

Deep Learning School

Задача детекции. Обзор моделей. Часть 2.



План

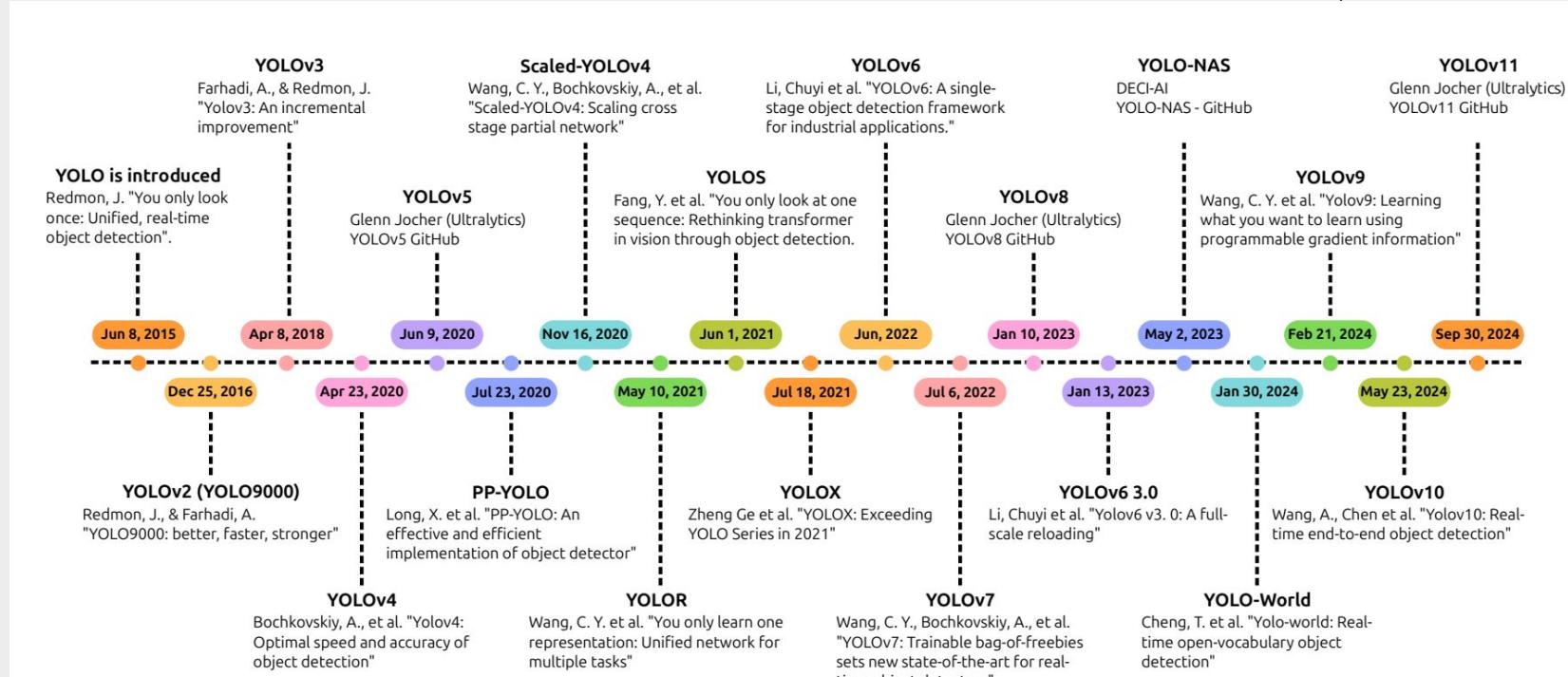
1. Семейство You Look Only Once (YOLO):
 - 1.1. Какое влияние оказало семейство на развитие детекторов,
 - 1.2. Новые подходы для детекции.
2. Необычные подходы к задаче детекции:
 - 2.1. CenterNet,
 - 2.2. CornerNet,
 - 2.3. Использование поворотных bounding boxes.



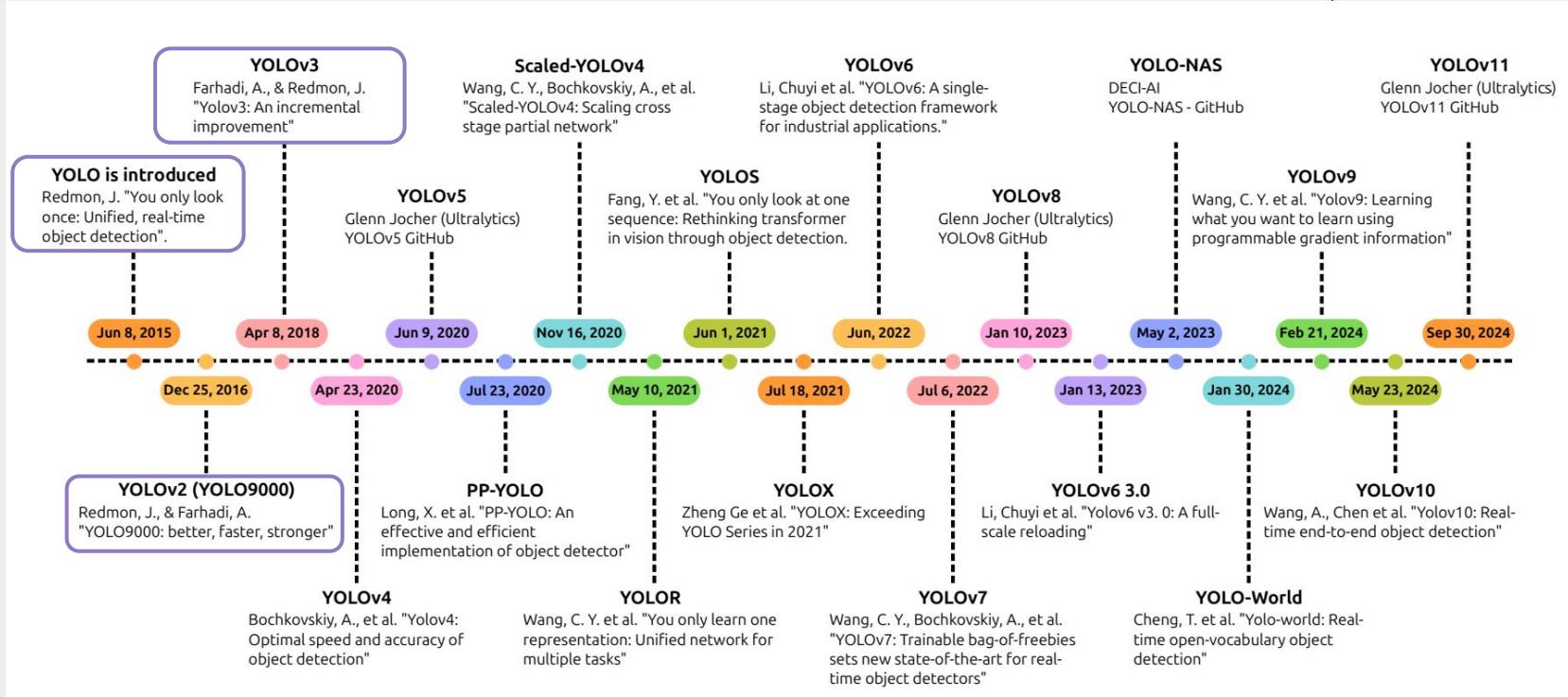
Семейство YOLO



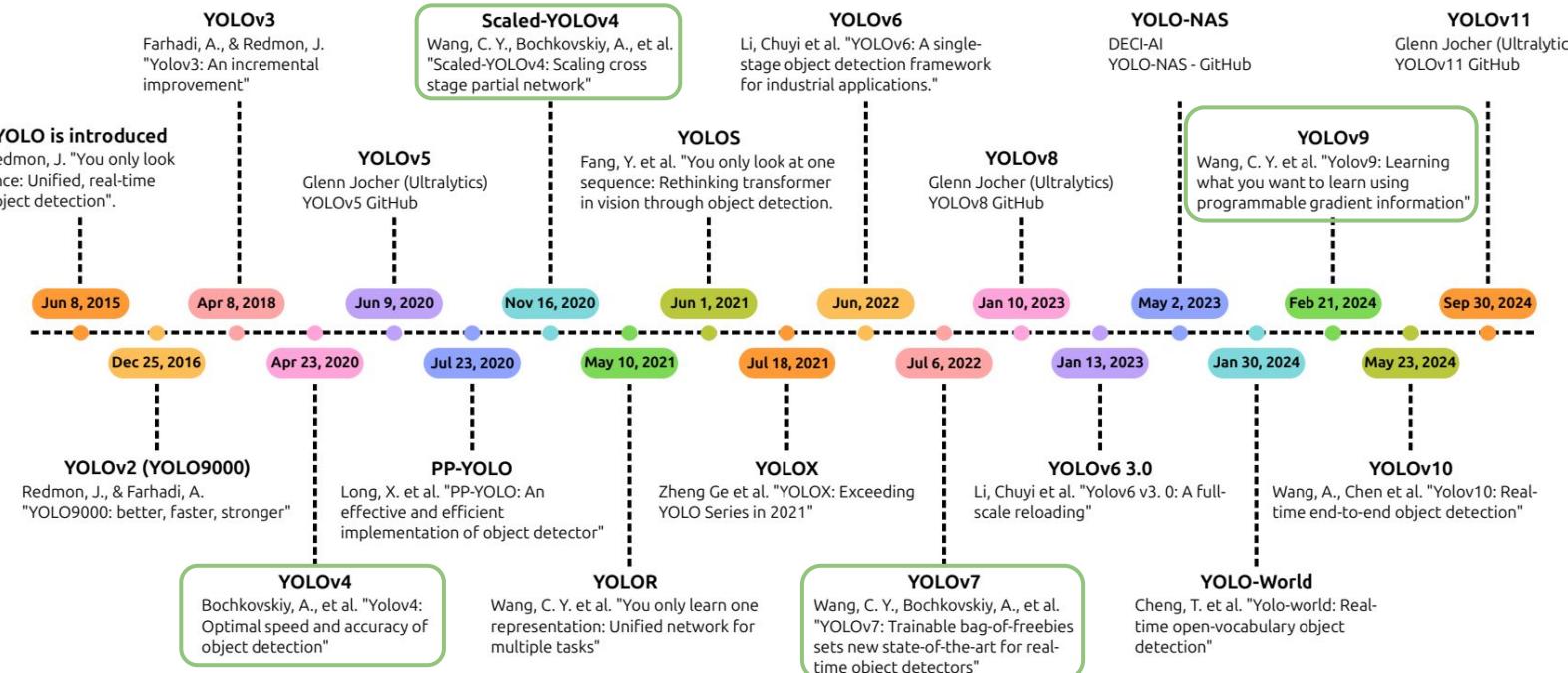
Семейство YOLO



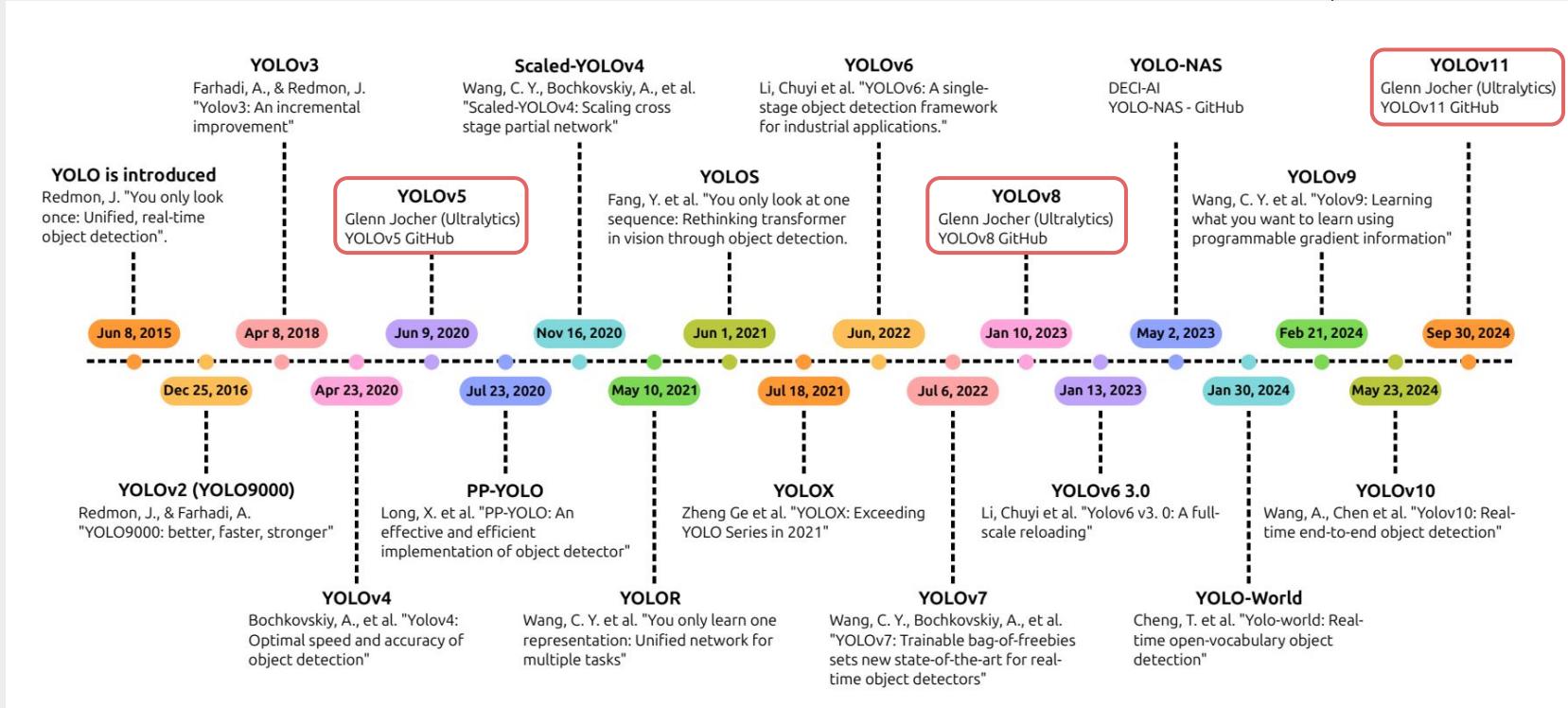
Семейство YOLO



Семейство YOLO



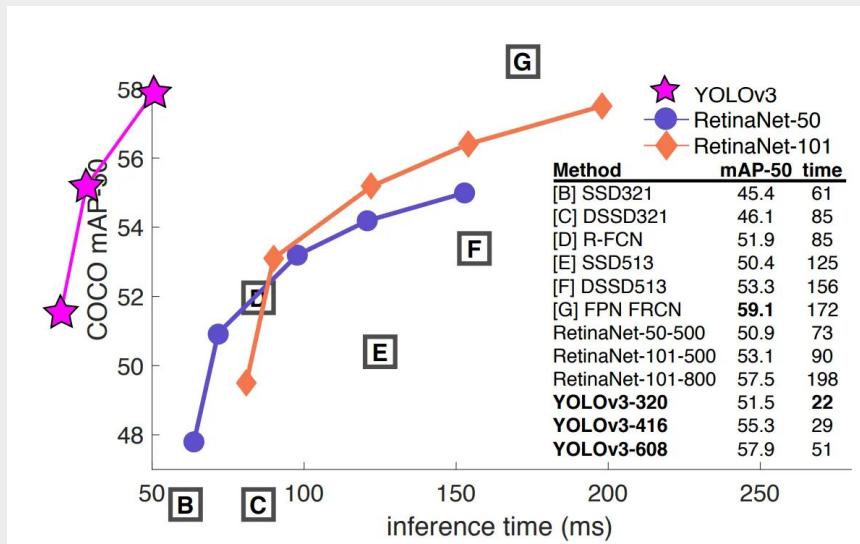
Семейство YOLO



YOLO. Несколько фактов.

