Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ _____Информатика и системы управления_____

КАФЕДРА _____Системы обработки информации и управления_____

Тема

Проект обнаружения целей на основе YOLO v5 - определение того, носит ли персонал строительной площадки каски

ИУ5-22М

Студент Ван Чаочао

2022 г.

Аннотация

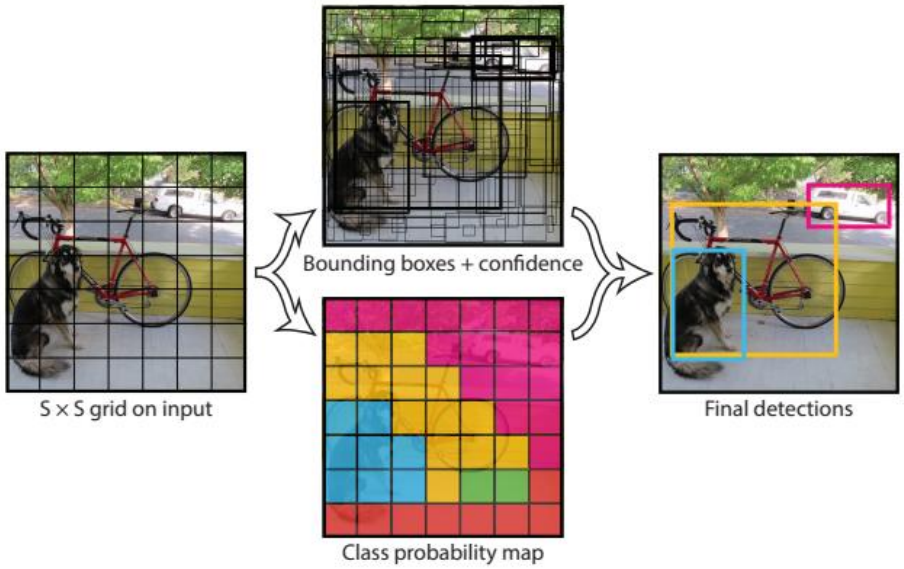
Эта работа основана на фреймворке Pytorch, с использованием Pycharm, на основе сет евой архитектуры YOLOV5 и формированием обучающего набора данных с помощью аннот ации labelimg, реализации видеоизображения, передаваемого через камеру, и определения то го, носит ли работник защитный шлем при входе в помещение. стройплощадка, иначе рабочи е ворота не откроются..

Содержание

1. **Базовые теоретические знания**

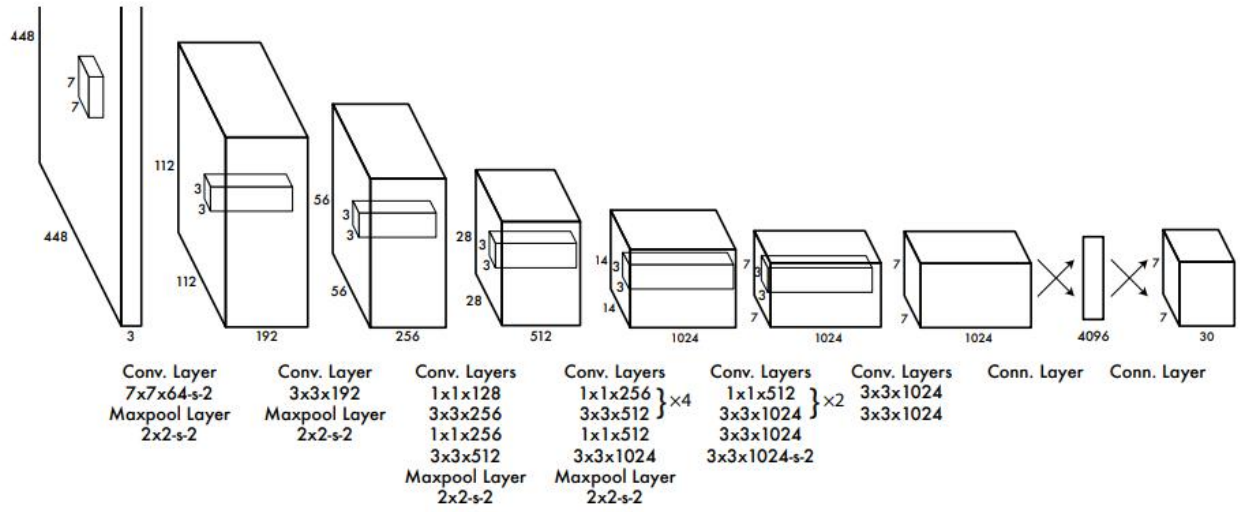
1.1 YOLO v1

1） Разделите изображение на ячейки сетки SxS, если центр объекта попадает в эту сетку, сетка отвечает за предсказание объекта



2） Каждая сетка должна предсказывать ограничивающие рамки B. Помимо предсказания положения, каждая ограничивающая рамка также должна предсказывать значение достоверности. Каждая сетка также прогнозирует баллы для категорий C.

