необходимо внимательно подбирать два порога: `conf_threshold` и `IoU_threshold`.

Precision-Recall curve

Варьируя `conf_threshold` можно построить, знакомую из задачи классификации, precision-recall кривую.

Она показывает соотношение точности и полноты в зависимости от изменения `confidence score`.



Precision-Recall curve

Допустим есть датасет из 3х картинок, на которых хотим измерить качество:

Изображение А: 2 объекта (A1, A2)

Изображение В: 1 объект (B1)

Изображение С: 3 объекта (C1, C2, C3)

Используем IoU_threshold = 0.5

* precision-recall curve считается для каждого класса независимо, поэтому считаем что на всех картинках объекты одного класса.

Nº pred	IMG	Confidence	IoU GT	Вывод
1	A	0.92	0.85 (A1)	TP
2	B	0.9	0.66 (B1)	TP
3	C	0.87	0.3 (C1)	FP
4	A	0.82	0.55 (A2)	TP
5	C	0.73	0.63 (C1)	TP
6	C	0.71	0.62 (C2)	TP
7	C	0.68	0.58 (C1)	FP
8	C	0.63	0.52 (C3)	TP

Precision-Recall curve

Считаем накопленные суммы ТР, FP по порогу с шагом 0.5

conf_th	Cumulative TP	Cumulative FP
0.95	0	0
0.9	2	0
0.85	2	1
0.8	3	1
0.75	4	1
0.7	5	1
0.65	5	2
0.6	6	2

Nº pred	IMG	Confidence	IoU GT	Вывод
1	A	0.92	0.85 (A1)	TP
2	B	0.9	0.66 (B1)	TP
3	C	0.87	0.3 (C1)	FP
4	A	0.82	0.55 (A2)	TP
5	C	0.73	0.63 (C1)	TP
6	C	0.71	0.62 (C2)	TP
7	C	0.68	0.58 (C1)	FP
8	C	0.63	0.52 (C3)	TP

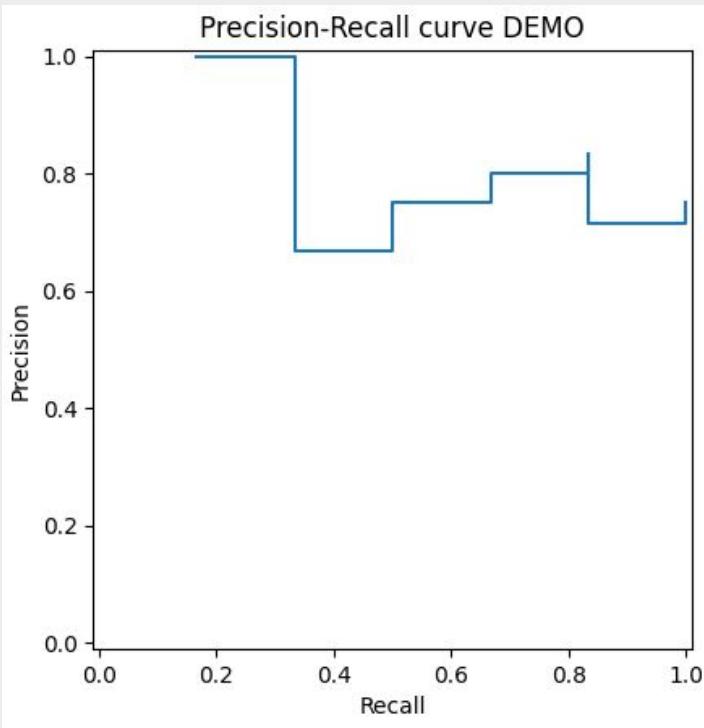
Precision-Recall curve

Вычисляем значения Precision и Recall на каждом шаге

conf_th	Cumulative TP	Cumulative FP	Precision	Recall
0.95	1	0	1.0	0.167
0.9	2	0	1.0	0.333
0.85	2	1	0.667	0.333
0.8	3	1	0.75	0.5
0.75	4	1	0.8	0.667
0.7	5	1	0.833	0.833
0.65	5	2	0.714	0.833
0.6	6	2	0.75	1

Precision-Recall curve

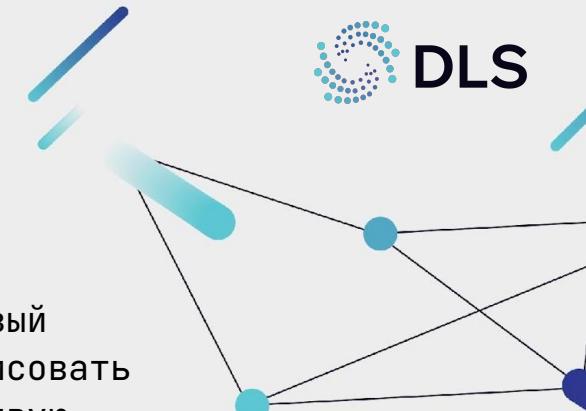
Рисуем график:



Чтобы сгладить итоговый результат, можно нарисовать интерполированную кривую precision-recall:

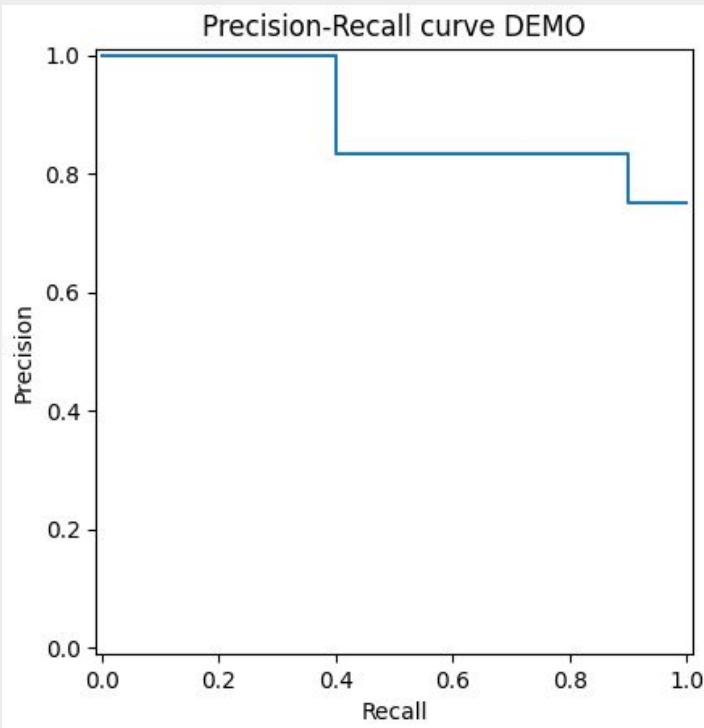
Для каждого порога recall (R) выбираем максимальное значение precision, среди точек, у которых $\text{recall} \geq R$.

Например в PASCAL VOC 2007 применяется 11-точечная интерполяция,
 $r \in \{0.0, 0.1, 0.2, \dots, 1.0\}$.