- YOLO – семейство real-time моделей детекции, которые изначально спроектированы так, чтобы обеспечивать высокую скорость работы с минимальной потерей точности.

Например, YOLOv3 была чуть хуже уже вышедших моделей (например RetinaNet), но при этом работала в 4 раза быстрее.



YOLO. Несколько фактов.

- YOLO – семейство real-time моделей детекции, которые изначально спроектированы так, чтобы обеспечивать высокую скорость работы с минимальной потерей точности.
- В отличии от остальных детекторов 15-18 годов, первые версии используют собственную backbone модель – Darknet. Её главная особенность – маленький размер (существует в двух вариациях 19 и 53 слоя).



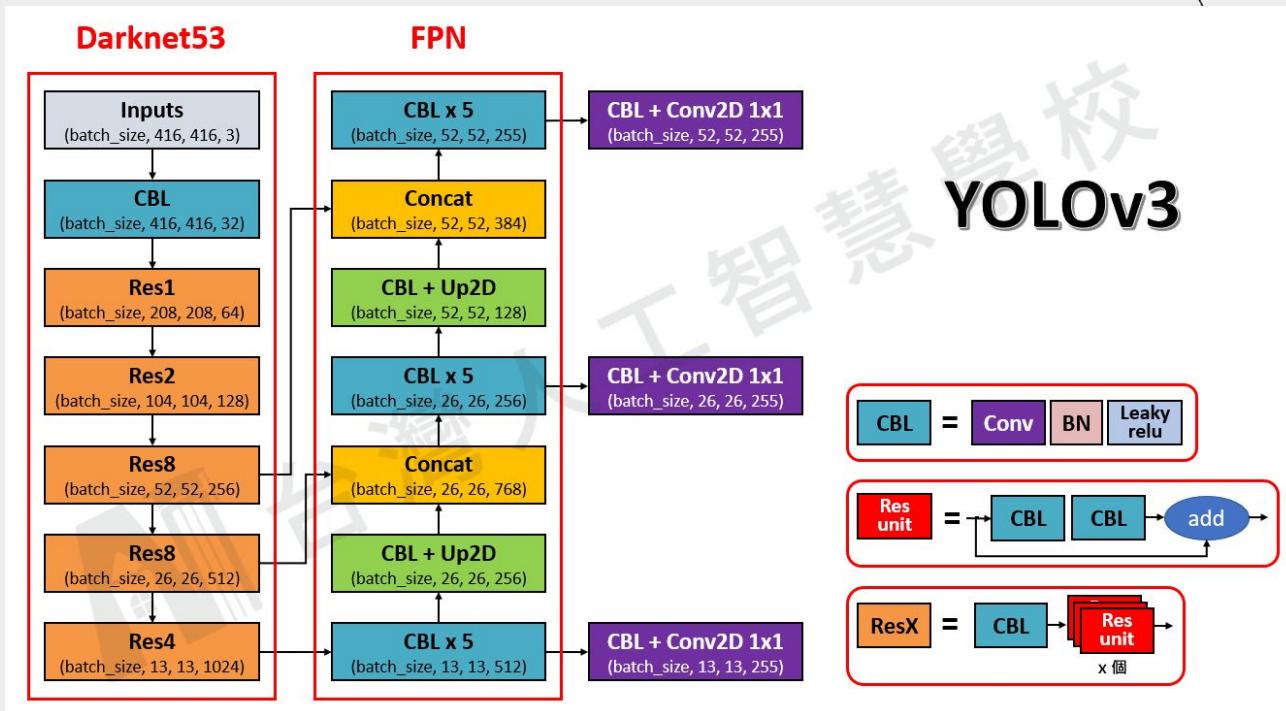
YOLO. Несколько фактов.

- YOLO – семейство real-time моделей детекции, которые изначально спроектированы так, чтобы обеспечивать высокую скорость работы с минимальной потерей точности.
- В отличии от остальных детекторов 15-18 годов, первые версии используют собственную backbone модель – Darknet. Её главная особенность – маленький размер (существует в двух вариациях 19 и 53 слоя).
- Также как у EfficientDet, YOLO архитектуры легко адаптируются под различные задачи. Обычно размеры моделей обозначаются буквами (n, m, s, l, x) в порядке увеличения модели. Модели для картинок большого размера обозначаются припиской "6" или -P. Например YOLOv5n6 или YOLOv4-P7.

YOLO. Несколько фактов.

- YOLO – семейство real-time моделей детекции, которые изначально спроектированы так, чтобы обеспечивать высокую скорость работы с минимальной потерей точности.
- В отличии от остальных детекторов 15-18 годов, первые версии используют собственную backbone модель – Darknet. Её главная особенность – маленький размер (существует в двух вариациях 19 и 53 слоя).
- Также как у EfficientDet, YOLO архитектуры легко адаптируются под различные задачи. Обычно размеры моделей обозначаются буквами (n, m, s, l, x) в порядке увеличения модели. Модели для картинок большого размера обозначаются припиской "6" или -P. Например YOLOv5n6 или YOLOv4-P7.
- Это самое популярное и самое простое в использовании семейство детекторов.
 - Ultralytics,
 - mmdetection,
 - Буквально у каждой модели есть код от авторов.

YOLOv3. Сквозь модели.

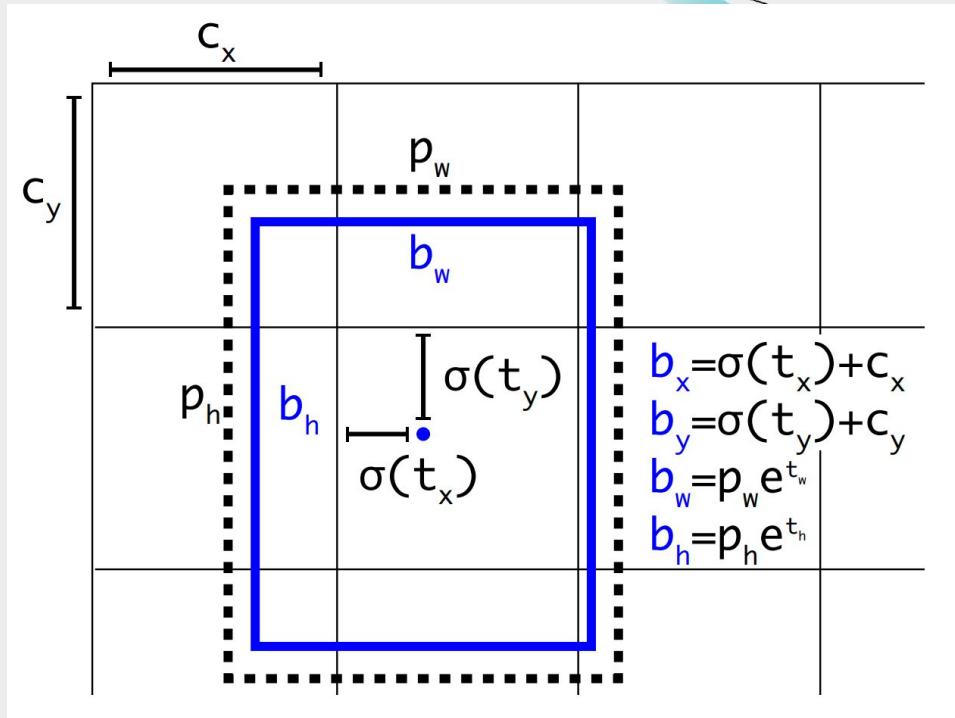


YOLOv3/v2. Output.

YOLOv2 и v3 предсказывали смещения для якорей.

Где,

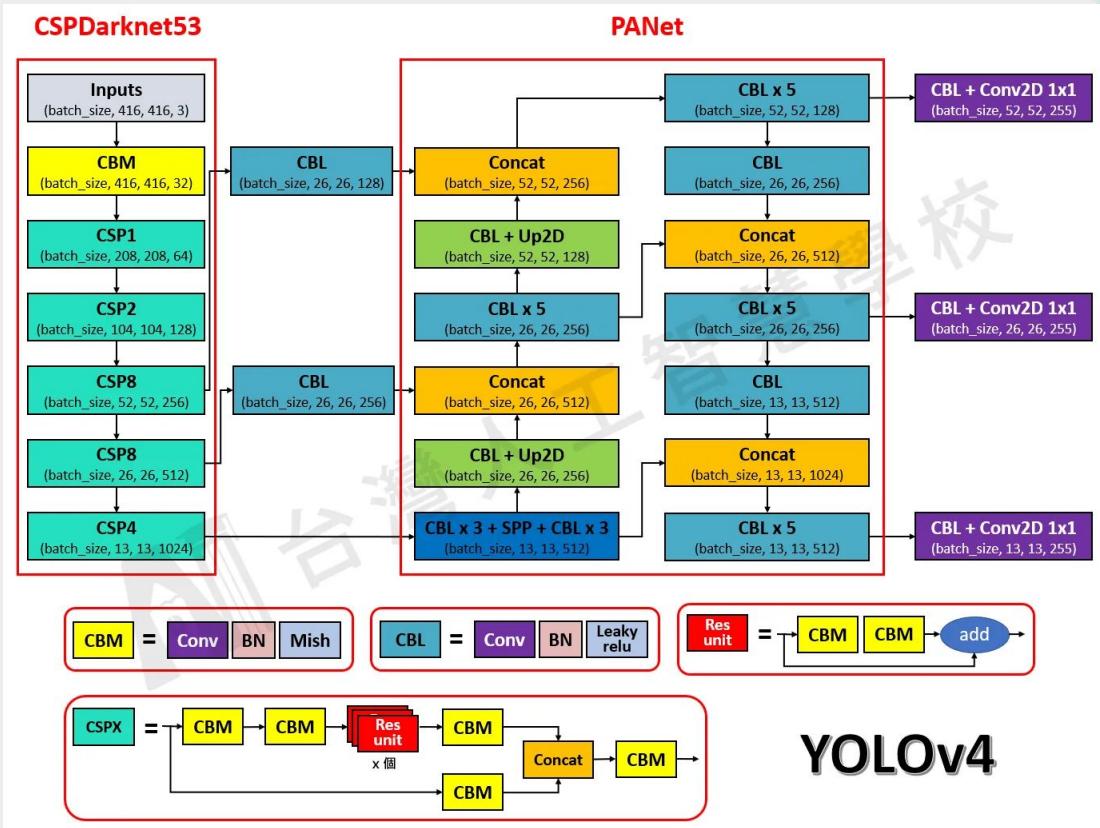
- t_x, t_y, t_w, t_h – предсказания модели,
- c_x, c_y – координаты левого верхнего угла ячейки,
- p_w, p_h – априорные значения высоты и ширины анкера,
- σ – сигмойда.



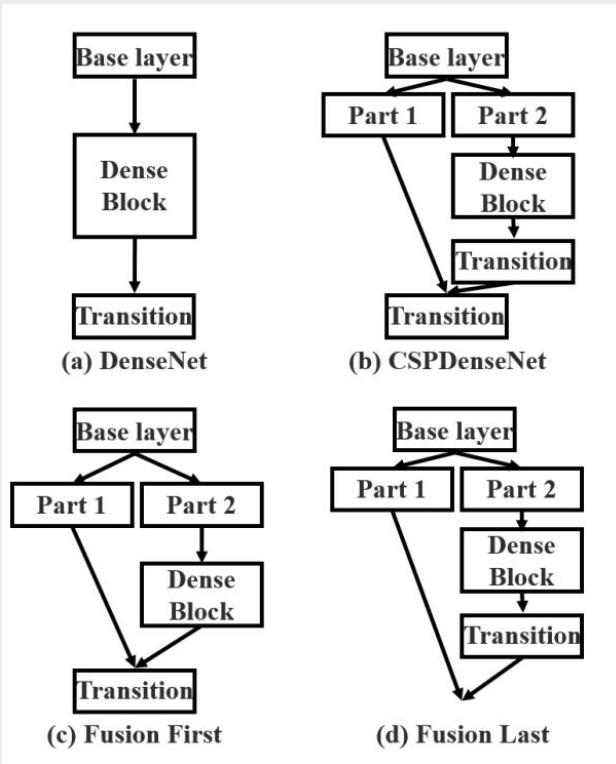
Source: YOLO9000: Better, Faster, Stronger

<https://arxiv.org/abs/1612.08242>

YOLOv4. Сквозь модели.

