# Yolov1 (2015)

* обнаружение объектов как задачу регрессии к пространственно разделенным ограничивающим рамкам и связанным вероятностям классов
* Единственная нейронная сеть предсказывает ограничивающие рамки (bounding boxes) и вероятности классов непосредственно из полных изображений за одно вычисление
* ~~Основан на R-CNN алгоритмах~~ Часто сравнивали с R-CNN
* ~~Основан на DarkNet: “We use the Darknet framework for all training and inference”~~
* Основан на GoogleNet
* Хотя он может быстро идентифицировать объекты на изображениях, ему сложно точно локализовать некоторые объекты, особенно маленькие.
* At test time we multiply the conditional class probabilities and the individual box confidence predictions,

(Во время тестирования мы умножаем вероятности условного класса и прогнозы достоверности отдельных блоков)

Pr(Classi|Object) ∗ Pr(Object) ∗ IOUtruth pred = Pr(Classi) ∗ IOUtruth pred (1)

which gives us class-specific confidence scores for each box. These scores encode both the probability of that class appearing in the box and how well the predicted box fits the object

(что дает нам оценки достоверности для каждого класса для каждого блока. Эти оценки кодируют как вероятность появления этого класса в блоке, так и то, насколько хорошо предсказанный блок соответствует объекту)

* Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an S × S grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an S × S × (B ∗ 5 + C) tensor.

Изображение выглядит как снимок экрана, прямоугольный, мозаика

Автоматически созданное описание

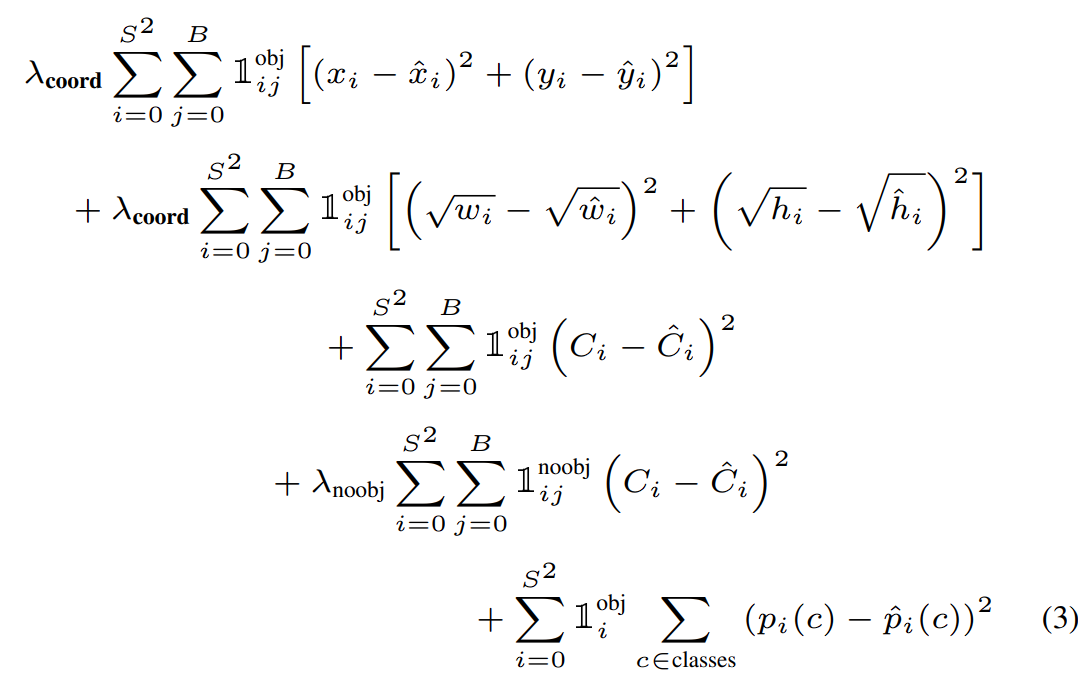
For evaluating YOLO on PASCAL VOC, we use S = 7, B = 2. PASCAL VOC has 20 labelled classes so C = 20. Our final prediction is a 7 × 7 × 30 tensor.