1.1.1 сетевая структура



Для оценки YOLO на PASCAL VOC мы используем S = 7,

B = 2. PASCAL VOC имеет 20 помеченных классов, поэтому C = 20.

Наш окончательный прогноз — это тензор 7 × 7 × 30.

1.1.2функция потерь

$$
\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}
$$

Ошибка ширины и высоты ограничительной рамки представляет собой разницу между корне вым знаком, потому что для маленькой цели и большой цели, если смещение равно той же ве личине в то же время, ошибка значения долговой расписки для маленькой цели составляет бо льше.

1.2. YOLOv2

1.2.1 Изменения в YOLOv2

1: нормализация партии

Добавляя BN ко всем сверточным слоям, mAP можно улучшить на 2%, а отсев можно удалить без переобучения.

2: классификатор высокого разрешения (классификатор высокого разрешения)

В предыдущей сети классификации (магистральной) в основном использовался вход с разрешением менее 256*256, в V1 оно было 224*224, а в v2 увеличилось до 448*448, что принесло улучшение mAP%4 .

Почему увеличение входного разрешения может улучшить mAP?

3: сверточный с якорными коробками

Использование предсказания ограничительной рамки цели на основе привязки в версии 1 напрямую предсказывает WH и координаты центра цели, поэтому эффект позиционирования

очень слабый.Использование предсказания ограничительной рамки привязки может эффективно решить эту проблему и улучшить сеть Обучить и учиться, чтобы добиться эффекта конвергенции.

В версии 2 слой пула в версии 1 удален, чтобы увеличить его разрешение, а исходный ввод уменьшен с 448 до 416, так что на карте объектов может быть нечетное количество позиций, так что остается только одна центральный блок.

При использовании якорных блоков, хотя mAP модели немного снижается, отзыв значительно улучшается, что указывает на то, что модель все еще имеет много возможностей для улучшения.

4: Кластеры измерений

Два поля привязки, сгенерированные в v1, помечаются вручную, а затем учатся корректировать положение поля с помощью обучения сети, но если мы выберем лучшие приоритеты, это будет более благоприятно для обучения сети.

Как получить приоритеты, автор рассказал с помощью метода кластеризации. Кластер K-средних Алгоритм кластеризации K-средних

Объяснение K-значений: k-средств

5: прямое предсказание местоположения

При использовании поля привязки обучение будет нестабильным, что в основном вызвано прогнозированием целевой центральной точки (x, y), Формула выглядит следующим образом:

$$x = (t_x * w_a) - x_a$$
$$y = (t_y * h_a) - y_a$$

Однако при таких разных tx, ty невозможно ограничить предсказание целевого ограничивающего прямоугольника, что приведет к тому, что ограничивающий прямоугольник появится в любой позиции изображения, а после превышения границы, даже если есть цель в другие позиции, это должна быть ячейка этой позиции для прогнозирования, что приводит к нестабильности сети. Поэтому автор предлагает новую формулу расчета: целевое смещение ограничено между 0 и 1 через сигмовидную функцию

$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$
$$b_w = p_w e^{t_w}$$
$$b_h = p_h e^{t_h}$$
$$Pr(\text{object}) * IOU(b, \text{object}) = \sigma(t_o)$$

Вместо метода прямого предсказания якоря в более быстрой CNN после использования класт еризации полученная ограничительная рамка и изменение координат центра прямо предсказа нной ограничительной рамки принесли улучшение нашей сети на 5%.