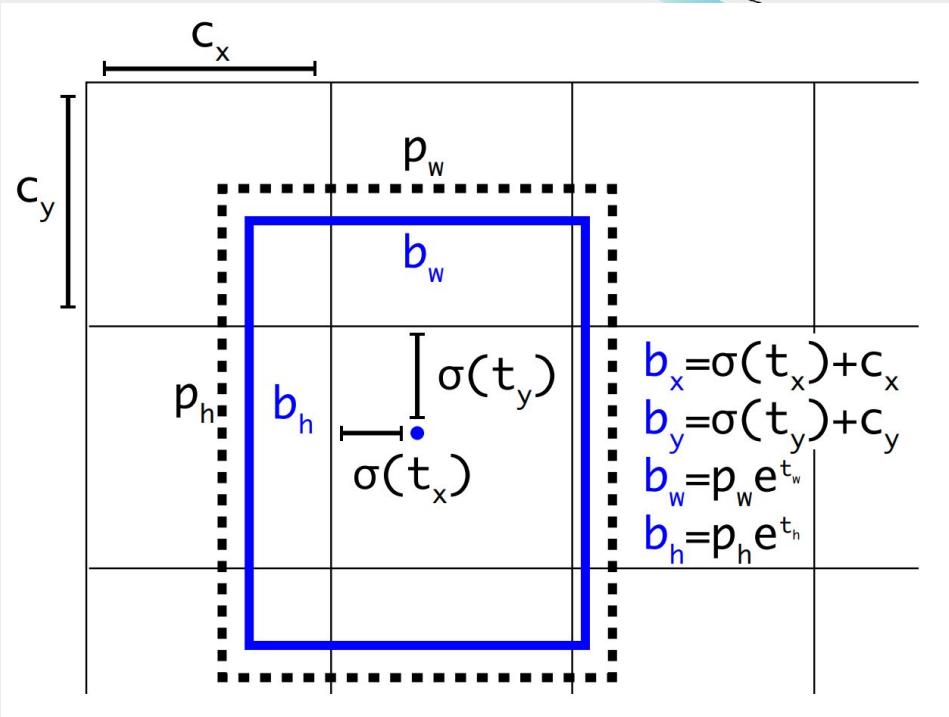


YOLOv4. Сквозь модели.



YOLOv4. Output.

Оказалось, что способ предсказания смещений из YOLOv3 работает плохо, если истинная координата центра бокса находится на границе ячейки:
 $b_x = c_x$ или $b_x = c_x + 1$.
Другими словами t должен быть $\pm\infty$.



Source: YOL09000: Better, Faster, Stronger

<https://arxiv.org/abs/1612.08242>

YOLOv4. Output.

Оказалось, что способ предсказания смещений из YOLOv3 работает плохо, если истинная координата центра бокса находится на границе ячейки:

$$b_x = c_x \text{ или } b_x = c_x + 1.$$

Другими словами t должен быть $\pm\infty$.

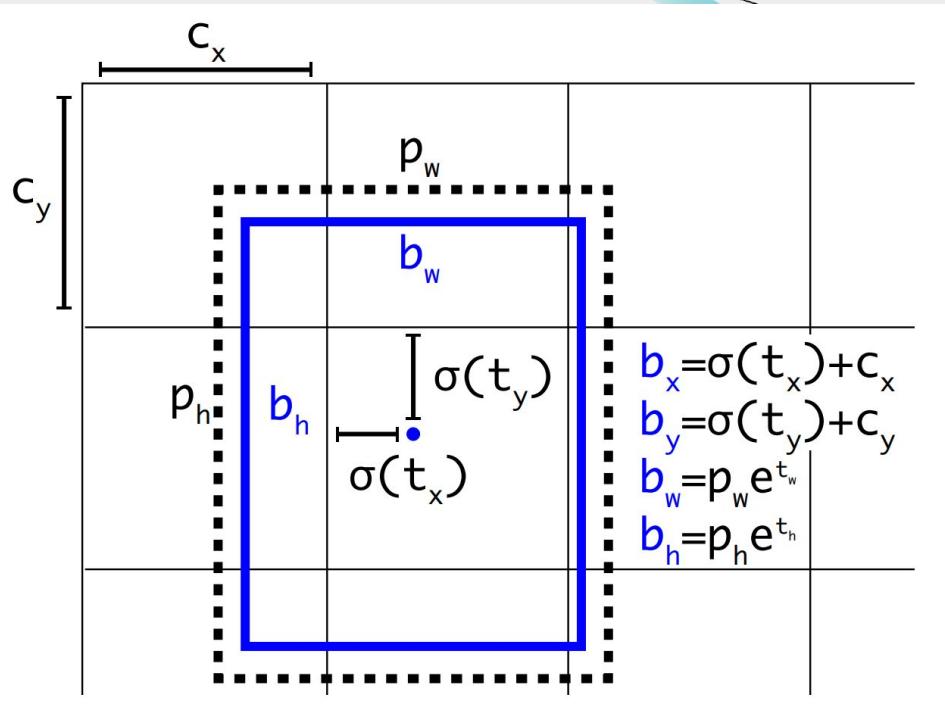
Введем константу $S > 1$:

$$b_x = \left(\sigma(t_x) * S - \frac{S-1}{2} \right) + c_x$$

$$b_y = \left(\sigma(t_y) * S - \frac{S-1}{2} \right) + c_y$$

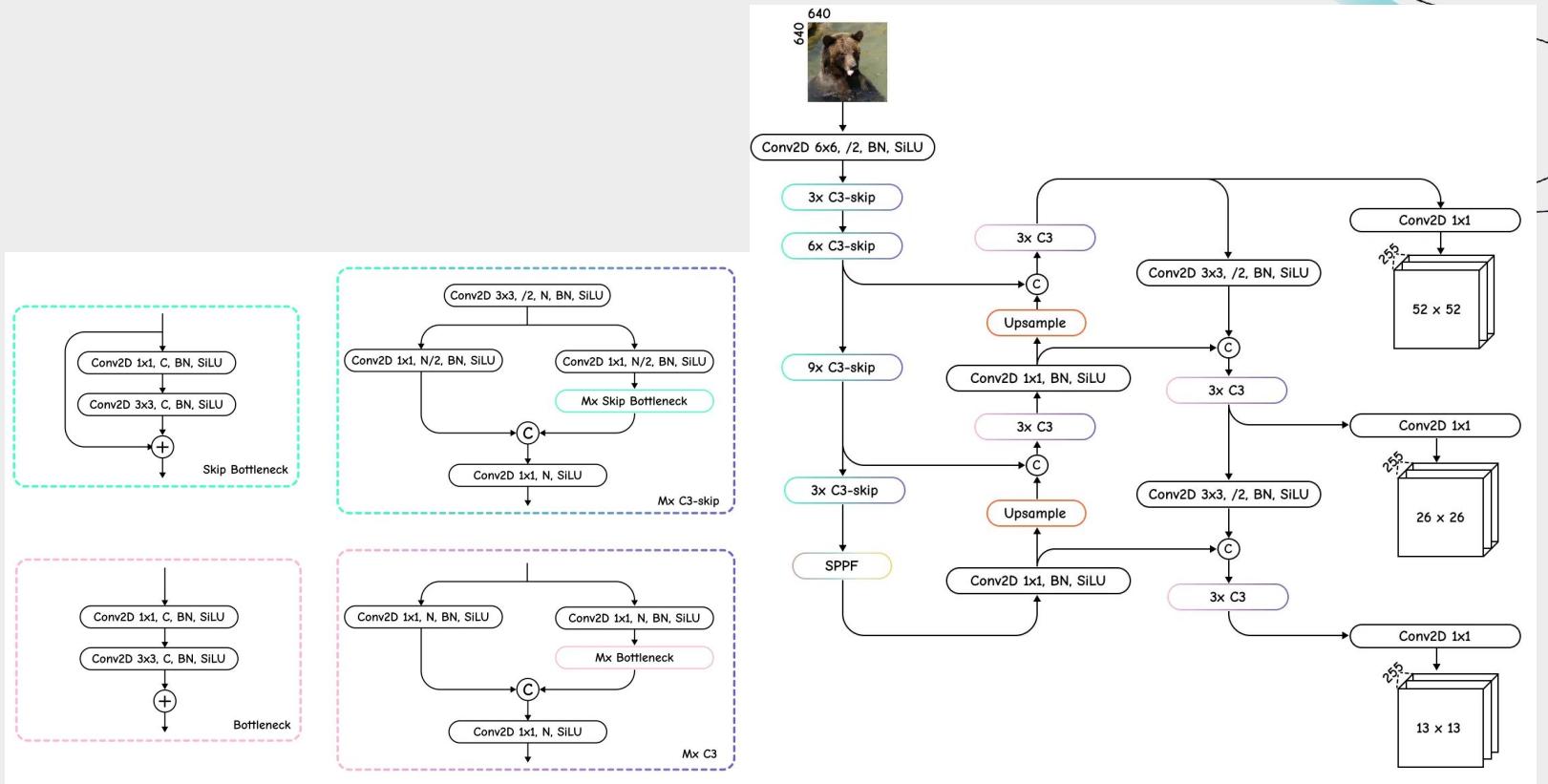
$$b_w = p_w * (\sigma(t_w) * S)^2$$

$$b_h = p_h * (\sigma(t_h) * S)^2$$



Source: YOLO9000: Better, Faster, Stronger
<https://arxiv.org/abs/1612.08242>

YOLOv5. Сквозь модели.



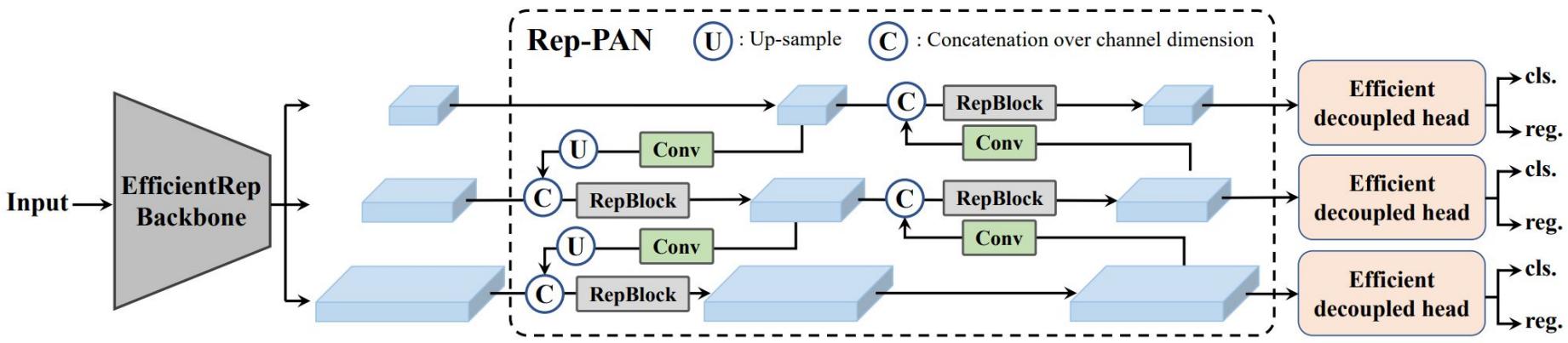
YOLOv5. Сквозь модели.

Основная особенность модели – экосистема.

Авторы модели компания Ultralytics, которая сразу создавала удобную среду. Вместе с моделью, авторы сделали:

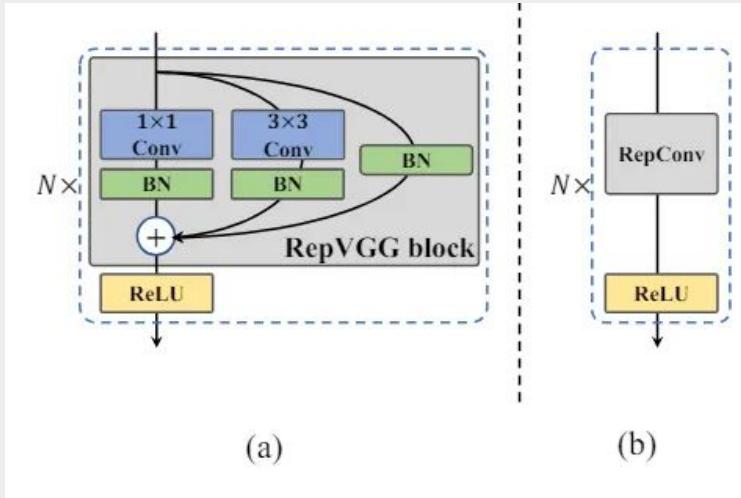
- Удобный интерфейс для использования модели: как запустить модель из коробки, обучить и сделать инференс,
- Легкая конвертация в другие форматы (TensorRT и тд),
- Есть связки с самыми популярными сервисами мониторинга (например clearml),
- Много фишек для MLOps и CICD.

YOLOv6. Сквозь модели.



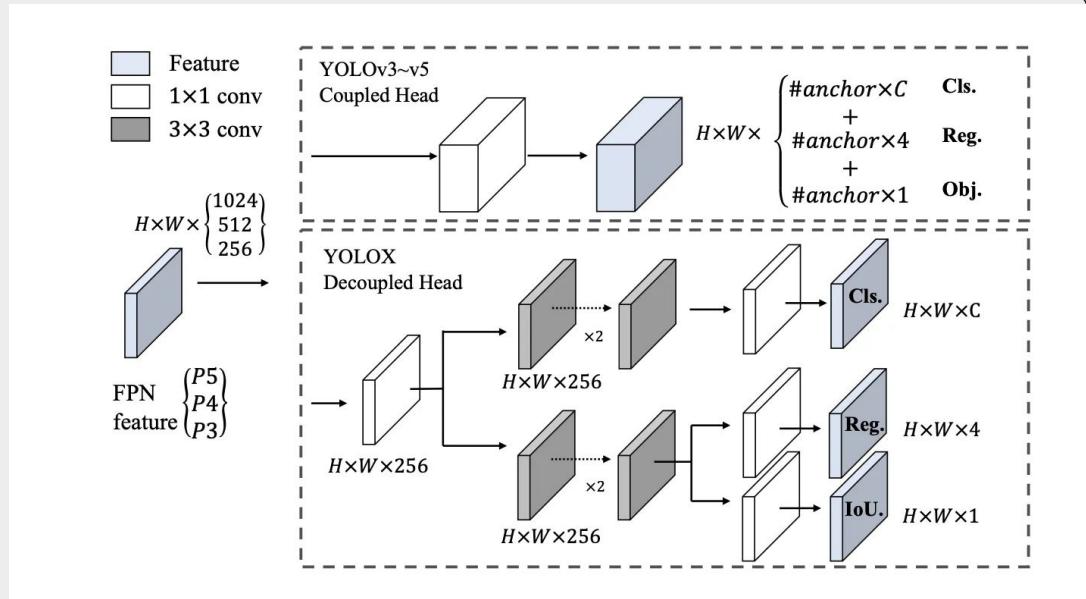
Source: YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications
<https://arxiv.org/abs/2209.02976>

YOLOv6. Сквозь модели.