* 2.1. Network Design
* We implement this model as a convolutional neural network and evaluate it on the PASCAL VOC detection dataset [9]. The initial convolutional layers of the network extract features from the image while the fully connected layers predict the output probabilities and coordinates. Our network architecture is inspired by the GoogLeNet model for image classification [34]. Our network has 24 convolutional layers followed by 2 fully connected layers. Instead of the inception modules used by GoogLeNet, we simply use 1 × 1 reduction layers followed by 3 × 3 convolutional layers, similar to Lin et al [22]. The full network is shown in Figure 3.
* Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1 × 1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224 × 224 input image) and then double the resolution for detection.

Изображение выглядит как текст, зарисовка, диаграмма, Шрифт

Автоматически созданное описание

* loss function:



Обратите внимание, что функция потерь наказывает ошибку классификации только в том случае, если объект присутствует в этой ячейке сетки (отсюда и вероятность условного класса, обсуждавшаяся ранее). Он также наказывает ошибку координат ограничивающего прямоугольника только в том случае, если этот предиктор «ответственен» за основной блок истинности (т. е. имеет самый высокий IOU среди всех предсказателей в этой ячейке сетки).

* YOLO имеет некоторое сходство с R-CNN. Каждая ячейка сетки предлагает потенциальные ограничивающие рамки и оценивает эти рамки с использованием сверточных функций. Однако наша система накладывает пространственные ограничения на предлагаемые ячейки сетки, что помогает снизить вероятность многократного обнаружения одного и того же объекта. Наша система также предлагает гораздо меньше ограничивающих рамок: всего 98 на изображение по сравнению с примерно 2000 при выборочном поиске. Наконец, наша система объединяет эти отдельные компоненты в единую, совместно оптимизированную модель.

# Yolov2 (2016)

* Убрали регресию
* улучшила оригинальную модель, включив в нее batch нормализацию, якорные рамки (anchor boxes) и кластеры размерностей
* Появилась возможность загружать изображения 448\*448, а не 224\*224
* Использует нейронные сети с более глубокими слоями и различными размерами ядер для лучшего обнаружения объектов различных размеров.
* превосходя самые современные методы, такие как более быстрый RCNN с ResNet и SSD, при этом все еще работая значительно быстрее
* Модель DarkNet-19: 19 convolutional layers and 5 maxpooling layers
* обучен работе с наборами данных обнаружения, такими как **[Pascal VOC](http://host.robots.ox.ac.uk/pascal/VOC/)** и [**MS COCO**](https://cocodataset.org/#home)
* ~~Обнаруживает более 9тыс категорий объектов~~