Precision-Recall curve

Рисуем график:



Чтобы сгладить итоговый результат, можно нарисовать интерполированную кривую precision-recall:

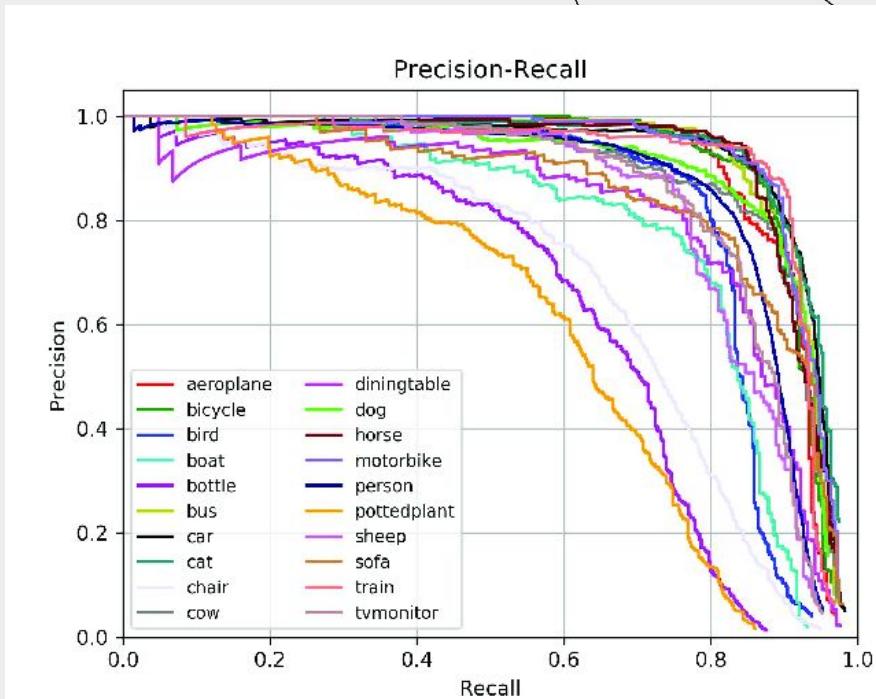
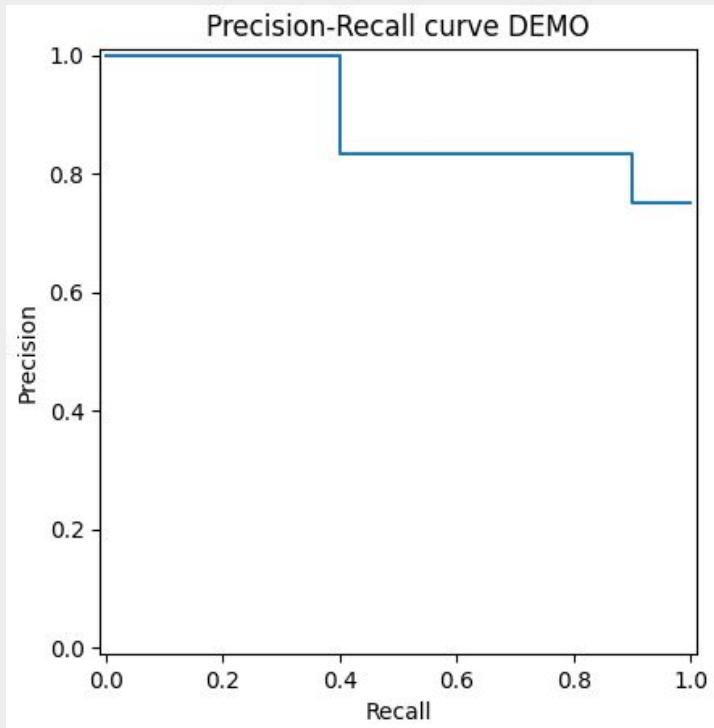
Для каждого порога recall (R) выбираем максимальное значение precision, среди точек, у которых $\text{recall} \geq R$.

Например в PASCAL VOC 2007 применяется 11-точечная интерполяция,
 $r \in \{0.0, 0.1, 0.2, \dots, 1.0\}$.



Precision-Recall curve

А так выглядит PR curve на COCO:



Average Precision

Average Precision (AP) – это площадь под кривой Precision-Recall.

Чтобы не считать интеграл, обычно Precision-recall кривую интерполируют, чтобы она стала невозрастающей:

$$P_{interp}(R_i) = \max_{j \geq i} P(R_j)$$

То есть текущее значение Precision (P) заменяется на максимум среди всех последующих значений метрики.
В таком случае используется формула:

$$AP = \sum_n (R_n - R_{n-1}) P_n$$

Mean Average Precision

Mean Average Precision (mAP) – это усредненное значение AP по всем классам.

$$mAP = \frac{1}{N} \sum_{i=1}^N AP_i$$

где, N - число классов.



Mean Average Precision. Вариации.

В зависимости от выбранного значения IoU при построении Precision-Recall кривой, выделяют следующие основные вариации AP:

Average Precision (AP):

AP	% AP at IoU=.50:.05:.95 (primary challenge metric)
AP ^{IoU=.50}	% AP at IoU=.50 (PASCAL VOC metric)
AP ^{IoU=.75}	% AP at IoU=.75 (strict metric)

AP Across Scales:

AP ^{small}	% AP for small objects: area < 32 ²
AP ^{medium}	% AP for medium objects: 32 ² < area < 96 ²
AP ^{large}	% AP for large objects: area > 96 ²

Это AP метрики для валидации моделей для датасета COCO.

Average Recall

Average Recall – метрика, показывающая насколько модель способна обнаружить ВСЕ объекты, присутствующие на изображении.

Она дополняет AP с точки зрения полноты предсказаний и может использоваться в задачах, где пропуск даже одного объекта может иметь критические последствия (например, в медицине).



Average Recall

Average Recall – метрика, показывающая насколько модель способна обнаружить ВСЕ объекты, присутствующие на изображении.

Она дополняет AP с точки зрения полноты предсказаний и может использоваться в задачах, где пропуск даже одного объекта может иметь критические последствия (например, в медицине).

- Чтобы лучше оценить полноту модели, вводятся ограничения на максимальное количество предсказаний для одного изображения:
 - AR@1 – Для каждого изображения берется только 1 предсказание (bbox with max conf),
 - AR@10 – На одно изображение максимум 10 предсказаний,
 - AR@100 – На одно изображение максимум 100 предсказаний.



Average Recall

Average Recall – метрика, показывающая насколько модель способна обнаружить ВСЕ объекты, присутствующие на изображении.

Она дополняет AP с точки зрения полноты предсказаний и может использоваться в задачах, где пропуск даже одного объекта может иметь критические последствия (например, в медицине).

- Чтобы лучше оценить полноту модели, вводятся ограничения на максимальное количество предсказаний для одного изображения (AR@1, AR@10, AR@100).
- Обычно, метрика считается для диапазона IoU порогов. В COCO диапазон от 0.5 до 0.95 с шагом 0.05.

Average Recall

Алгоритм расчета:

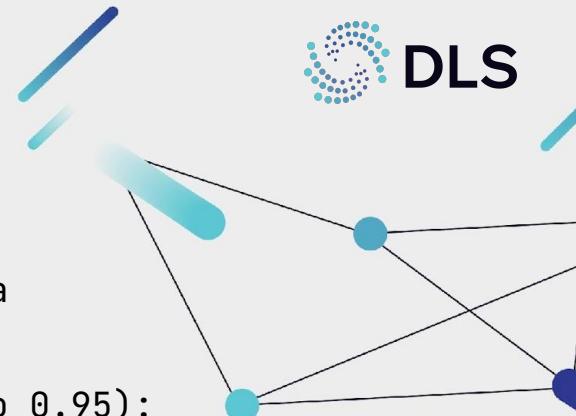
1. Заранее выбираем максимальное количество предсказаний на изображении (k),



Average Recall

Алгоритм расчета:

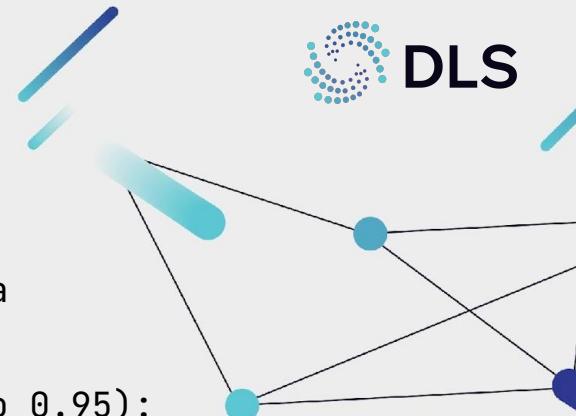
1. Заранее выбираем максимальное количество предсказаний на изображении (k),
2. Для каждого выбранного порога IoU T из диапазона (0.5 до 0.95):