Используя **reparameterization trick** на инференсе, мы можем представить RepVGG блок как один конволюционный блок 3×3 . Это позволяет значительно уменьшить время инференса.

YOLOv6. Сквозь модели.



На картинке Decoupled Head из YOLOX.

В YOLOv6 его упростили:

1. Убрали один коволюционный слой, после разделения на 2 ветви
2. Начали изменять количество каналов в зависимости от размера feature map.

Task Alignment Learning

TAL это алгоритм для решения задачи Label assignment.

Основная идея – учитывать не только положение предсказаний, то и класс предсказания.



Task Alignment Learning

TAL это алгоритм для решения задачи Label assignment.

Он состоит из следующих этапов:

1. Для каждого предсказанного ббокса вычисляем метрику t :

$$t = s^\alpha \times u^\beta$$

Где,

- s – classification score (вероятность принадлежности к классу GT),
- u – IoU между предсказанием и GT,
- α, β – нормализационные константы (обычно 6 и 1),

2. Фильтруем предсказания по GT:

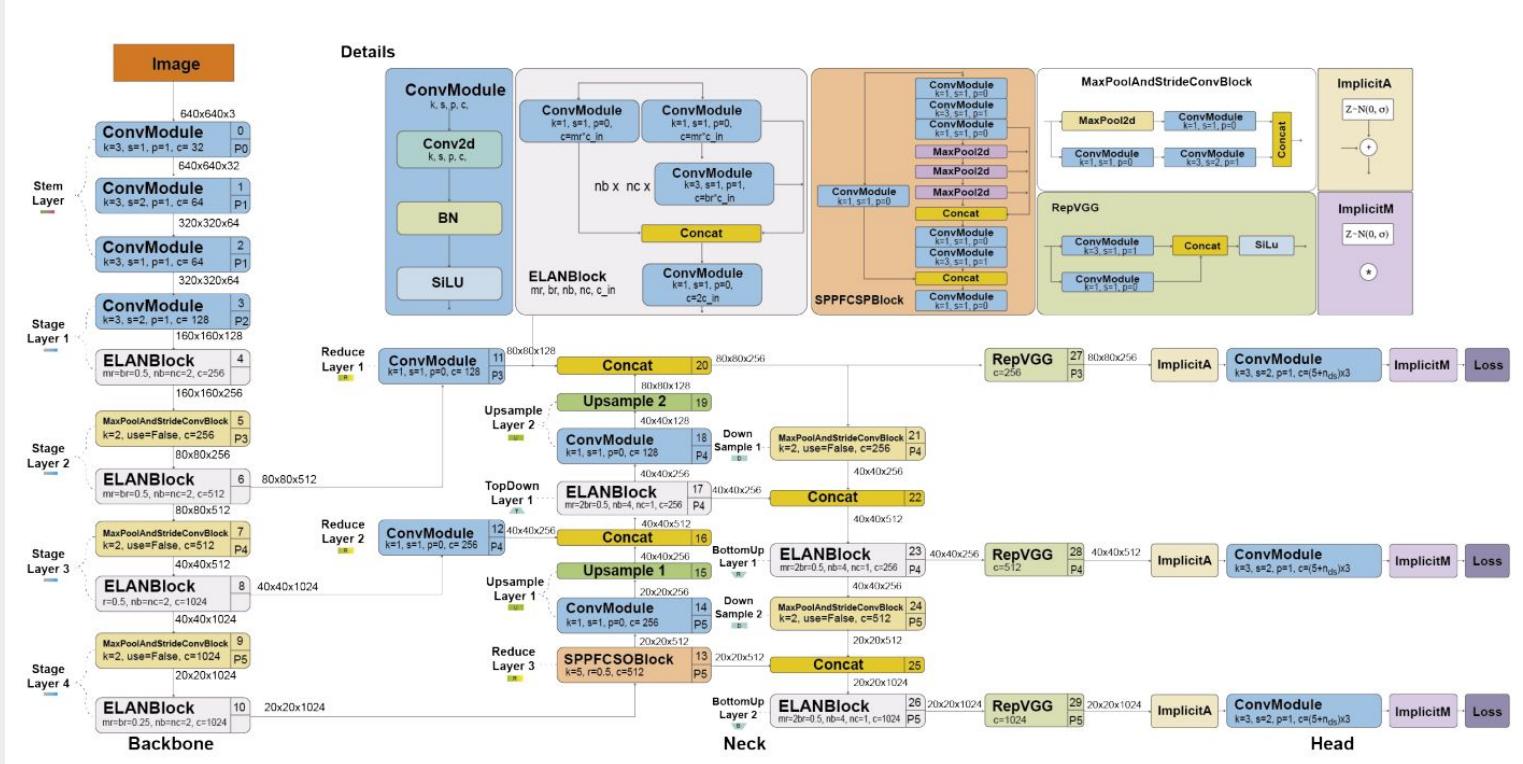
Выбираем только те предсказания, центры которых находятся внутри GT.

3. Для каждого GT, top-k предсказаний с наивысшим значением метрики t обозначаются “положительными”.

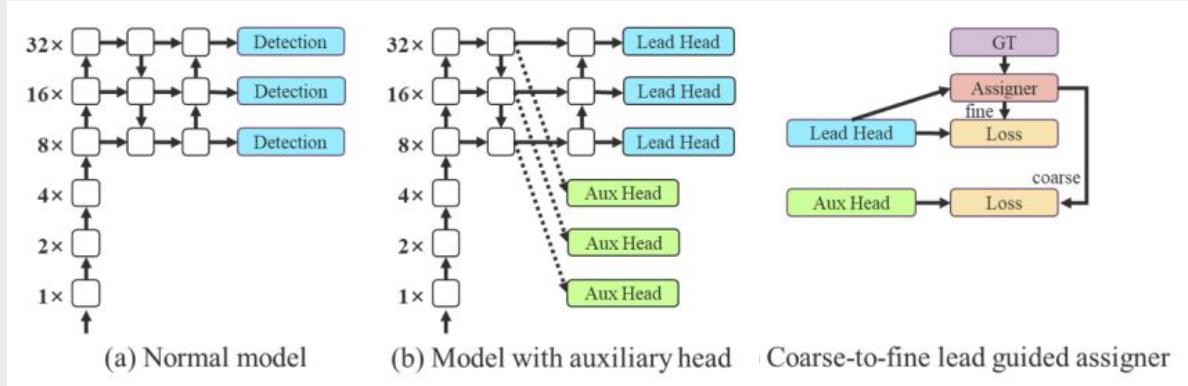
Если предсказание соответствует нескольким GT, оно назначается тому, с которым IoU больше.



YOLOv7. Сквозь модели.



YOLOv7. Сквозь модели.

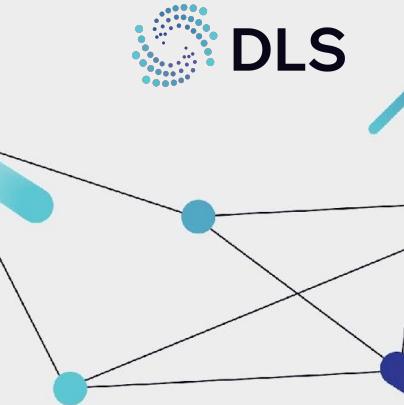
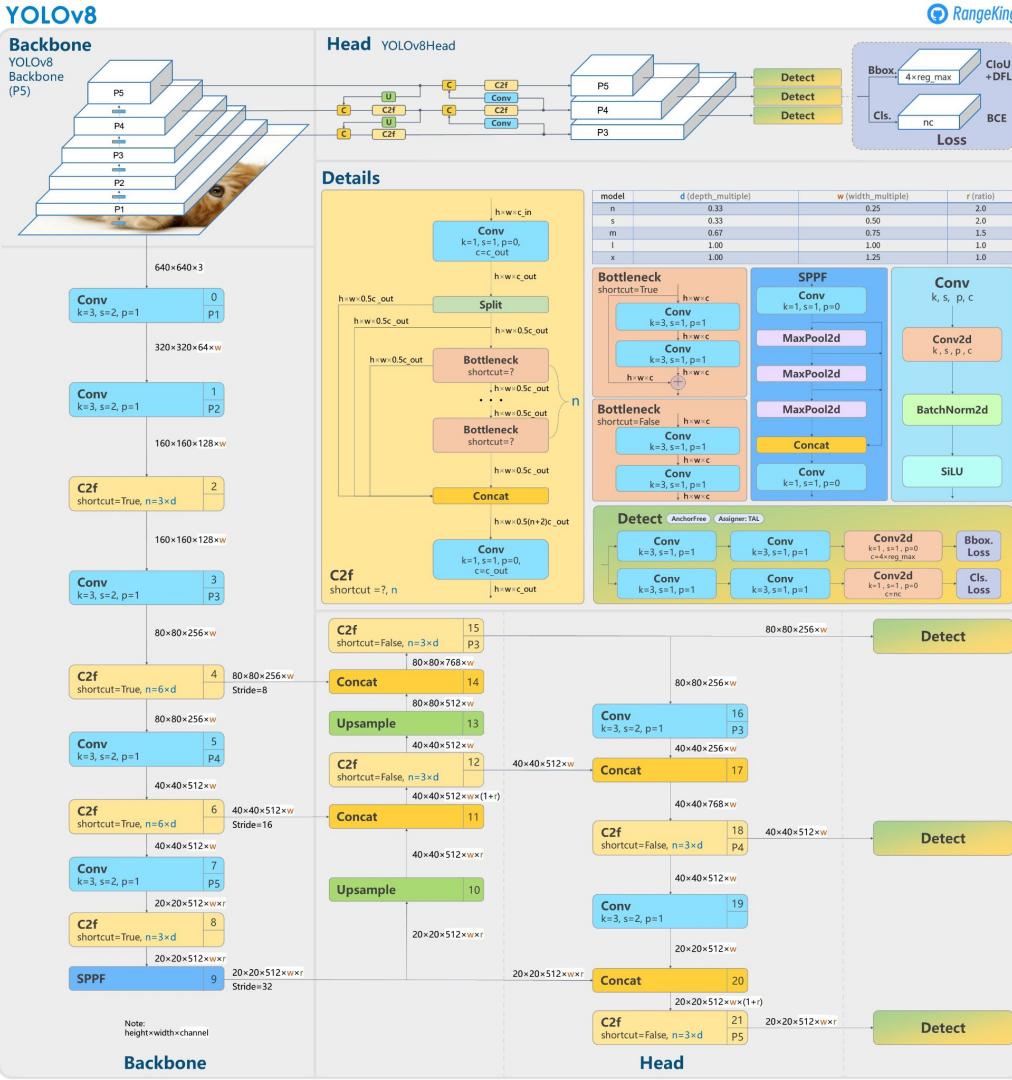


Source: YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for
real-time object detectors
<https://arxiv.org/pdf/2207.02696>

1. Во время обучения к промежуточным слоям добавляются вспомогательные головы. Это помогает лучше распространять градиенты по сети.

2. Дополнительные головы также участвуют в обучении. Для них назначаются более “мягкие” маски, то есть больше ячеек будет считаться положительными. Так ей будет проще обучаться.
 Для основной головы критерии такие же строгие как раньше.

YOLOv8.



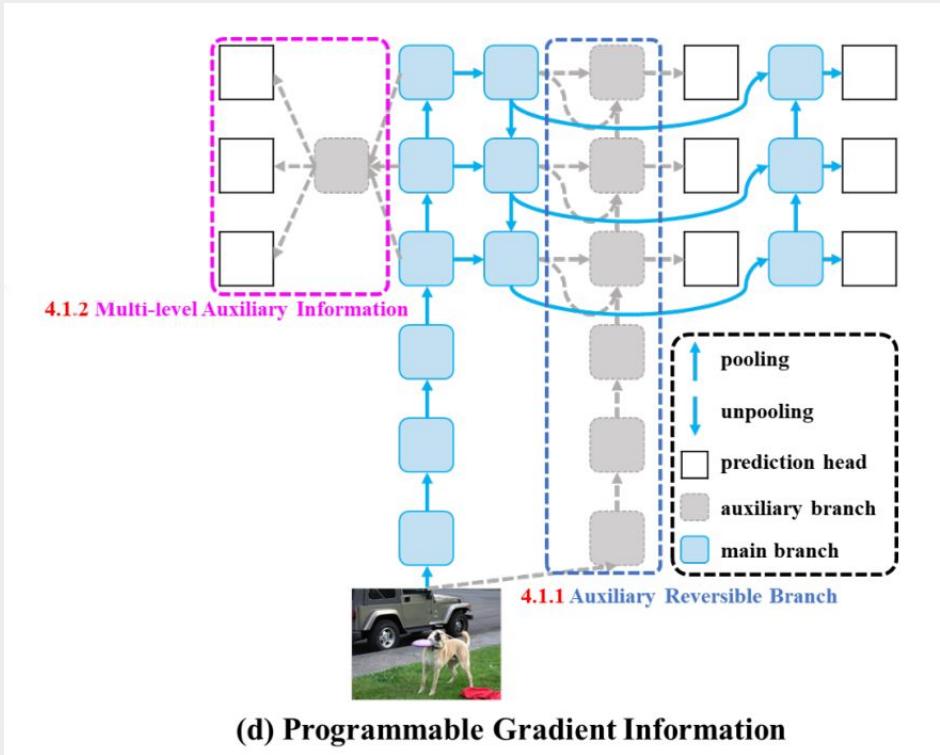
YOLOv8. Сквозь модели.

YOLOv8 anchor-free архитектура, которую адаптировали под много задач.

Ultralytics, авторы архитектуры расширили свою экосистему, и с помощью YOLOv8 можно было удобно решать много задач.