6: Мелкозернистые функции

Карта объектов высокого уровня объединяется с картой объектов нижнего уровня, а карта об ъектов низкого уровня 26*26*512 и карта объектов высокого уровня 13*13*1024 объединяют ся в направлении глубины через сквозной слой. . Это приводит к увеличению на 1%

Через сквозной слой ширина и высота становятся 1/2 от оригинала, а глубина становится в 4 раза больше оригинала

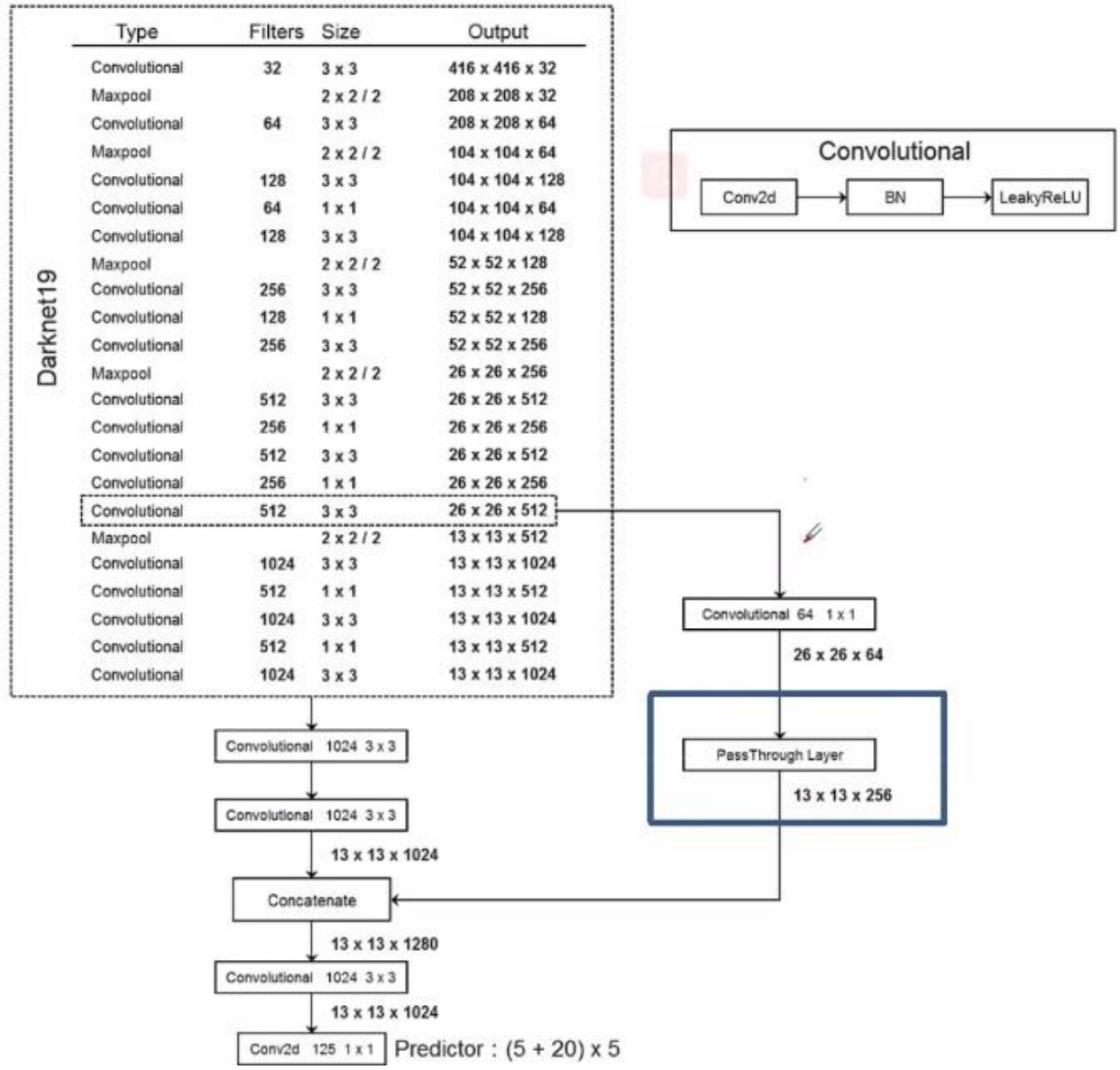
7: Многомасштабное обучение

Коэффициент масштабирования: ввод: 416*416. вывод: 13 * 13 Таким образом, коэффициент масштабирования YOLO равен 32 = 416/13.