Average Recall

Алгоритм расчета:

1. Заранее выбираем максимальное количество предсказаний на изображении (k),
2. Для каждого выбранного порога IoU T из диапазона (0.5 до 0.95):
 - Для каждого класса C :
 - Для каждого изображения где есть хотя бы 1 объект класса C :



Average Recall

Алгоритм расчета:

1. Заранее выбираем максимальное количество предсказаний на изображении (k),
2. Для каждого выбранного порога IoU T из диапазона (0.5 до 0.95):
 - Для каждого класса C :
 - Для каждого изображения где есть хотя бы 1 объект класса C :
 - 2.1. Фильтруем предсказания для класса C ,
 - 2.2. Сортируем по confidence и отбираем первые k ,
 - 2.3. Жадно сопоставляем предсказания и GT по критерию $\text{IoU} \geq T$,
 - 2.4. Считаем Recall для данного изображения и класса C ,



Average Recall

Алгоритм расчета:

1. Заранее выбираем максимальное количество предсказаний на изображении (k),
2. Для каждого выбранного порога IoU T из диапазона (0.5 до 0.95):
 - Для каждого класса C :
 - Для каждого изображения где есть хотя бы 1 объект класса C :
 - 2.1. Фильтруем предсказания для класса C ,
 - 2.2. Сортируем по confidence и отбираем первые k ,
 - 2.3. Жадно сопоставляем предсказания и GT по критерию $\text{IoU} \geq T$,
 - 2.4. Считаем Recall для данного изображения и класса C ,
 - Усредняем значения recall по всем изображениям для класса C и порога T ,
 - Усредняем полученные значения по классам, чтобы получить $AR@k$ при пороге T ,
 - 3. Усредняем $AR@k$ по всем порогам T , чтобы получить итоговое значение $AR@k$.

Average Recall. Вариации.

В случае Average Recall, в COCO используются следующие метрики:

Average Recall (AR):

$AR^{max=1}$

% AR given 1 detection per image

$AR^{max=10}$

% AR given 10 detections per image

$AR^{max=100}$

% AR given 100 detections per image

AR Across Scales:

AR^{small}

% AR for small objects: area < 32^2

AR^{medium}

% AR for medium objects: $32^2 < \text{area} < 96^2$

AR^{large}

% AR for large objects: area > 96^2

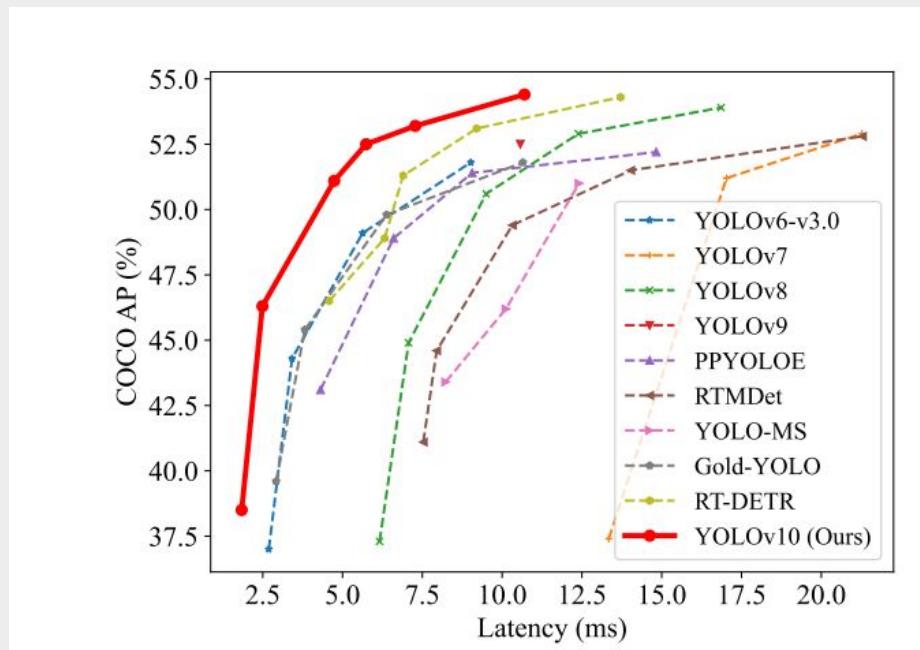
Метрики детекции.

1. Качество предсказания
 - a. Precision, Recall, F1-score
 - b. Mean Average Precision (mAP)
 - c. Average Recall (AR)
2. Время инференса
 - a. Inference time
 - b. FPS
 - c. Latency (Задержка обработки)
 - d. Throughput (Количество обработанных данных за секунду)



Runtime метрики

Обычно используются для наглядного представления зависимости качества от скорости.



Например вот график из статьи про YOLOv10. Это детектор, которому важно работать быстро. Поэтому авторы показывают как изменяется качество модели, в зависимости от скорости работы.

Inference Time vs Latency

Inference Time – время, которое требуется модели для выполнения прямого прохода (forward pass) по входным данным.

Latency – общее время задержки, при обработке одного батча данных.

Она включает в себя:

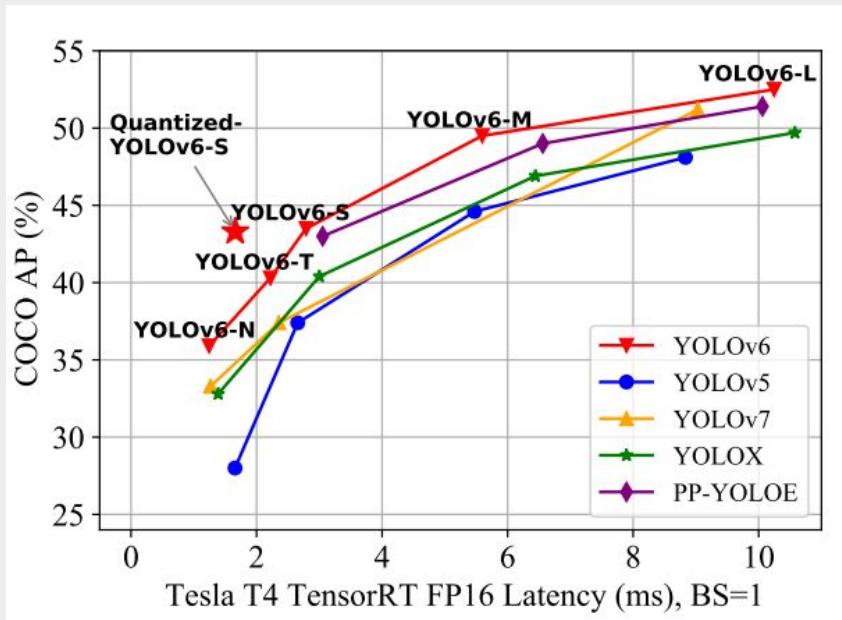
- Время на препроцессинг данных,
- Inference time,
- Передача данных (Например CPU → GPU и назад),
- Постпроцессинг данных (Например NMS).

Для измерения Latency используют батчи размером 1.

Также графиком указывают могут указывать тип данных, железо и фреймворк. Потому что все эти параметры могут существенно влиять на время задержки.



Latency



Например Latency в статье YOLOv6.

FPS vs Throughput

FPS (Frames per second) – количество изображений, которые способна обработать в секунду. В качестве метрики может использоваться в задачах видеоанализа и в real-time задачах.

Если система обрабатывает кадры последовательно, то FPS приблизительно равен $1 / \text{Latency}$.

FPS vs Throughput



FPS (Frames per second) – количество изображений, которые способна обработать в секунду. В качестве метрики может использоваться в задачах видеоанализа и в real-time задачах.