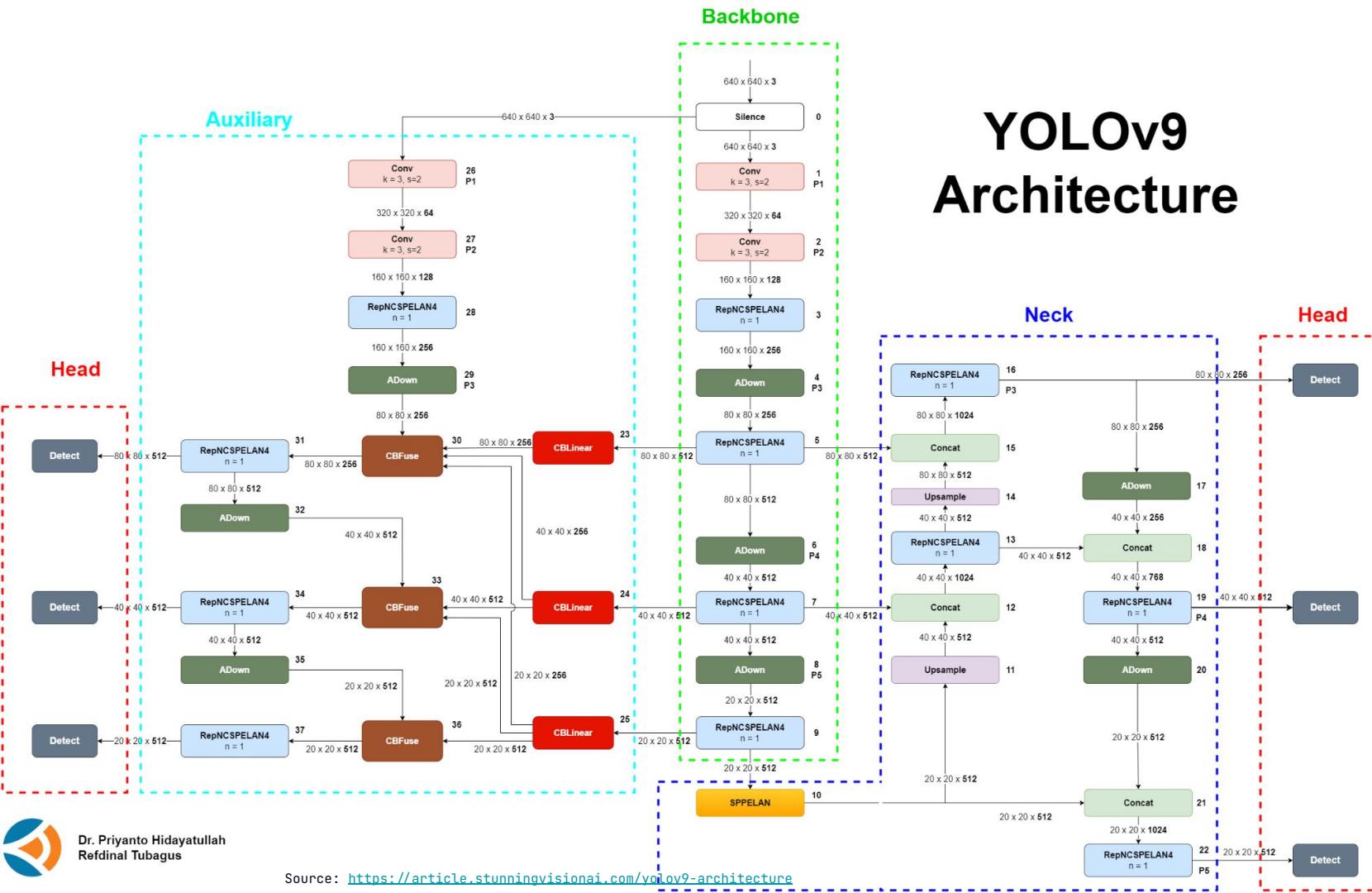
Model	Filenames		Task
YOLOv8	yolov8n.pt yolov8m.pt yolov8x.pt yolov8s.pt yolov8l.pt		Detection
YOLOv8-seg	yolov8n-seg.pt yolov8m-seg.pt yolov8x-seg.pt yolov8s-seg.pt yolov8l-seg.pt		Instance Segmentation
YOLOv8-pose	yolov8n-pose.pt yolov8m-pose.pt yolov8x-pose.pt yolov8s-pose.pt yolov8l-pose.pt yolov8x-pose-p6.pt		Pose/Keypoints
YOLOv8-obb	yolov8n-obb.pt yolov8m-obb.pt yolov8x-obb.pt yolov8s-obb.pt yolov8l-obb.pt		Oriented Detection
YOLOv8-cls	yolov8n-cls.pt yolov8m-cls.pt yolov8x-cls.pt yolov8s-cls.pt yolov8l-cls.pt		Classification

YOLOv9. Сквозь модели.

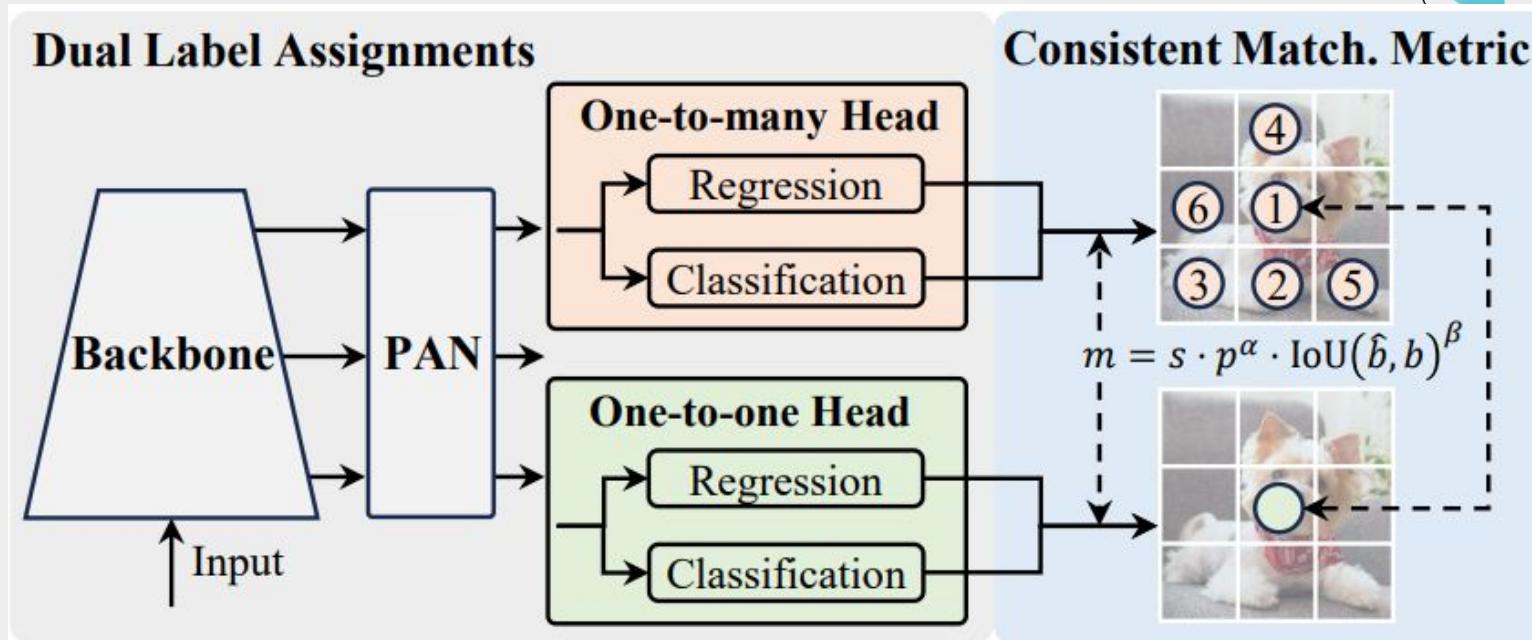


Source: <https://arxiv.org/pdf/2402.13616.pdf>

YOLOv9 Architecture

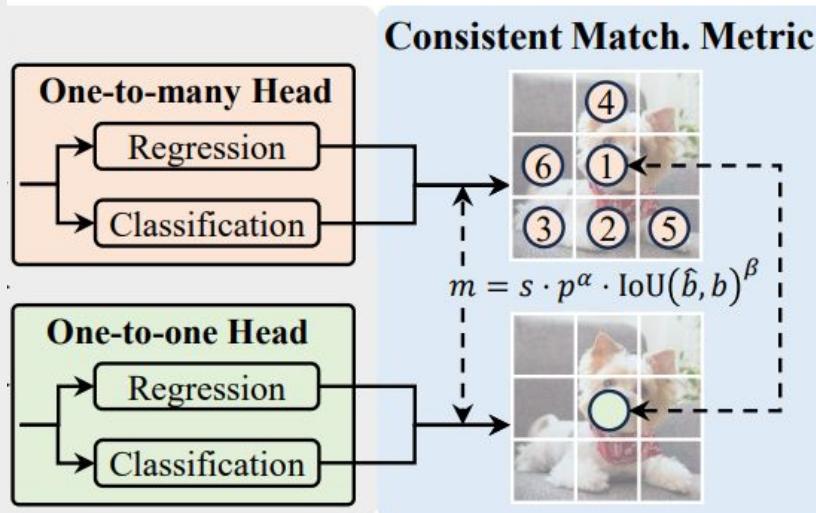


YOLOv10. Сквозь модели.



Source: YOLOv10: Real-Time End-to-End Object Detection
<https://arxiv.org/pdf/2405.14458.pdf>

YOLOv10. Сквозь модели.

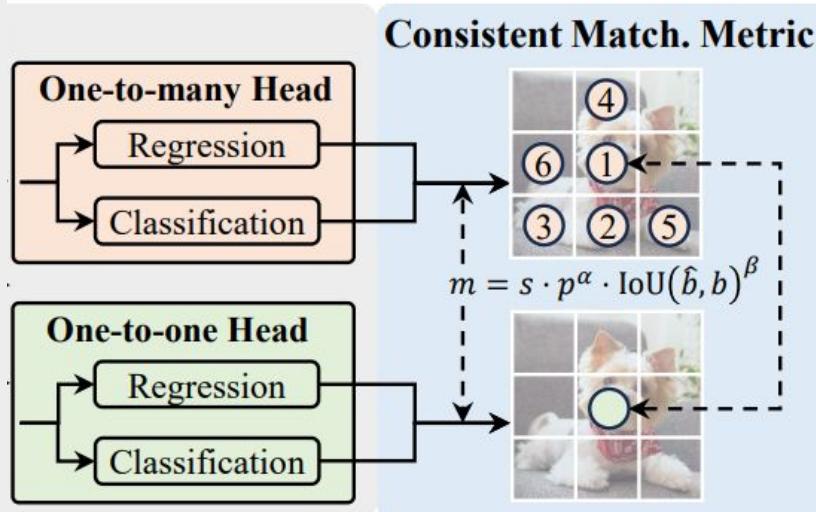


One-to-Many:

Вспомогательная голова, которая использует большое число кандидатов (положительных боксов) в процессе обучения.

Таким образом модель получает больше информации для прорыва градиентов, что помогает ей лучше обучаться.

YOLOv10. Сквозь модели.



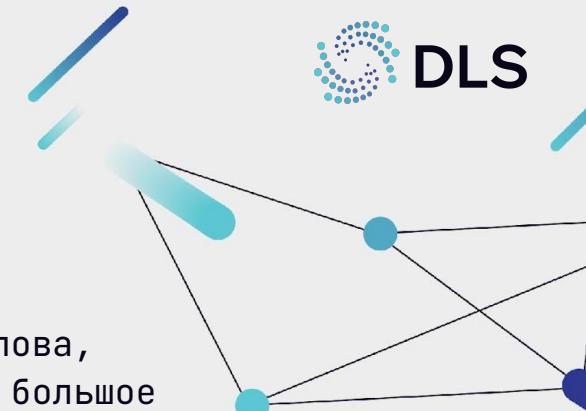
One-to-Many:

Вспомогательная голова, которая использует большое число кандидатов (позитивных блоков) в процессе обучения.

One-to-one:

Основная голова, отвечающая за предсказания на inference. Выдает только 1 предсказание для каждого объекта.

В процессе обучения, задача label assignment решается с помощью Венгерского алгоритма (Hungarian algorithm).



YOLOv10. Сквозь модели.

Венгерский алгоритм *in a nutshell*:

Строим матрицу стоимостей для предсказаний и GT
(IoU + class score)

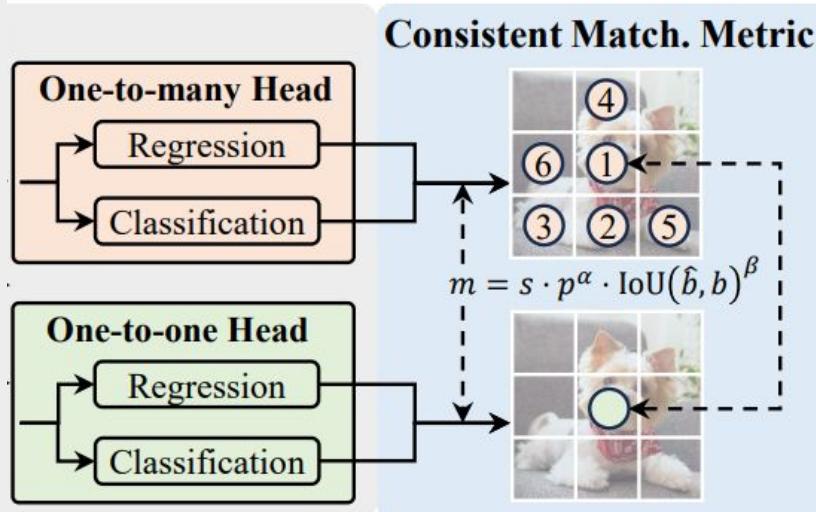
Дальше минимизируем общую “стоимость” сопоставления GT и предсказания.

В итоге, каждому GT назначается **строго одно** предсказание.

Минус – алгоритм достаточно медленный $O(n^3)$.



YOLOv10. Сквозь модели.



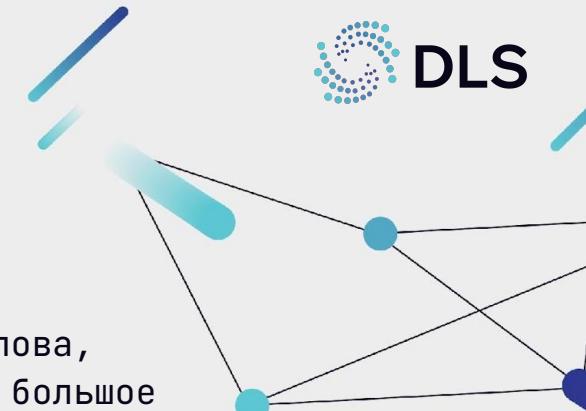
One-to-Many:

Вспомогательная голова, которая использует большое число кандидатов (позитивных блоков) в процессе обучения.