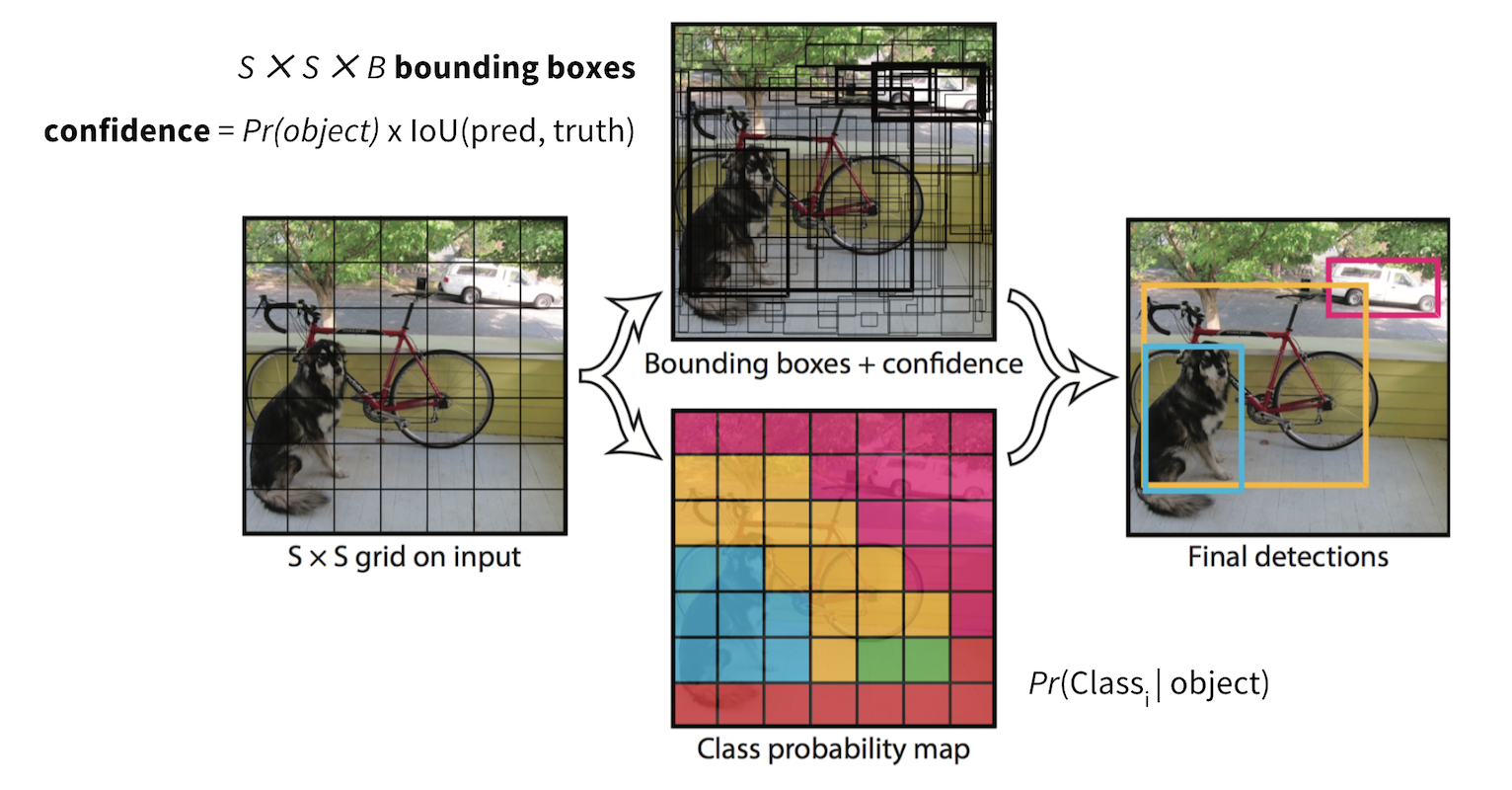
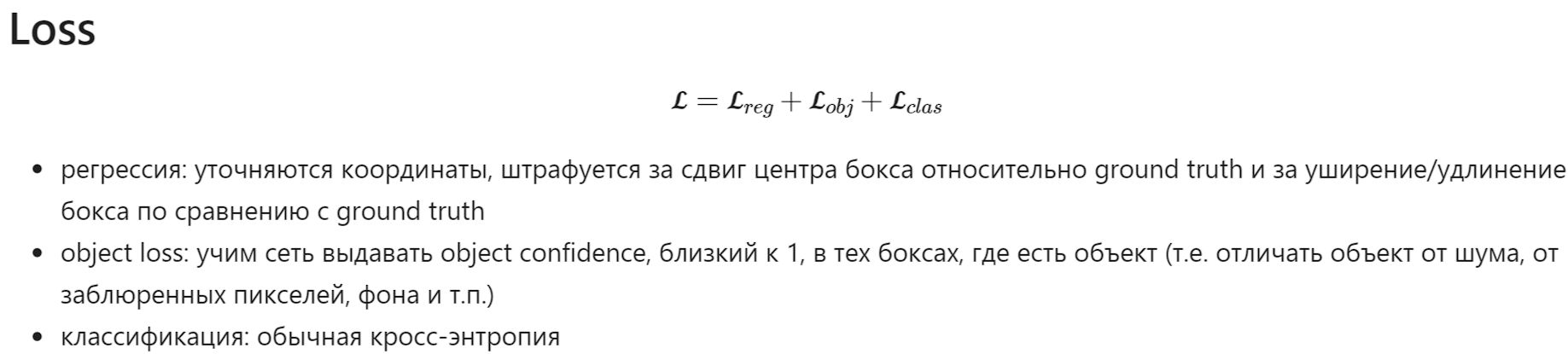
# Yolov3 (2018)