Если система обрабатывает кадры последовательно, то FPS приблизительно равен $1 / \text{Latency}$.

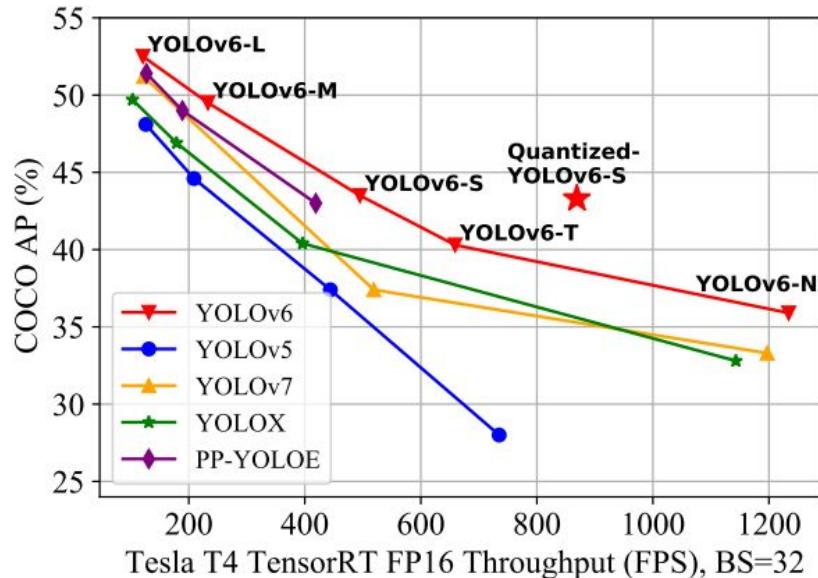
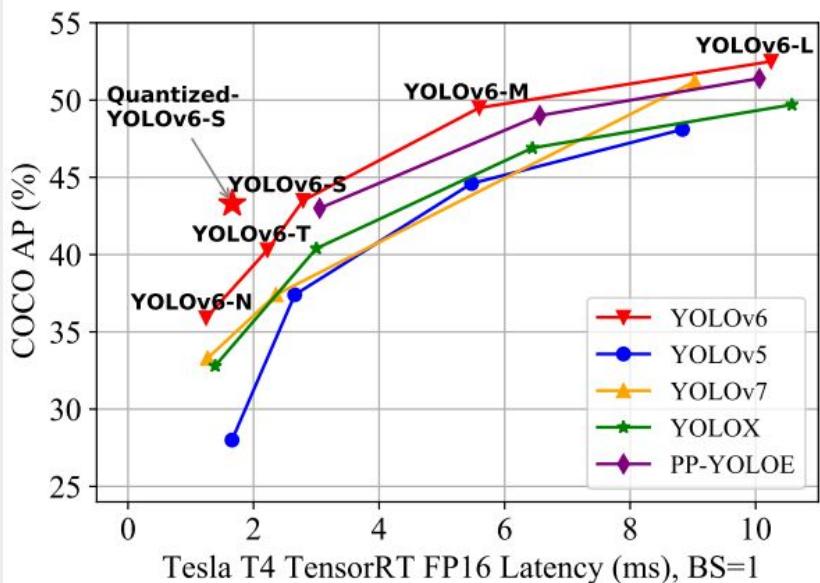
Throughput (Пропускная способность) – Общее количество данных за единицу времени. Можно рассчитать по формуле:

$$(\text{кол-во изображений}) / (\text{общее время работы (сек)})$$

Может считаться как для последовательно обрабатываемых батчей, так и в параллельном режиме. Отражает производительность системы. При удачном распараллеливании будет больше, чем $1 / \text{Latency}$.

Для измерения используют батчи большого размера (8, 16, 32 и больше), чтобы достичь максимальной загрузки GPU.

FPS vs Throughput



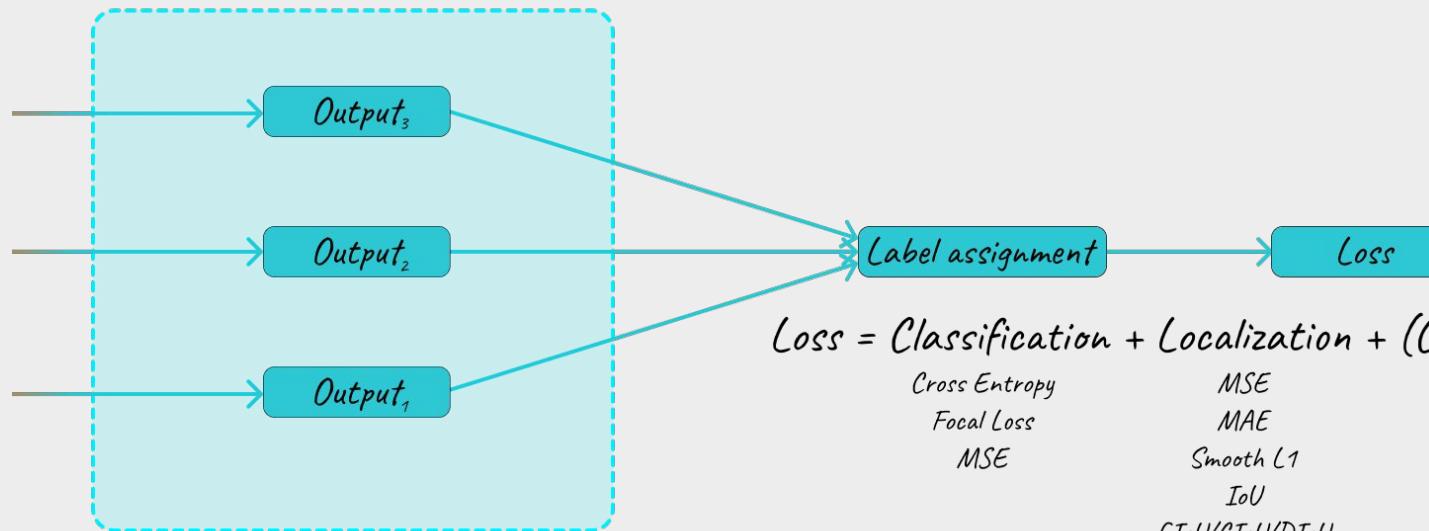
Latency и Throughput в статье YOLOv6.

Как обучать детектор?



Как обучать детектор?

Head



$$Loss = Classification + Localization + (Confidence)$$

Cross Entropy
Focal Loss
MSE

MSE
MAE
Smooth L1
IoU
GIoU/CIoU/DIoU

Cross Entropy

Как обозначить “пустой” бокс?

1. Confidence(objectness) score and classes,
2. Classes ($N + 1$), где 1 это фон,
3. Classes (N), где для ббоксов без объектов истинные вероятности равны нулю.

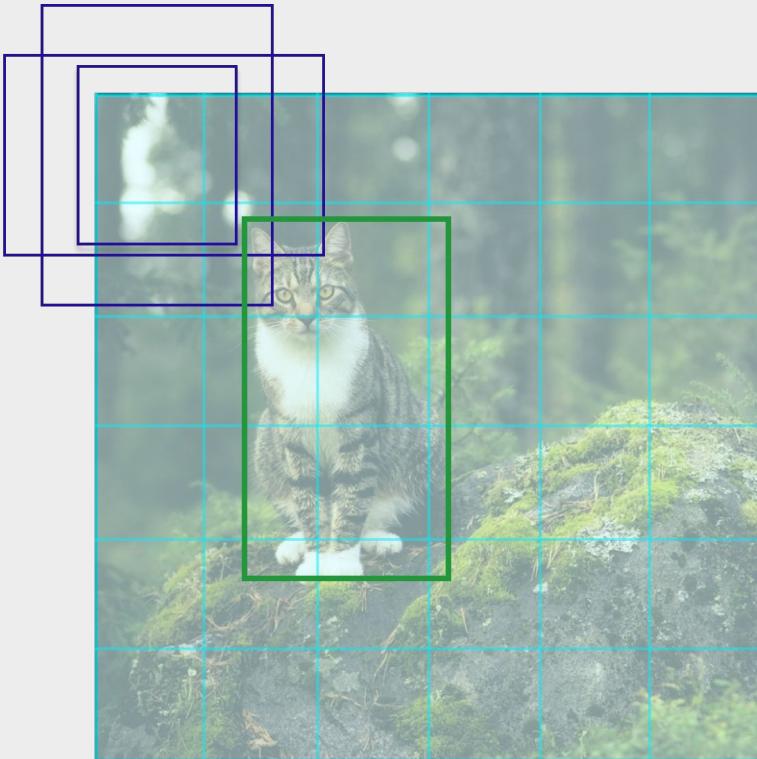
Выбор влияет на лосс.