Каждые 10 пакетов итераций случайным образом измените размер ввода {320, 352, ...608}

1.2.2 YOLO v2 рамка модели

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			



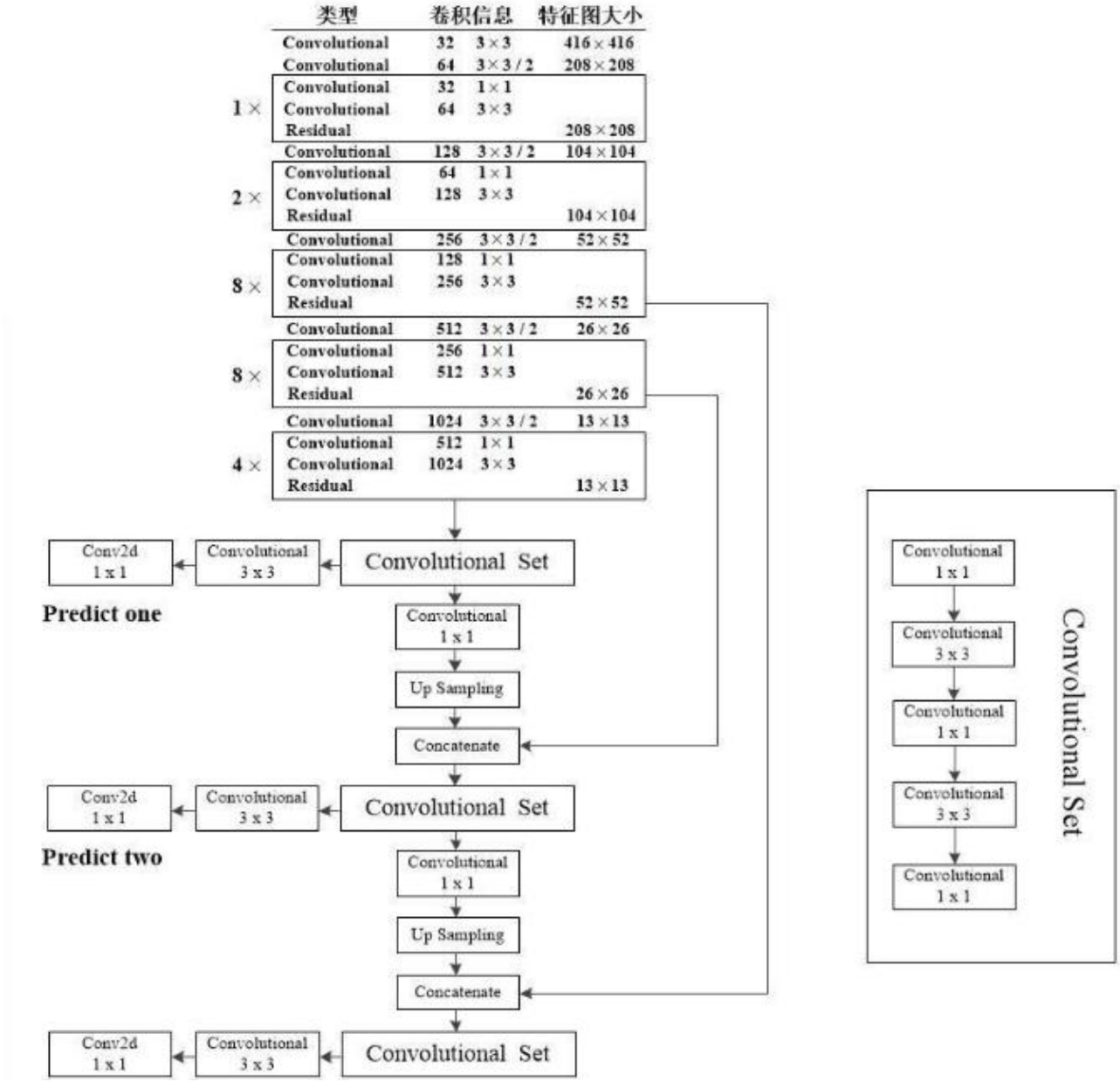
Удален последний сверточный слой и добавлен слой passThrough.

1.3 YOLOv3

Он в основном объединяет некоторые функции более популярных сетей,

В его сети сверточный слой используется для замены исходного слоя пула Max, что повышае т точность, но в то же время снижает количество FP. Поскольку объединение приведет к пот ере информации,

1.3.1 Архитектура сетевой модели YOLOV3



Удалите средний пул даркнета 53, а затем сделайте прогнозы по трем слоям признаков разной глубины.