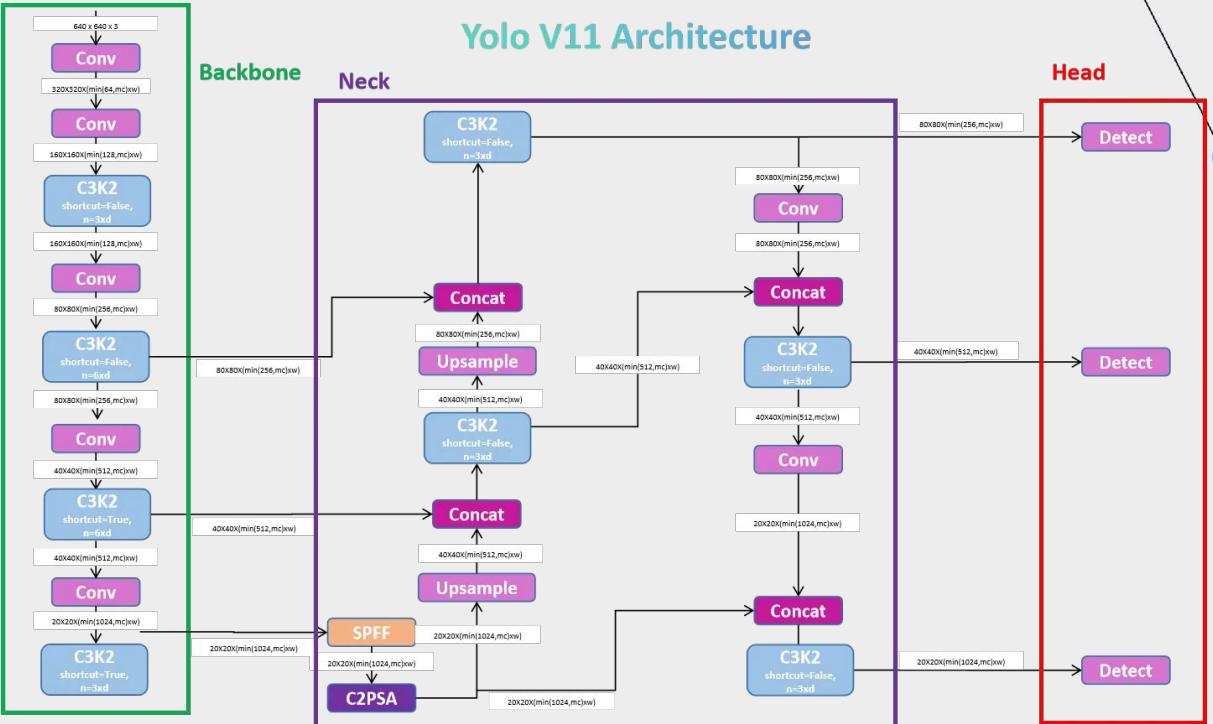
One-to-one:

Основная голова, отвечающая за предсказания на inference. Выдает только 1 предсказание для каждого объекта.

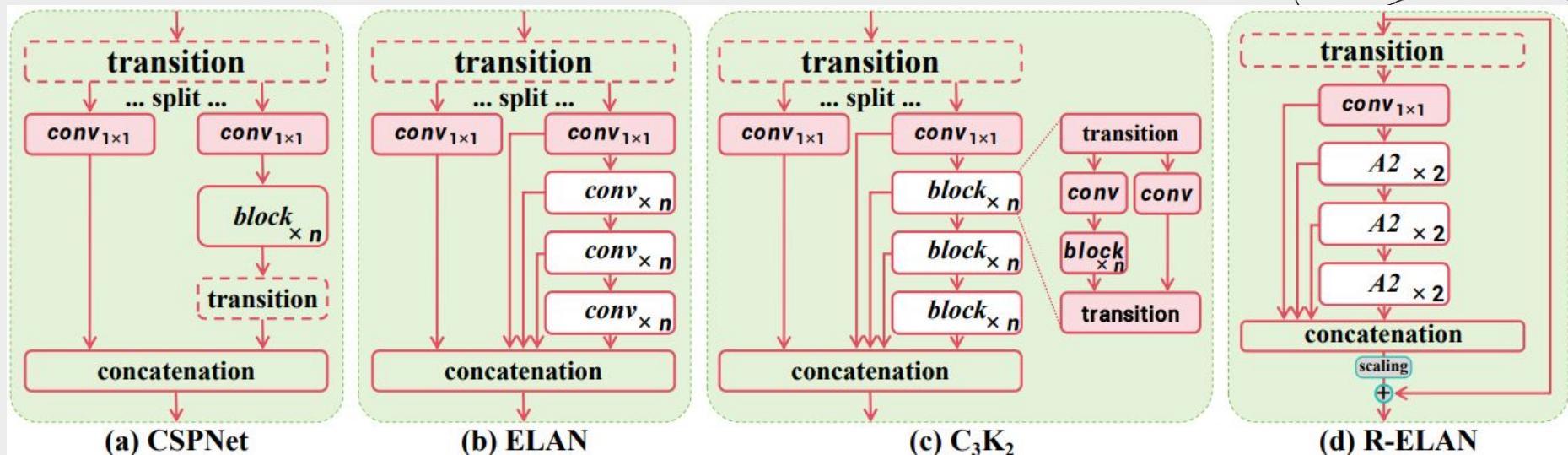
В процессе обучения, задача label assignment решается с помощью Венгерского алгоритма (Hungarian algorithm).



YOLOv11. Сквозь модели.



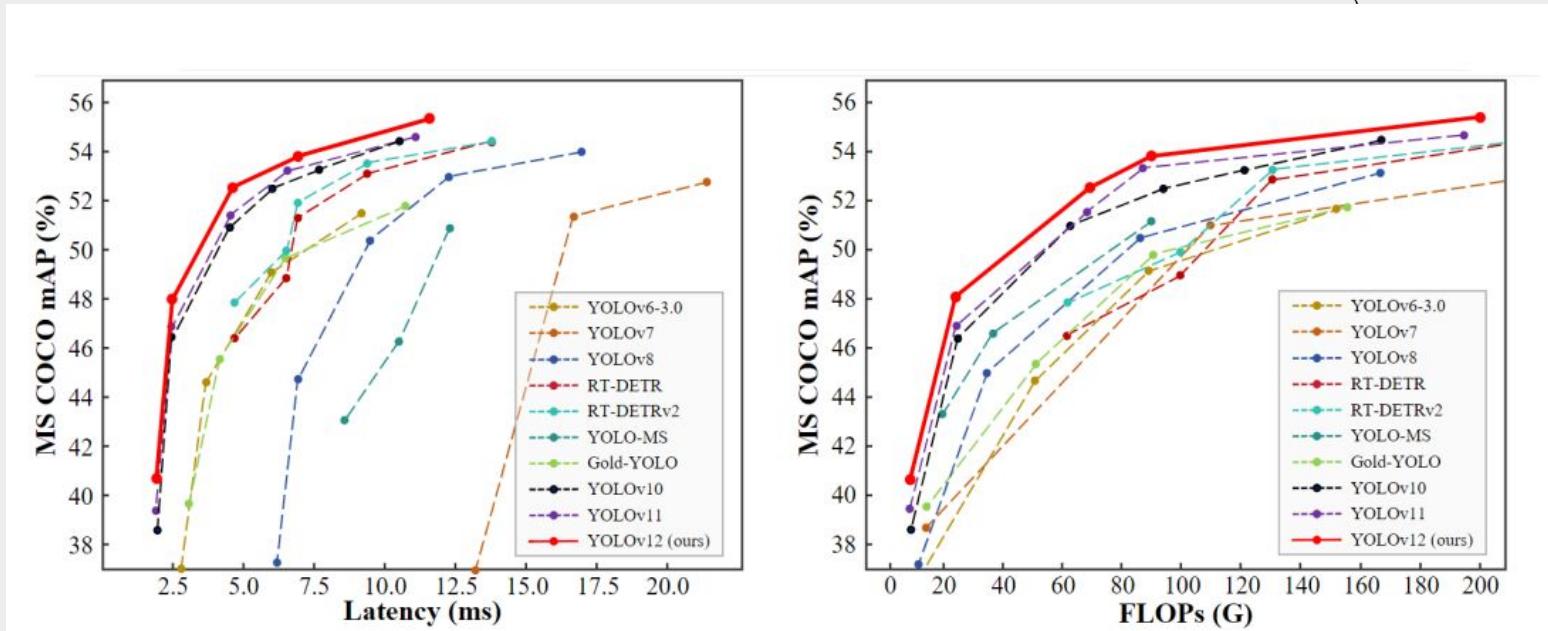
YOLOv12. Сквозь модели.



Основные блоки для предыдущих YOLO моделей.

Блок для YOLOv12
A2 – FlashAttention.

YOLOv12. Сквозь модели.



Семейство YOLO

Версия	Использует якоря?	Backbone	Neck	Additional features
4	Да	CSPDarknet53	PAN + SPP	CIoU NMS
5	Да	CSPDarknet53 (C3)	PAN	Удобный, быстрый, популярный
6	Нет	EfficientRep	RepPAN	TAL + Reparameterization Trick + VariFocal Loss
7	Да	E-Elan based	E-Elan PAN	Auxiliary head + coarse-to-fine assigner
8	Нет	CSPDarknet53 (C2f)	PAN	Удобный, быстрый, популярный + multitask
9	Нет	CSP-ELAN based	CSP-ELAN PAN	Programmable Gradient Information
10	Нет	YOLOv8 as baseline		НЕ использует NMS на Inference
11	Нет	DarkNet53 (C3K2)	PAN	Удобный, быстрый, популярный + multitask
12	Нет	R-ELAN + Area Attention (A2)	PAN	Первая номерная часть использующая трансформеры

Нестандартные подходы



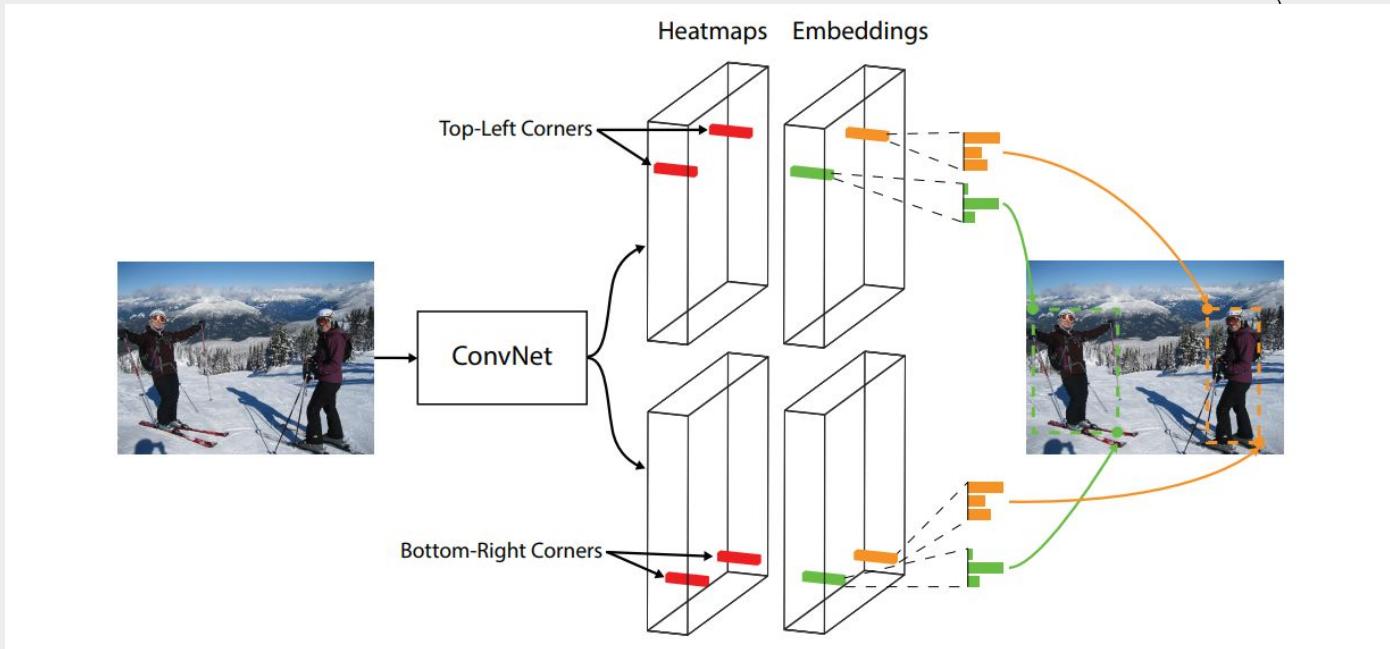
CornerNet

Подход, основанный на предсказании верхнего левого и правого нижнего краёв bounding box'а.



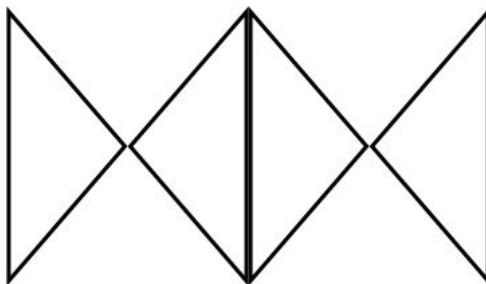
CornerNet

Подход, основанный на предсказании верхнего левого и правого нижнего краёв bounding box'а.