Label assignment

Label (target/bbox) assignment – этап, призванный ответить на вопрос “**Какие предсказания использовать для обучения модели?**”.

Мы сможем ответить на этот вопрос чуть позже, сейчас же посмотрим на несколько примеров.

Label assignment

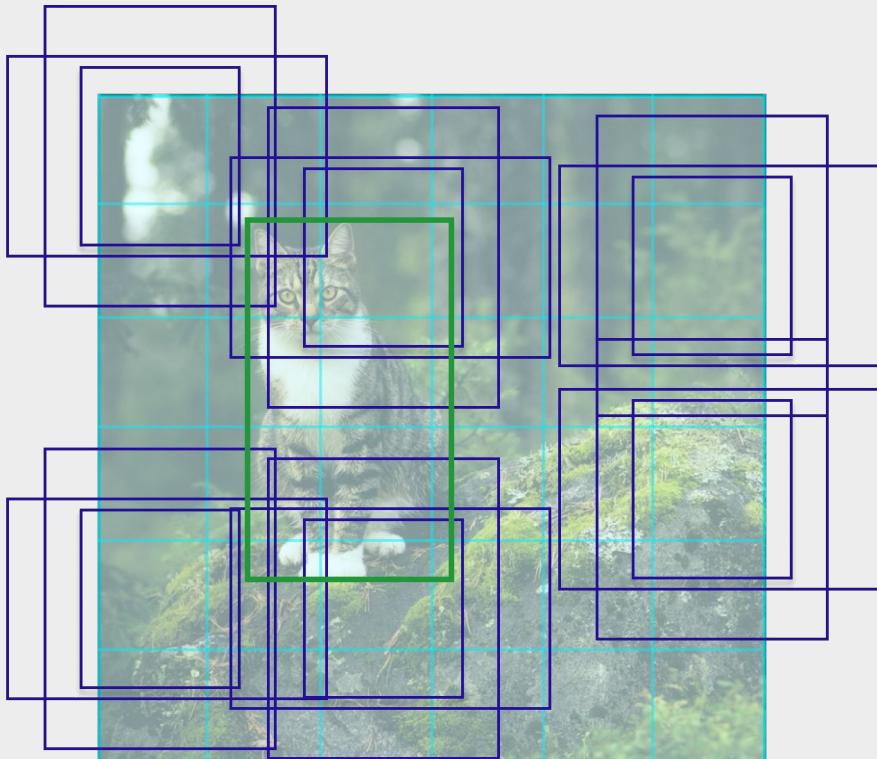


Допустим, для каждого пикселя выходной feature map у нас есть 3 якоря.

Синие квадраты – исходные якоря,
Зеленый – истинный ббокс (GT)



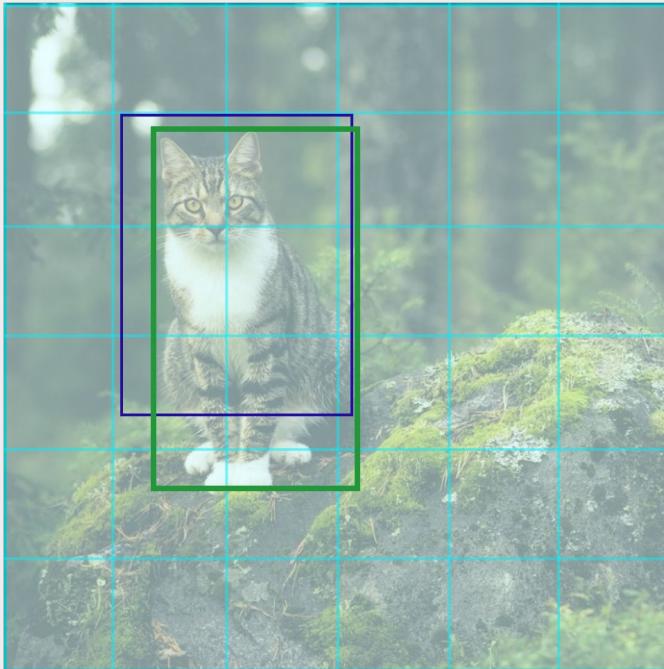
Label assignment



Допустим, для каждого пикселя выходной feature map у нас есть 3 якоря.

Синие квадраты – исходные якоря,
Зеленый – истинный ббокс (GT)

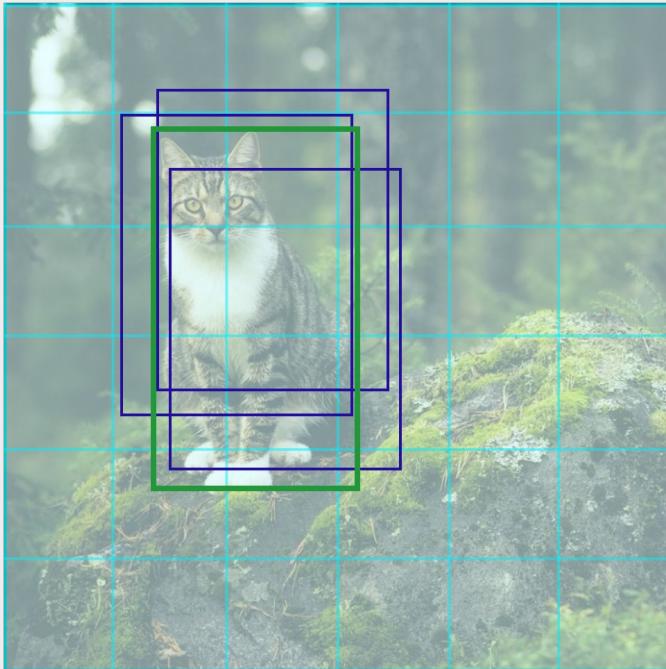
Label assignment. One-to-one.



One-to-one – Для одного бокса выбирается только **одно предсказание**, которое будет участвовать в локализационной части лосса.



Label assignment. One-to-many.



One-to-many – Для одного бокса выбирается несколько предсказаний, которые будут участвовать в локализационной части лосса.



Типы предсказаний



Положительные предсказания – предсказания, которые были сопоставлены определенному **GT**.

Такие предсказания участвуют и в **классификационной** части (их класс совпадает с **GT**), и в **локализационной** части лосса.



Типы предсказаний

Положительные предсказания – предсказания, которые были сопоставлены определенному **GT**.

Такие предсказания участвуют и в **классификационной** части (их класс совпадает с **GT**), и в **локализационной** части лосса.

Отрицательные – предсказания, которые не были сопоставлены ни с одним **GT**.
Они участвуют в **только confidence** части лосса. (Или в **классификационной**, если **confidence** нет).

Типы предсказаний

Положительные предсказания – предсказания, которые были сопоставлены определенному **GT**.

Такие предсказания участвуют и в **классификационной** части (их класс совпадает с **GT**), и в **локализационной** части лосса.

Отрицательные – предсказания, которые не были сопоставлены ни с одним **GT**.
Они участвуют в **только confidence** части лосса. (Или в классификационной, если **confidence** нет).