1.3.2Расчет убытков

$$L(o,c,O,C,l,g) = \lambda_1 L_{conf}(o,c) + \lambda_2 L_{cla}(O,C) + \lambda_3 L_{loc}(l,g)$$

$$L_{conf}(o,c) = -\frac{\sum_i (o_i \ln(\hat{c}_i) + (1-o_i)\ln(1-\hat{c}_i))}{N}$$

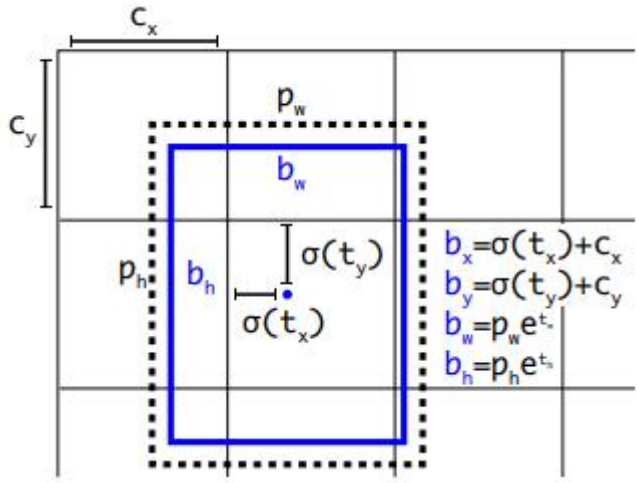
$$\hat{c}_i = Sigmoid(c_i)$$

$$L_{cla}(O,C) = -\frac{\sum_{i \in pos} \sum_{j \in cla} (O_{ij} \ln(\hat{C}_{ij}) + (1 - O_{ij}) \ln(1 - \hat{C}_{ij}))}{N_{pos}}$$

$$\hat{C}_{ij} = Sigmoid(C_{ij})$$

$$L_{loc}(t,g) = \frac{\sum_{i \in pos} (\sigma(t_x^i) - \hat{g}_x^i)^2 + (\sigma(t_y^i) - \hat{g}_y^i)^2 + (t_w^i - \hat{g}_w^i)^2 + (t_h^i - \hat{g}_h^i)^2}{N_{pos}}$$

Ограничивающие рамки с априорными размерами и местоположением прогноз. Мы предсказ ываем ширину и высоту коробки как смещения от центроидов кластера. Предсказываем коор динаты центра поле относительно местоположения приложения фильтра с помощью сигмоид ы функция. Эта цифра является откровенным самоплагиатом из



1.4 YOLO SPP

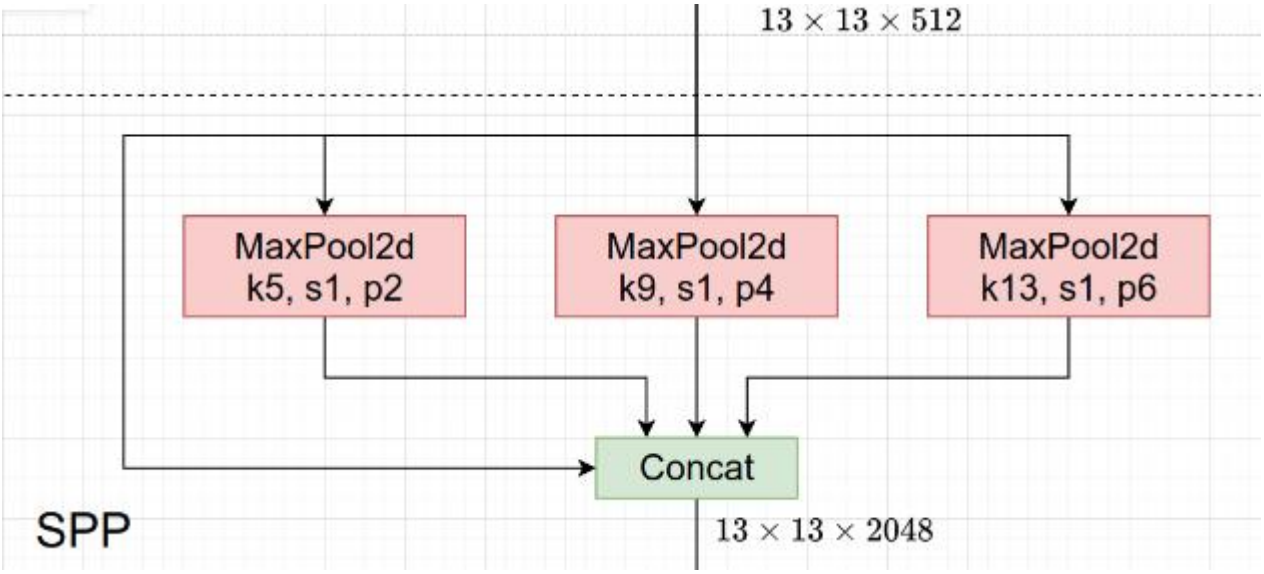
1) Мозаичное улучшение изображения

Случайным образом выберите четыре изображения для объединения, эффект:

1. Увеличить разнообразие данных, 2. Увеличить количество целей, 3. БН может считать параметры нескольких картинок одновременно

2) Модуль СПП

По сравнению с v3 модуль spp с тремя максимальными пулами добавлен к функциональному уровню 13 * 13. Остальные такие же, как в v3, а четыре глубоко интегрированы.