* больше повысила производительность модели за счет использования более эффективной опорной сети, множества якорей и объединения пространственных пирамид.
* Использует сложные блоки (residual blocks) и Darknet-53 для улучшения производительности и точности.
* Вводит возможность обнаружения объектов различных масштабов с использованием нескольких слоев на разных уровнях сети. То есть разные скейлы
* Улучшенная работа с anchor boxes
* Вместо классического Feature Pyramid Network используют Path Aggreagtion Network (более хитрое смешивание фичей с разных масштабов)
* Многомасштабное обнаружение – появилось 3 головы
* Использования k-means для anchor boxes
* Появился Non-Maximum Supression

# Yolov4 (2020)

* Повились mosaic data аугментация, новая головка обнаружения без привязки и новая функция потерь.
* Объединение функций:
  + Weighted-Residual-Connections (WRC)
  + Cross Stage Partial connections (CSP)
  + Cross mini-Batch Normalization (CmBN)
  + Self-adversarial training (SAT)
  + Mish activation
  + Mosaic data augmentation
  + DropBlock regularization
  + CIoU – новая метрика
  + lossCIoU– новая функция потерь

# Yolov5 (2020)

* улучшена производительность модели и добавлены новые функции, такие как оптимизация гиперпараметров, интегрированное отслеживание экспериментов и автоматический экспорт в популярные форматы экспорта.
* Реализован на PyTorch, поэтому лишен ограничений фреймворка DarkNet
* CIoU-NMS
* DIoU
* 
* 
* Изображение выглядит как текст, снимок экрана, Шрифт, документ

  Автоматически созданное описание

# Гиперпараметры

* lr0: 0.001 # начальный learning rate
* lrf: 0.01 # конечный learning rate = lr0 \* lrf
* momentum: 0.937 # «импульс» в оптимизаторе
* weight\_decay: 0.0005 # L2 регуляризация, добавляет штраф к функции потерь, зависящий от весов модели
* warmup\_epochs: 3.0 # каждые n эпох применяется learning rate и после увеличивается до lrf
* warmup\_momentum: 0.8 # определяет начальное значение для momentum
* warmup\_bias\_lr: 0.1 # определяет начальное значение скорости обучения для смещений
* box: 0.05 # вес компонента потери бокса в функции потерь, влияющего на то, насколько большое значение придается точному прогнозированию координат ограничивающего бокса
* cls: 0.5 # вес потери классификации в общих функции потерь, влияющей на важность правильного прогнозирования класса по отношению к другим компонентам
* cls\_pw: 1.0 # cls BCELoss positive\_weight
* obj: 1.0 # obj loss gain (scale with pixels)
* obj\_pw: 1.0 # obj BCELoss positive\_weight
* iou\_t: 0.4 # iou threshold – порог IoU
* anchor\_t: 4.0 # anchor threshold – порог бокса
* fl\_gamma: 0.0 # focal loss gamma – задает значение для $ \ гамма $

# аугментации:

* hsv\_h: 0.015 # изменение оттенка
* hsv\_s: 0.3 # изменение насыщенности
* hsv\_v: 0.2 # изменение яркости
* degrees: 20.0 # диапазон поворота картинки
* translate: 0.1 # смещает картинку
* scale: 0.5 # масштабирует изображение
* shear: 0.0 # наклоняет изображение на заданный градус по диагонали
* perspective: 0.0 # наклоняет изображение
* flipud: 0.5 # переворачивает по вертикали
* fliplr: 0.5 # переворачивает по горизонтали
* mosaic: 1.0 # 4 рандомных изображения подгоняются под один размер соединяются, а затем из этого квадрата берётся рандомный кроп из центра
* mixup: 0.0 # смешивает 2 изображения и их маски
* copy\_paste: 0.0 # копирует объект с одного изображения и вставляет его на другое