Нейтральные – предсказания, которые **вообще не участвуют** в обучении.

Classification loss

1. CrossEntropy,
2. Focal Loss,
3. VariFocal Loss



Cross Entropy

$$L = \lambda_{pos} \sum_{i \in \mathcal{P}} \left[- \sum_{c=1}^C y_{i,c} \log \hat{p}_{i,c} \right] + \lambda_{neg} \sum_{j \in \mathcal{N}} [- \log \hat{p}_{j,0}]$$

Где,

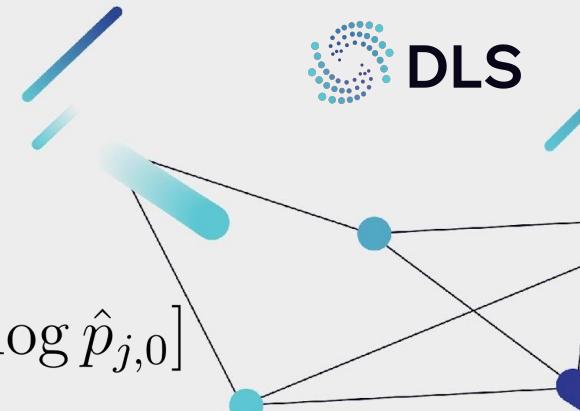
λ_{pos} – Вес для положительных ббоксов,

λ_{neg} – Вес для отрицательных ббоксов,

\mathcal{P} – Множество положительных ббоксов,

\mathcal{N} – Множество отрицательных ббоксов.

Считается для положительных и иногда для отрицательных предсказаний.



Focal Loss

$$FL(\hat{p}_i, p_i) = \alpha_t(1 - p_t)^\gamma \log(p_t)$$

$$p_t = \begin{cases} p, & \text{if } y = 1 \\ 1 - p, & \text{otherwise} \end{cases}$$

$$\alpha_t = \begin{cases} \alpha, & \text{if } y = 1 \\ 1 - \alpha, & \text{otherwise} \end{cases}$$

Простая интерпритация
для детекции:

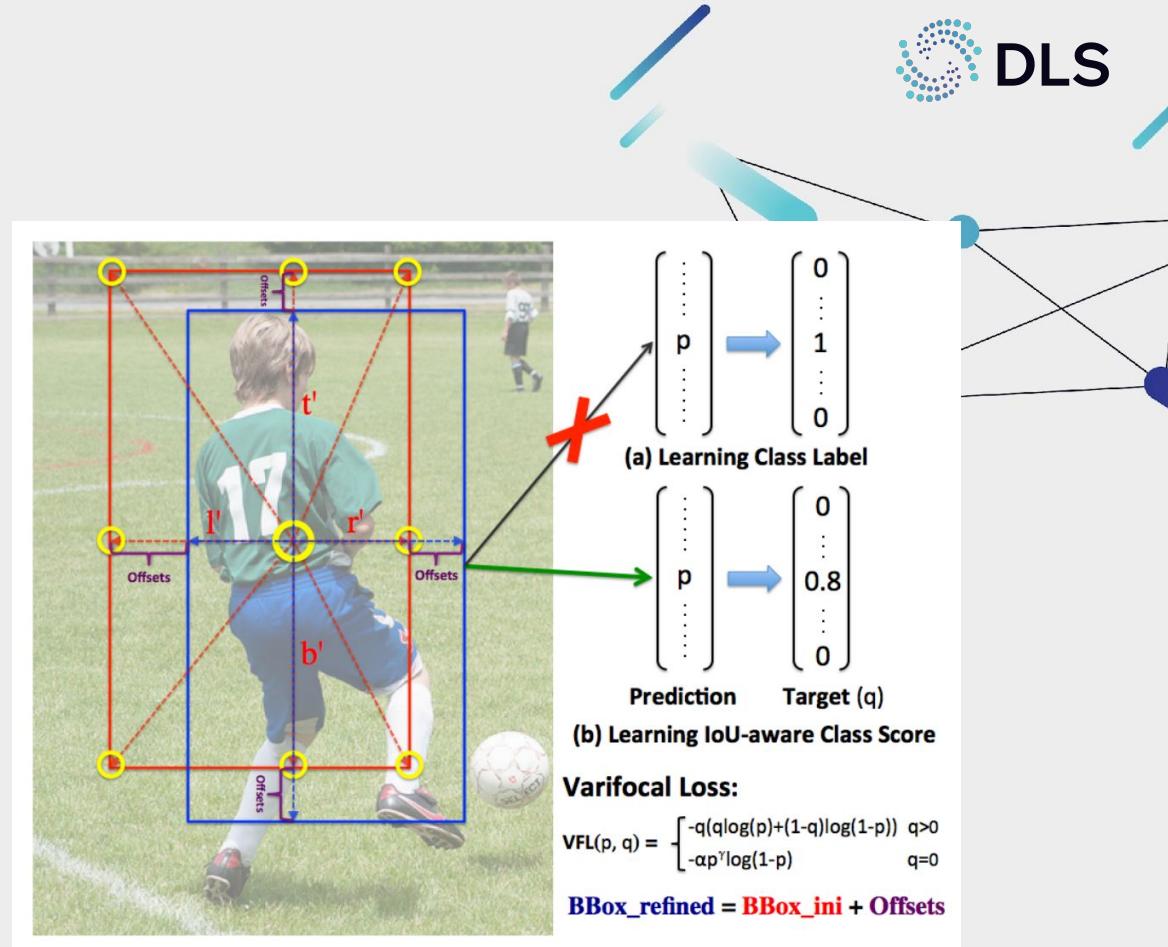
Если $y = 1$, то рассматриваемый ббокс – “положительный”. Таким образом мы можем уменьшать значимость большого количества “отрицательных” ббоксов.



VariFocal Loss

Идеи:

1. ассиметрично оценивать “положительные” и “отрицательные” ббоксы.
Для положительных используем BCE, для отрицательных – Focal loss.
2. Соединить классификационный и локализационный лоссы за счет перевешивания таргета ($Q = \text{IoU}$).



Source: VarifocalNet: An IoU-aware Dense Object Detector
<https://arxiv.org/abs/2008.13367>

Localization loss

1. MSE/MAE loss,
2. SmoothL1 (Huber loss),
3. Intersection over Union (IoU),
4. Generalized IoU(GIoU),
5. Distance IoU(DIoU),
6. Complete IoU (CIoU),
7. Efficient IoU (EIoU).



SmoothL1 (Huber loss)



$$\text{smooth } L_1(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

Комбинация L1 и L2, которая использует L2 для ошибок меньше 1, и L1 для всех остальных значений.

Идея: минимизируем разницу между предсказанными и реальными размерами боксов.



Intersection over Union

Intersection over Union (IoU):

$$\text{Loss} = 1 - \text{IoU}$$



Intersection over Union

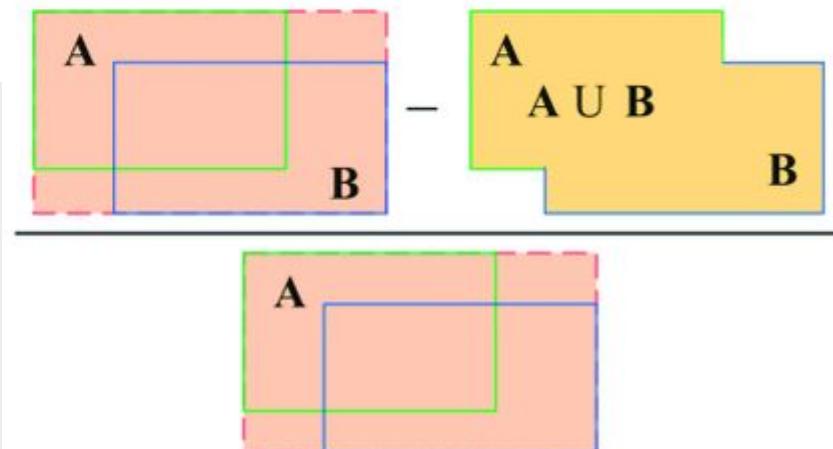
Intersection over Union (IoU):

$$\text{Loss} = 1 - \text{IoU}$$

Generalized IoU (GIoU):

$$GIoU = IoU - \frac{|C \setminus (A \cup B)|}{|C|}$$

Идея: Добавление минимальной выпуклой формы (C) помогает в случае, когда боксы не пересекаются. В этом случае, у IoU лосса не будет градиентов, а у GIoU будут.