3.очаговая потеря

В основном это предлагается для одноэтапной сети обнаружения цели, потому что существует большое количество блоков-кандидатов, но количество положительных отсчетов чрезвычайно мало, а количество отрицательных отсчетов чрезвычайно велико, поэтому количество отсчетов несбалансировано. .

Хотя потери, вызванные выборкой, невелики, из-за большого количества отрицательных выборок сумма потерь может также маскировать потери положительных выборок. Поэтому автор статьи шаг за шагом совершенствует формулу расчета, чтобы уменьшить вклад потерь легко разделяемых выборок.

Разница между CE и FL для твердых образцов не очень велика.
Фокусная потеря фокусируется на поиске образцов, которые трудно изучить.Если набор данных помечен неправильно, это образец, который трудно изучить, и он будет учиться в неправильном направлении. Поэтому FL чувствителен к шумовым помехам,

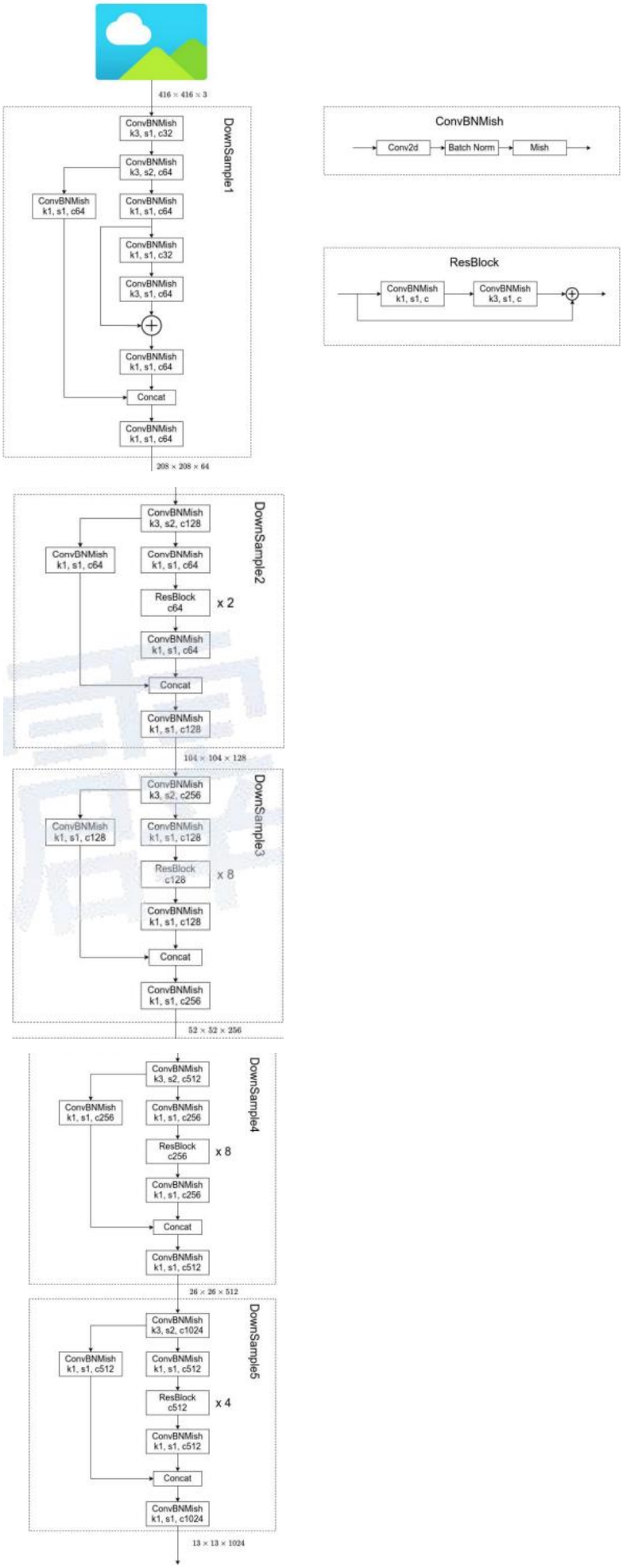
1.5 YOLOv4

сетевая структура: Backbone: CSPDarknet53 Neck: SPP, PAN Head: YOLOv3

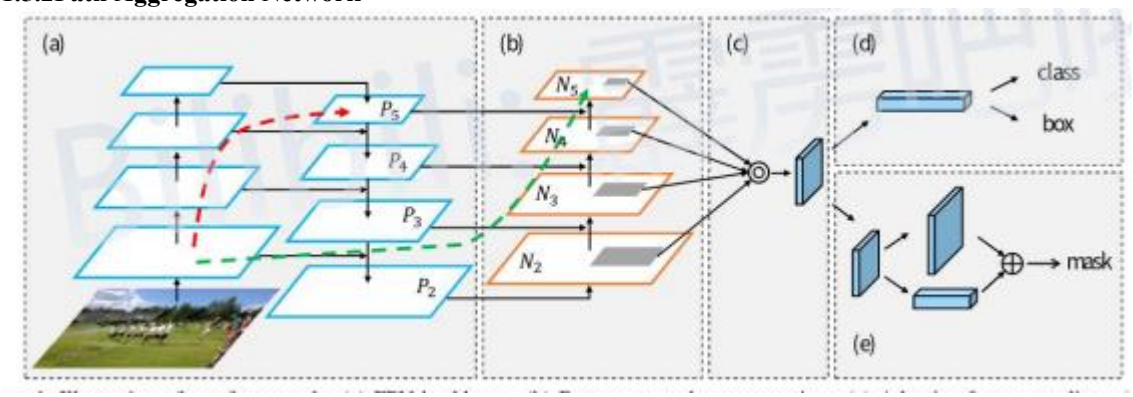
Стратегия оптимизации:

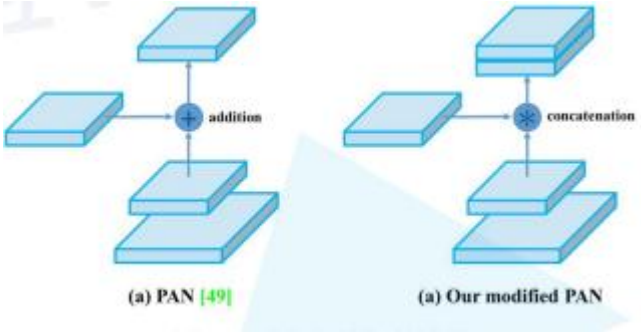
1) Eliminate grid sensitivity 2)Mosaic data augmentation 3) IoU threshold(match positive samples) 4)Optimizered Anchors 5) CIOU