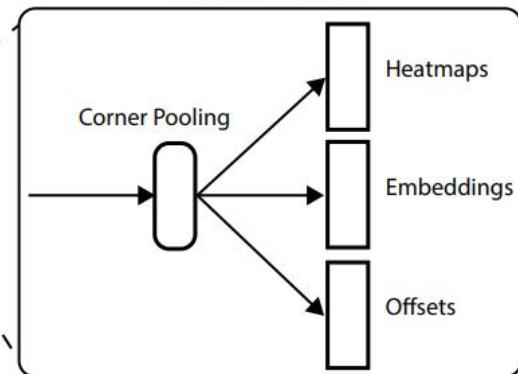
Source: CornerNet: Detecting Objects as Paired Keypoints
<https://arxiv.org/abs/1808.01244>

CornerNet



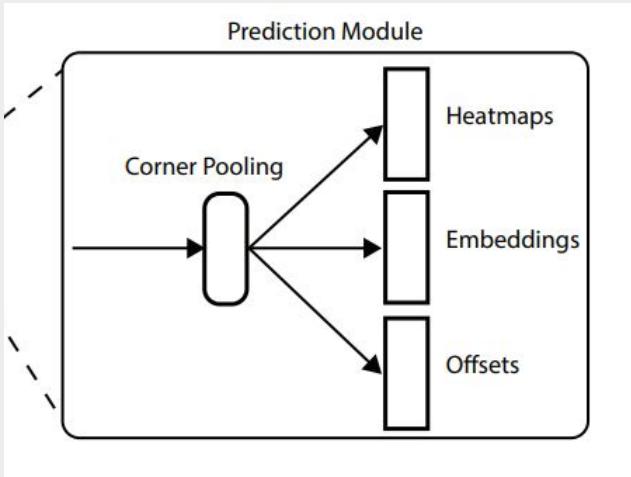
Hourglass Network

Top-left Corners
Prediction Module
Bottom-right corners
Prediction Module



- Backbone – Два encoder-decoder блоков подряд,
- Head – Два идентичных и независимых блоков для предсказания левого верхнего и правого нижнего углов.
- Выход – 3 группы feature map'ов.

CornerNet. Output.



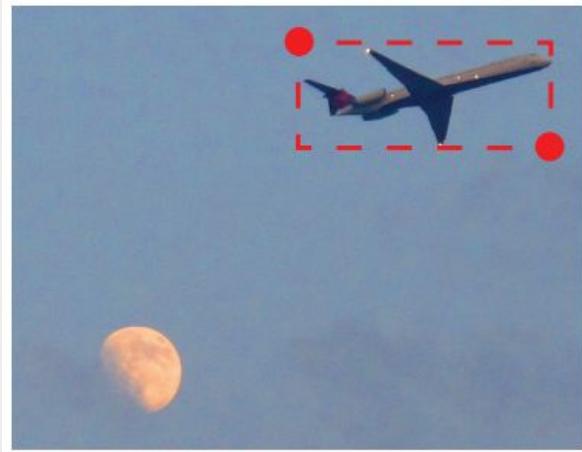
Heatmaps – Feature map в котором предсказывается вероятность наличия левого верхнего **TL** (или правого нижнего **BR**) угла конкретного Класса (Размер $H' \times W' \times C$)

Embeddings – Позиция i, j содержит числовой эмбеддинг для угла в позиции i, j . Близкие по расстоянию эмбеддинги для углов **TL** и **BR** означает что это углы одного бокса. Углы разных классов имеют одинаковые боксы. Размер эмбеддинга – 128. (Размер $H' \times W' \times 128$)

Offsets – Дополнительное смещение угла, чтобы восстановить положение углов на исходном изображении, тк размеры выходного feature map'a могут не совпадать с исходными. (Размер $H' \times W' \times C$)



CornerNet. Corner Pooling.



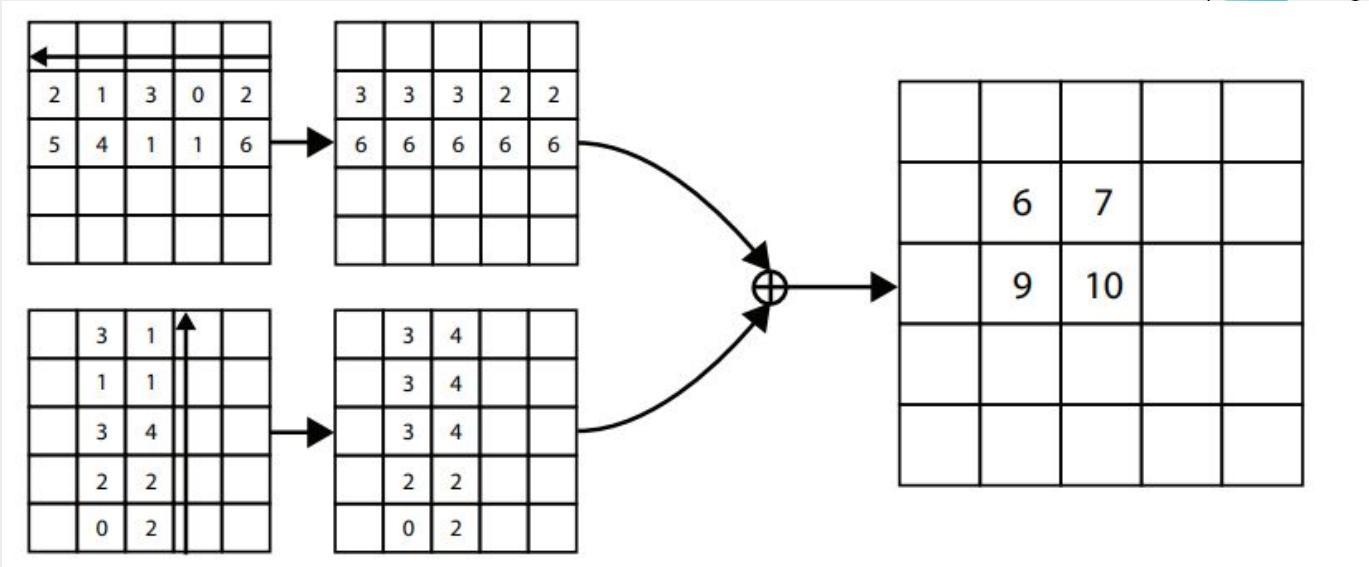
Есть объекты, углы которых никак не взаимодействуют с объектом, тогда модели сложно учиться, так как вокруг угла нет визуальной информации за которую можно ухватиться.

Corner Pooling пытается решить эту проблему добавляя больше контекста.

Например, чтобы обнаружить угол **TL**, то есть крайний пиксель объекта, полезно знать есть ли признаки объекта “сверху” и “слева” от данной точки.



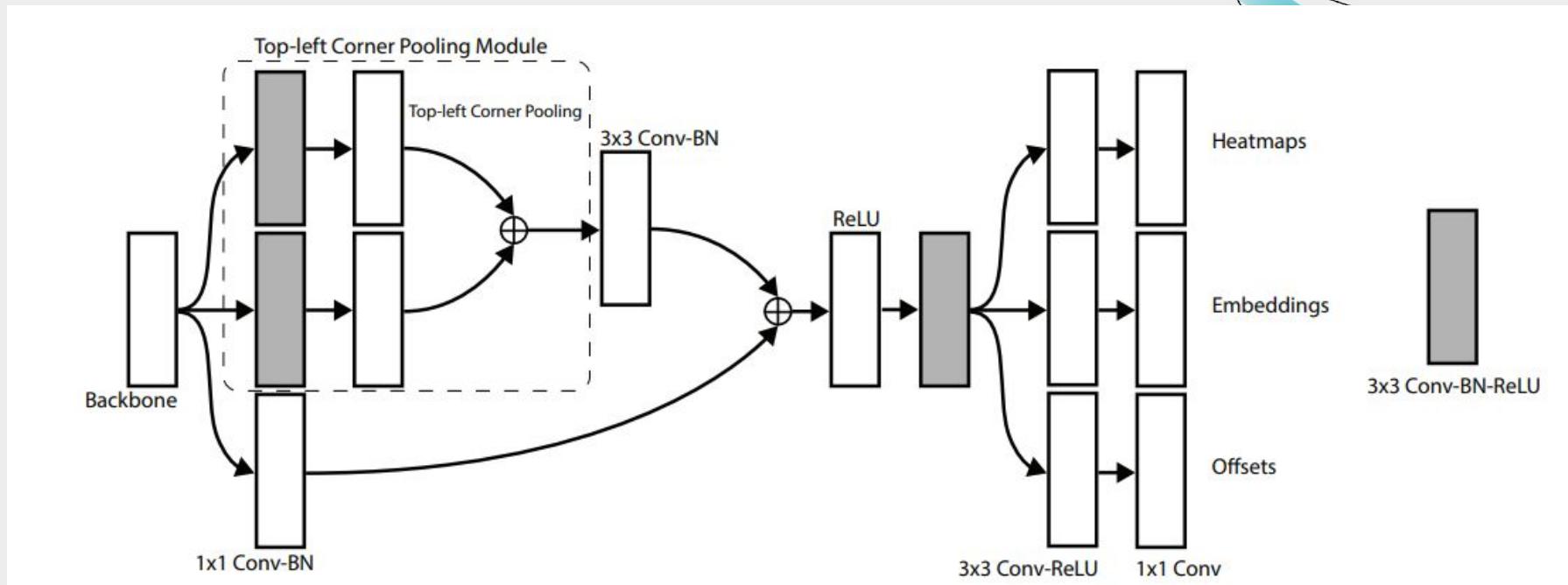
CornerNet. Corner Pooling.



Corner Pooling состоит из 2x параллельных max-pooling слоев, которые находят максимальные значения в двух направлениях, независимо для каждой feature map'ы.

Затем результаты двух max-pooling операций складываются.

CornerNet. Head.



CornerNet. Training.

Лосс для обучения состоит из 4x частей:

$$L = L_{det} + \alpha L_{pull} + \beta L_{push} + \gamma L_{off}$$

Где,

L_{det} – детекционный лосс,

L_{pull}, L_{push} – лосс эмбеддинга,

L_{off} – offset лосс.

Нормировочные константы равны 0.1, 0.1 и 1 соответственно.

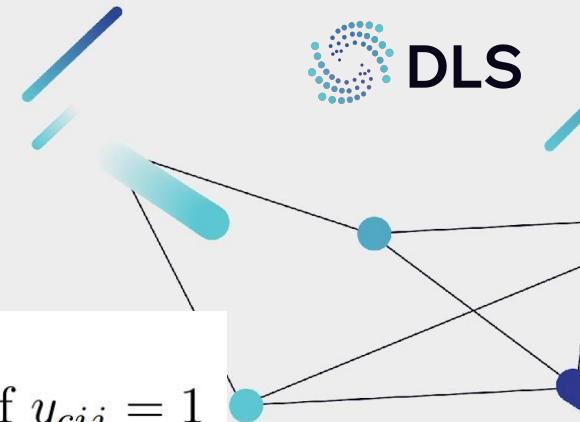


CornerNet. Training.

Лосс детекции – измененный Focal loss:

$$L_{det} = \frac{-1}{N} \sum_{c=1}^C \sum_{i=1}^H \sum_{j=1}^W \left\{ \begin{array}{ll} (1 - p_{cij})^\alpha \log(p_{cij}) & \text{if } y_{cij} = 1 \\ (1 - y_{cij})^\beta (p_{cij})^\alpha \log(1 - p_{cij}) & \text{otherwise} \end{array} \right.$$

Основная идея – точки, близкие к истинному углу не должны иметь нулевой лейбл, а должны уменьшаться по гауссиане.
(P_{cij} – скор в точке i, j для класса c)



CornerNet. Training.

Основная идея – точки, близкие к истинному углу не должны иметь нулевой лейбл, а должны уменьшаться по гауссиане.



CornerNet. Training.

Двухсоставной лосс для эмбеддингов:

$$L_{pull} = \frac{1}{N} \sum_{k=1}^N \left[(e_{t_k} - e_k)^2 + (e_{b_k} - e_k)^2 \right],$$

$$L_{push} = \frac{1}{N(N-1)} \sum_{k=1}^N \sum_{\substack{j=1 \\ j \neq k}}^N \max(0, \Delta - |e_k - e_j|)$$

Где, e_{t_k} это эмбеддинг, левого верхнего угла для объекта k , а e_{b_k} – эмбеддинг правого нижнего угла.

e_k – среднее между e_{t_k} и e_{b_k} .

Δ – константа, равная 1.

Идея: Сближать эмбеддинги левого и правого углов одного объекта и разделять средние эмбеддинги углов разных объектов.



CornerNet. Training.

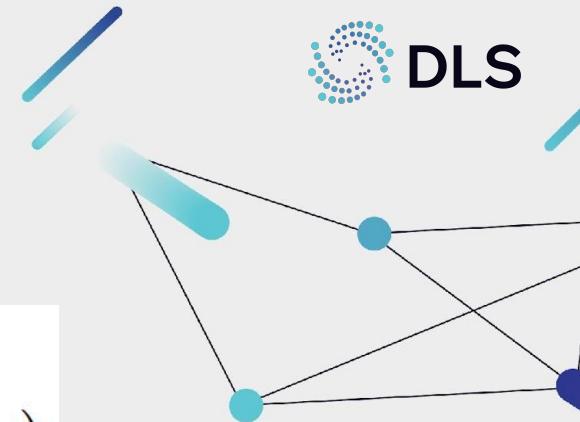
Для отклонений используется уже знакомый smoothL1.

$$L_{off} = \frac{1}{N} \sum_{k=1}^N \text{SmoothL1Loss} (\mathbf{o}_k, \hat{\mathbf{o}}_k)$$

$$\mathbf{o}_k = \left(\frac{x_k}{n} - \left\lfloor \frac{x_k}{n} \right\rfloor, \frac{y_k}{n} - \left\lfloor \frac{y_k}{n} \right\rfloor \right)$$

Где \mathbf{o}_k это offset для объекта k , а n – downsampling factor или отношение размера исходной картинки к размеру feature map.

Идея: Предсказываем потерю при округлении, чтобы обеспечить точное позиционирование углов.



CornerNet. Итоги.

Авторы представили необычный подход к решению задачи детекции, при этом показывающий хорошую точность.

Плюсы:

- Нет зависимости от anchor boxes, размеры ббоксов выбираются автоматически,
- Благодаря смещениям и corner pooling может достаточно точно определять углы объектов,

Минусы:

- Чувствительна к нечетким границам объектов, например если объект частично перекрыт.
- Модель вообще никак не учитывает информацию внутри ббокса, что повышает количество ложных срабатываний.

Качество: 42.2 mAP COCO



CenterNet

Идейное продолжение CornerNet'а.

Наблюдение: Проанализировав False Discovery rate ($FD = 1 - mAP$) авторы поняли что CornerNet допускает слишком много ложных срабатываний.



CenterNet

Идейное продолжение CornerNet'а.

Наблюдение: Проанализировав False Discovery rate ($FD = 1 - mAP$) авторы поняли что CornerNet допускает слишком много ложных срабатываний.

Вывод: Корректное сопоставление углов (левого верхнего и правого нижнего) это сложная задача, особенно когда объекты находятся близко друг к другу или частично перекрываются. Эмбеддинги помогают, но ошибка группировки все равно оставалась.