Distance IoU (DIoU)

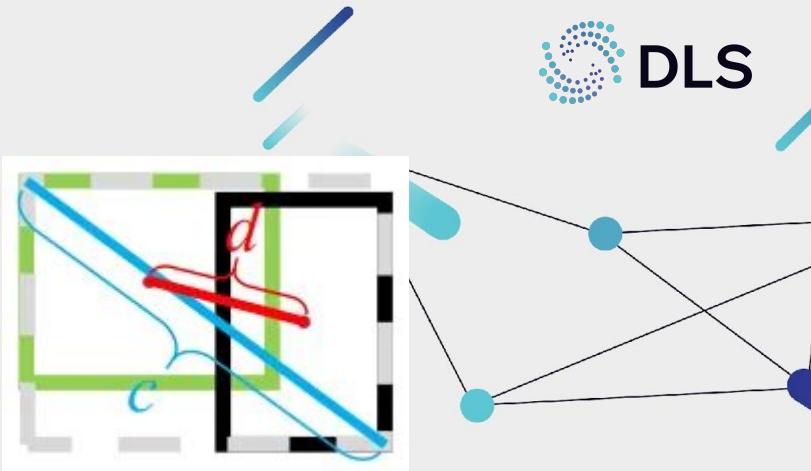
$$DIoU = IoU - \frac{\rho^2(b, b^{gt})}{c^2}$$

Где,

ρ – Евклидово расстояние между координатами центров предсказанного и истинного ббоксов (красная линия на рисунке),

c – Длина диагонали минимальной выпуклой формы для двух ббоксов.

Идея: Напрямую минимизируем расстояние между центрами ббоксов.



Complete IoU (CIoU)

$$L_{CIoU} = L_{DIoU} + \alpha v$$

где,

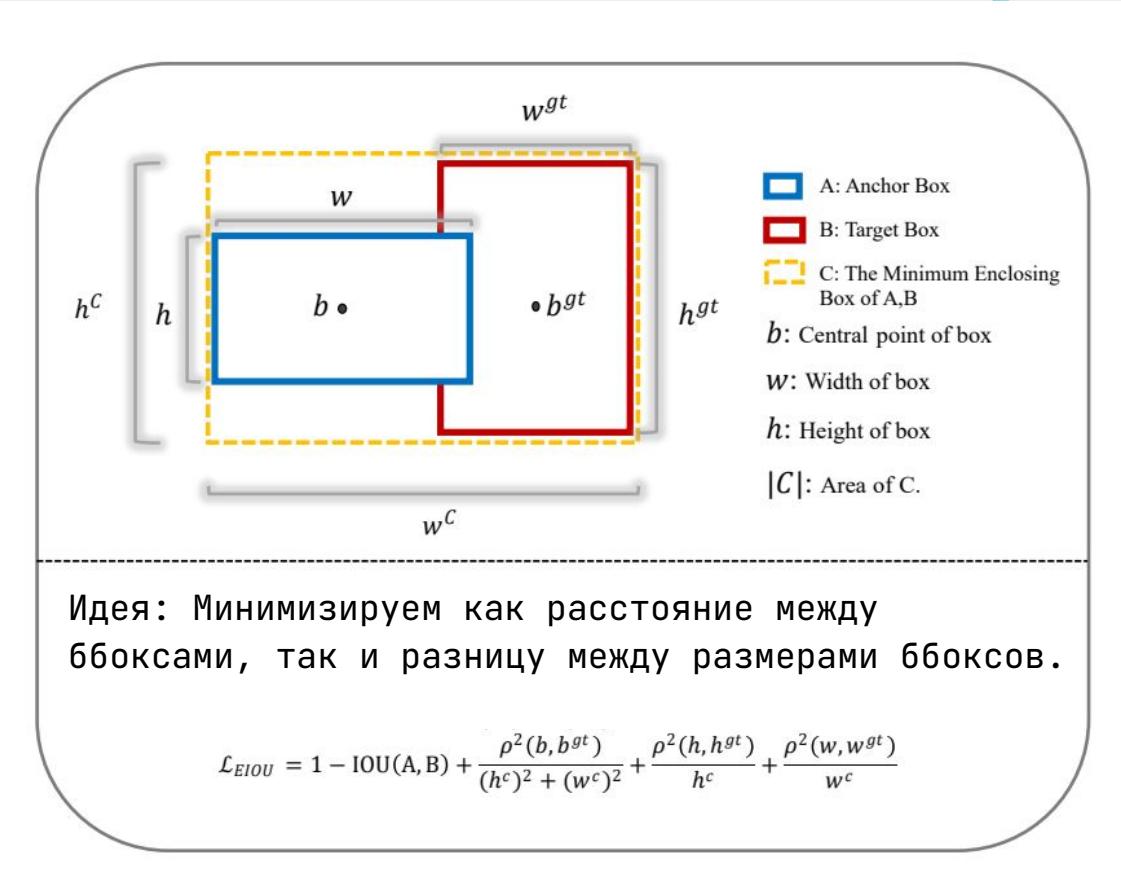
$v = \frac{4}{\pi^2} \left(\arctan \frac{w_{gt}}{h_{gt}} - \arctan \frac{w_p}{h_p} \right)^2$ – Отражает разницу в соотношении сторон между ббоксами,

$\alpha = \frac{v}{(1 - \text{IoU}) + v}$ – Коэффициент баланса между DIoU и соотношением сторон (Равен нулю при $\text{IoU} < 0.5$).

Идея: Объединяем ошибку локализации ($\text{IoU} +$ расстояние между центрами) и ошибку форму через соотношение сторон и балансировочным коэффициентом.



Efficient IoU (EIoU)



Source: Focal and Efficient IoU Loss for Accurate

Bounding Box Regression

<https://arxiv.org/abs/2101.08158>

Focal-EIoU Loss

$$L_{Focal-EIoU} = (1 - IoU(b, b^{gt}))^\gamma L_{EIoU}$$

Где,

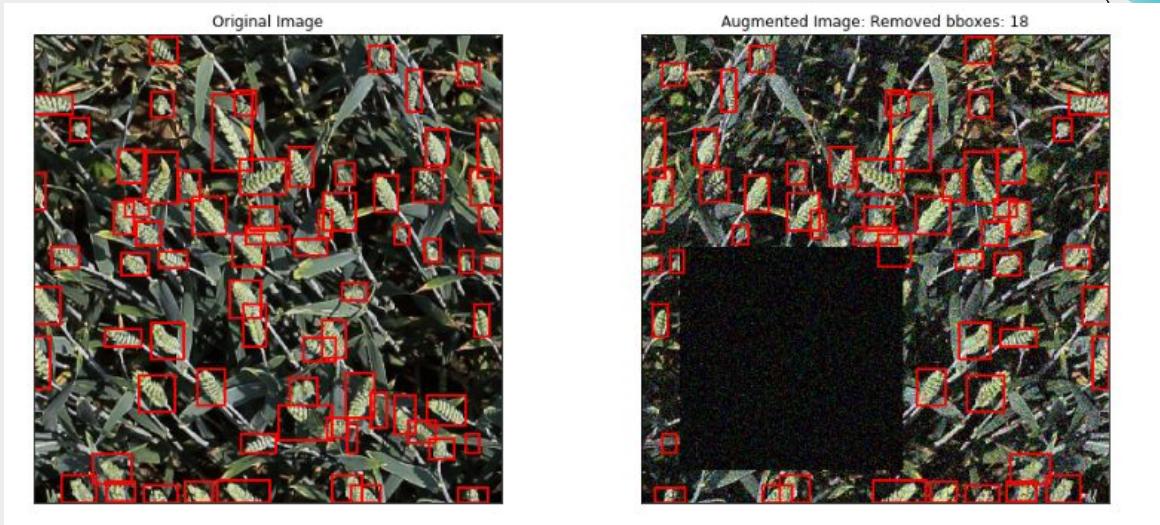
γ – параметр фокусировки, контролирующий степень переобучения на сложных примерах

Идея: Давайте поднимать лосс для примеров с маленьким IoU.

Augmentations



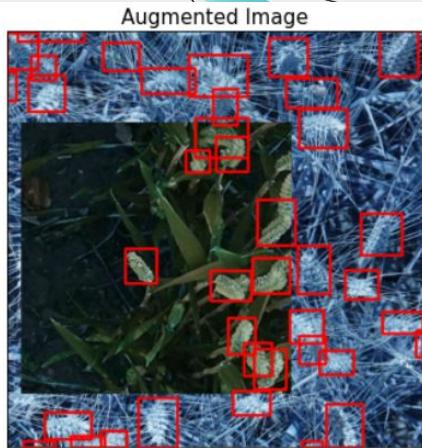
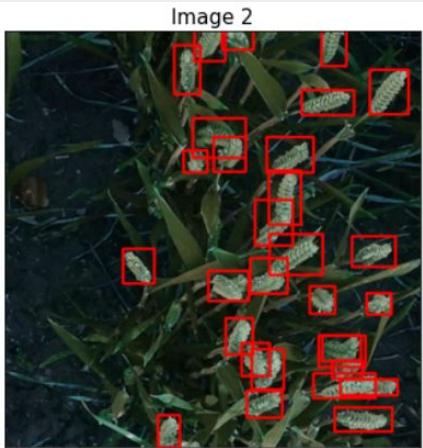
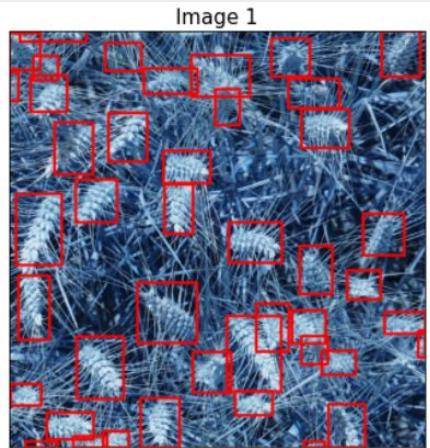
Cutout



Source: <https://www.kaggle.com/code/kaushal2896/data-augmentation-tutorial-basic-cutout-mixup>

Удаляем часть изображения и все боксы внутри

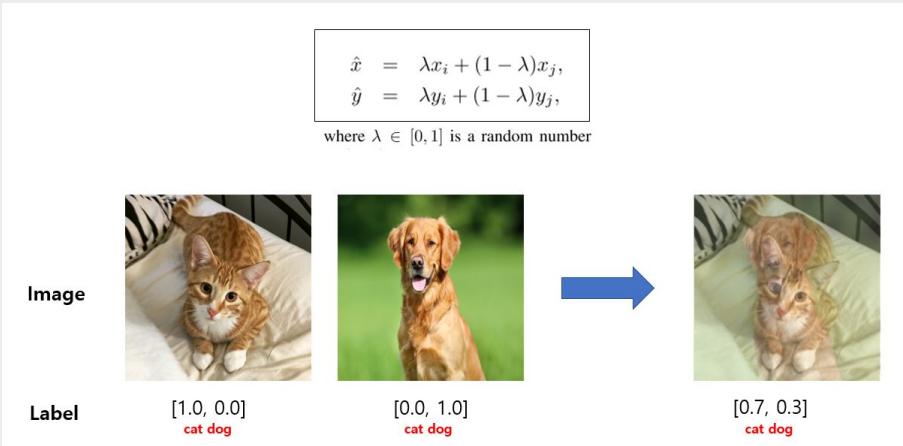
CutMix



Source: <https://github.com/keras-team/keras-cv/issues/35>

Объединяя два изображения и не забывая перенести ббоксы

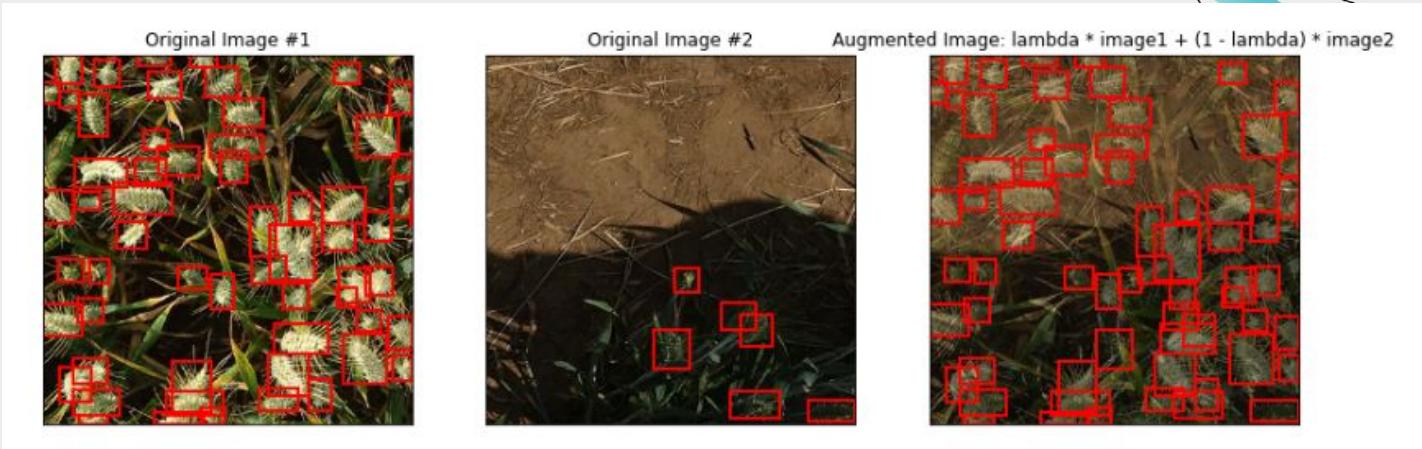
Mixup



Source: <https://www.kaggle.com/code/kaushal2896/data-augmentation-tutorial-basic-cutout-mixup>

Накладываем одно изображение на другое с какой-то прозрачностью

Mixup



Source: <https://www.kaggle.com/code/kaushal2896/data-augmentation-tutorial-basic-cutout-mixup>

При этом на итоговом изображении есть все ббоксы, у которых
confidence score меняется пропорционально прозрачности

Mosaic Augmentation



Комбинируем 4 изображения в одно, смешивая их контекст и изменяя размер объектов.



Mosaic Augmentation



Комбинируем 4 изображения в одно, смешивая их контекст и изменяя размер объектов.



Для ббоксов считается итоговое соотношение сторон и, если оно изменилось больше, чем заданный порог, ббокс удаляется.

Заключение

Сегодня мы с вами узнали:

- Как выглядит общая структура детектора,
- Как он делает предсказания,
- Основные метрики в детекции,
- Как обучается детектор и какие бывают лосс функции,
- Какие аугментации используются в детекции.



В следующей серии

Переходим к разбору детекторов и поговорим о:

- Семействе RCNN,
- И разберем множество архитектур, таких как
- SSD,
- YOLO,
- FPN,
- RetinaNet,
- FCOS,

И другие.