CenterNet

Идейное продолжение CornerNet'а.

Наблюдение: Проанализировав False Discovery rate ($FD = 1 - mAP$) авторы поняли что CornerNet допускает слишком много ложных срабатываний.

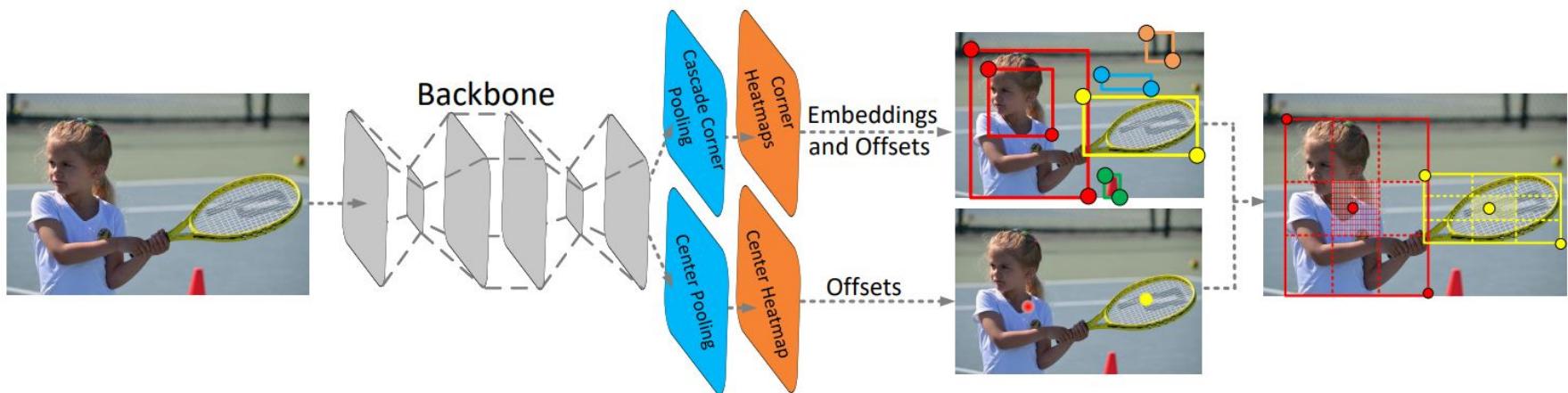
Вывод: Корректное сопоставление углов (левого верхнего и правого нижнего) это сложная задача, особенно когда объекты находятся близко друг к другу или частично перекрываются. Эмбеддинги помогают, но ошибка группировки все равно оставалась.

Предложенное решение: Добавим центр объекта. Так как он определяется легче и может служить опорной точкой для последующей регрессии размеров и коррекции положения.



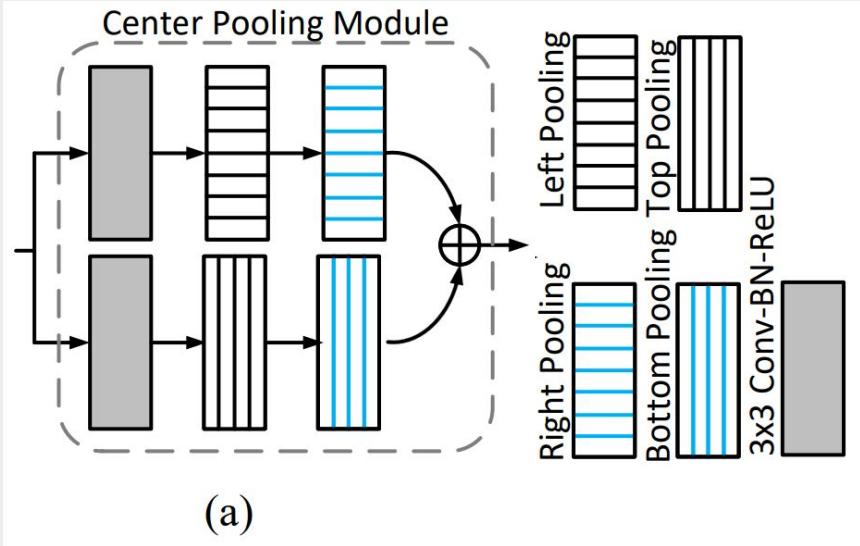
CenterNet

Предложенная архитектура:

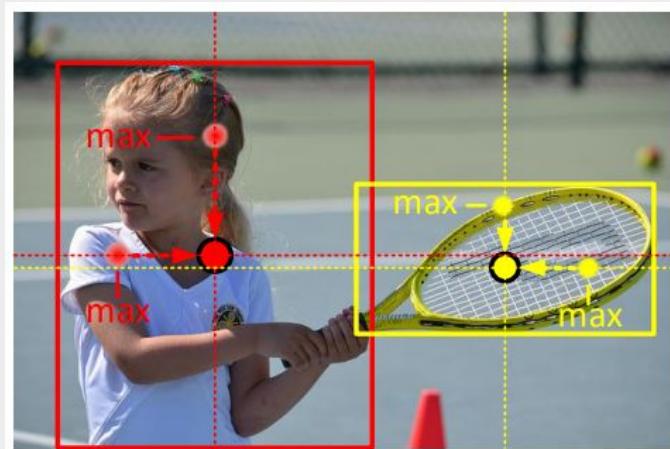


Source: CenterNet: Keypoint Triplets for Object Detection
<https://arxiv.org/abs/1904.08189>

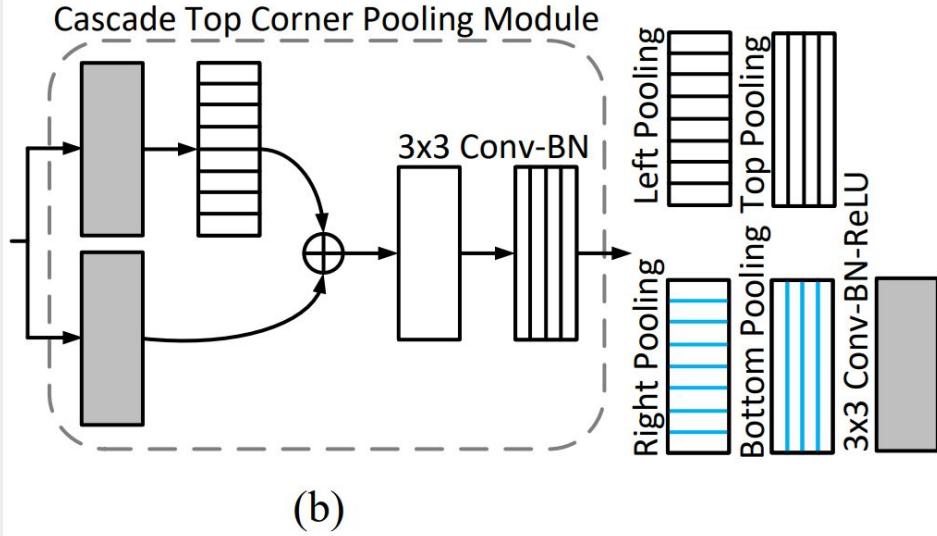
CenterNet. Center Pooling.



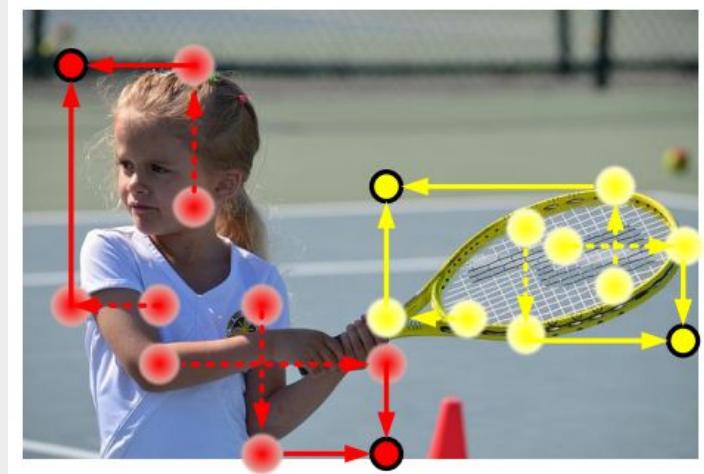
Находим максимальные значения признаков по вертикали и горизонтали, тем самым улучшая выделение центра объекта.



Cascade Top Pooling



Добавление конволовионных слоев позволило учитывать не только границы объектов, но и опираться на значения по всему пространству признаков.



CenterNet. Output.

В итоге у модели будут следующие выходные слои:

1. Center Heatmap ($H' \times W' \times C$):

Для каждого пикселя предсказывается вероятность того, что в данном месте находится **центр объекта** определенного класса.

2. Top-Left (TL) Heatmap ($H' \times W' \times C$):

Вероятность того, что в конкретном пикселе находится **верхний левый угол** объекта определенного класса.

3. Bottom-Right (BR) Heatmap ($H' \times W' \times C$):

Вероятность того, что в конкретном пикселе находится **правый нижний угол** объекта определенного класса.

4. Embeddings ($H' \times W' \times 128$):

5. Offset map для центра ($H' \times W' \times C$):

Смещение для центра ббокса.

6. (Опционально) Offset map для углов ($H' \times W' \times C$):

Смещение для TL и BR.



CenterNet. Training.

В целом повторяет процесс обучения CornerNet.

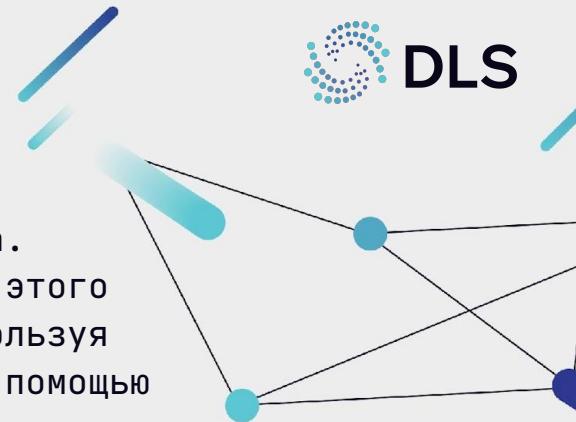
Также как и углы, heatmap для центров не является бинарной маской. Вокруг каждого центра объекта задается область в которой точки считаются центральными. Размер области зависит от размера объекта.

Важно: значение класса для предсказанного ббокса учитывается сразу в 3x heatmap'ах (TL, BR и Cetner). Чтобы ббокс считался правильно предсказанным, классы должны совпадать.

Итоговый лосс:

- Focal loss для heatmap'ов,
- Smooth 1L для offsets,
- CornerNet Embedding Loss для эмбеддингов.

CenterNet. Итоги.

- 
1. В итоге удалось получить более точную версию CornerNet'а.
 2. Авторы представили multi-scale вариант CenterNet'а. Для этого они прогоняли через модель одно и то же изображение используя маштабирование ($0.5 - 2$), затем соединяя предсказания с помощью NMS.
 3. Такую архитектуру можно относительно легко расширять до других задач типа Instance segmentation и других.

Качество: **47 mAP** на COCO против **42.2** у CornerNet.

CenterNet. Фактичек.

Существует ещё одна модель с названием CenterNet, которая вышла одновременно с той, которую мы только что разобрали:

1. CenterNet: Keypoint Triplets for Object Detection (17 апреля 2019)
2. Objects as Points (2 апреля 2019)

Они обе были на ICCV 2019 и обе стали очень известными (3800 и 4600 цитирований).

Не перепутайте:)

Orientated Bounding Boxes

Расширение задачи object detection, позволяющие предсказывать наклонные боксы. Обычно используется в задачах со спутниковыми снимками.



Orientated Bounding Boxes

Расширение задачи object detection, позволяющие предсказывать наклонные боксы.

У нас появляется новый параметр – θ обозначающий угол поворота ббокса.

1. As regression task

Может обучаться регрессионными лоссами, но это нестабильно, тк (θ и 180° тут обозначают одно и то же)

2. Regression improvement

Вместо угла предсказываем $\sin(\theta)$ и $\cos(\theta)$, а затем считаем $\theta = \text{atan2}(\sin(\theta), \cos(\theta))$.

3. Classification for 180 classes or less

4. Modulo loss (лосс, который может учитывает периодичность угла)

Другие способы label assignment'a



Label Assignment starts

Процедура совмещения якорей (или предсказаний) с истинными ббоксами одна из важнейших составляющих процесса обучения детекторов. В рамках лекций мы не рассказали о многих методах, коротко поговорим о некоторых из них:

1. ATSS
2. PAA
3. OTA и SimOTA

Label Assignment starts

Процедура совмещения якорей (или предсказаний) с истинными боксами одна из важнейших составляющих процесса обучения детекторов. В рамках лекций мы не рассказали о многих методах, коротко поговорим о некоторых из них:

1. ATSS ([Adaptive Training Sample Selection](#))

Для каждого предсказания отдельно подбираем порог IoU на основе **распределения IoU** между GT и K ближайшими боксами вокруг него.

2. PAA

3. OTA и SimOTA



Label Assignment starts

Процедура совмещения якорей (или предсказаний) с истинными ббоксами одна из важнейших составляющих процесса обучения детекторов. В рамках лекций мы не рассказали о многих методах, коротко поговорим о некоторых из них:

1. ATSS
2. PAA ([Probabilistic Anchor Assignment](#))

Вообще не имеет параметров. Считаем распределение IoU для одного GT и всех предсказаний (или якорей). Затем кластеризуем распределение на 2 класса “положительные” и “отрицательные” предсказания.

3. OTA и SimOTA



Label Assignment starts

Процедура совмещения якорей (или предсказаний) с истинными ббоксами одна из важнейших составляющих процесса обучения детекторов. В рамках лекций мы не рассказали о многих методах, коротко поговорим о некоторых из них:

1. ATSS
2. PAA
3. OTA и SimOTA (Optimal Transport Assignment Problem)

Основаны на **теории оптимального транспорта** для поиска наилучшего сопоставления между предсказаниями и GT. Матрица стоимости строится на основе IoU и class score.

SimOTA – использует жадный отбор предсказаний на основе IoU + class score.



Заключение

В рамках лекции мы:

1. Узнали о том что такая задача детекции,
1. Познакомились с базовыми концептами этой задачи,
 - a. Anchor / Bounding box
 - b. NMS, IoU
 - c. Label assignment
 - d. И другие
2. Подробно поговорили о моделях детекции.