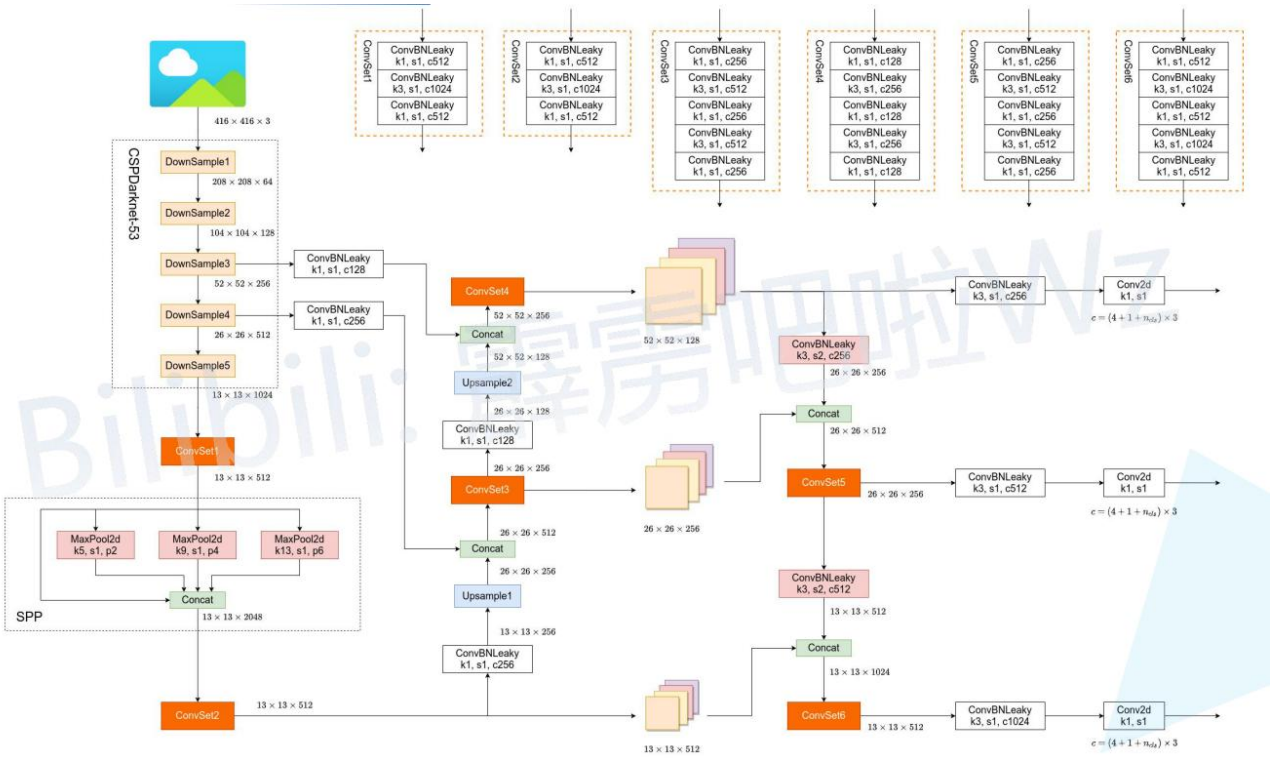
1.5.1 сетевая структура



1.5.2Path Aggregation Network







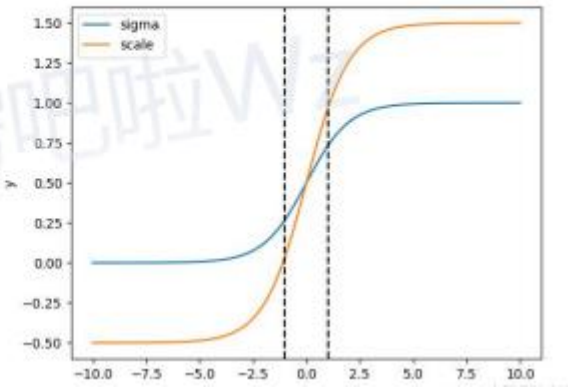
1.5.3Стратегия оптимизации

Eliminate grid sensitivity

$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$

$$b_x = (\sigma(t_x) \cdot scale_{xy} - \frac{scale_{xy} - 1}{2}) + c_x$$
$$b_y = (\sigma(t_y) \cdot scale_{xy} - \frac{scale_{xy} - 1}{2}) + c_y$$

$$b_x = (2 \cdot \sigma(t_x) - 0.5) + c_x$$
$$b_y = (2 \cdot \sigma(t_y) - 0.5) + c_y$$



1.6 YOLO v5

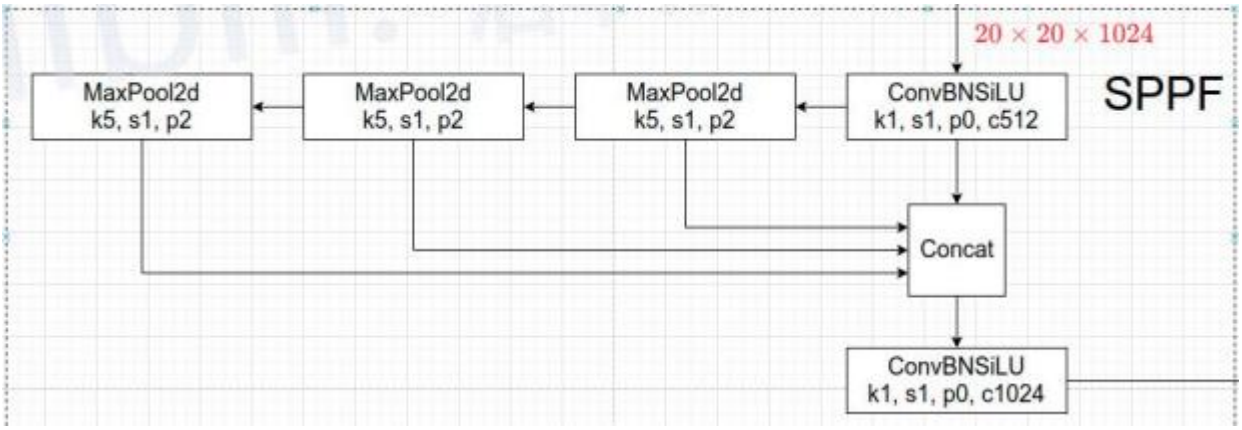
1.6.1 сетевая структура

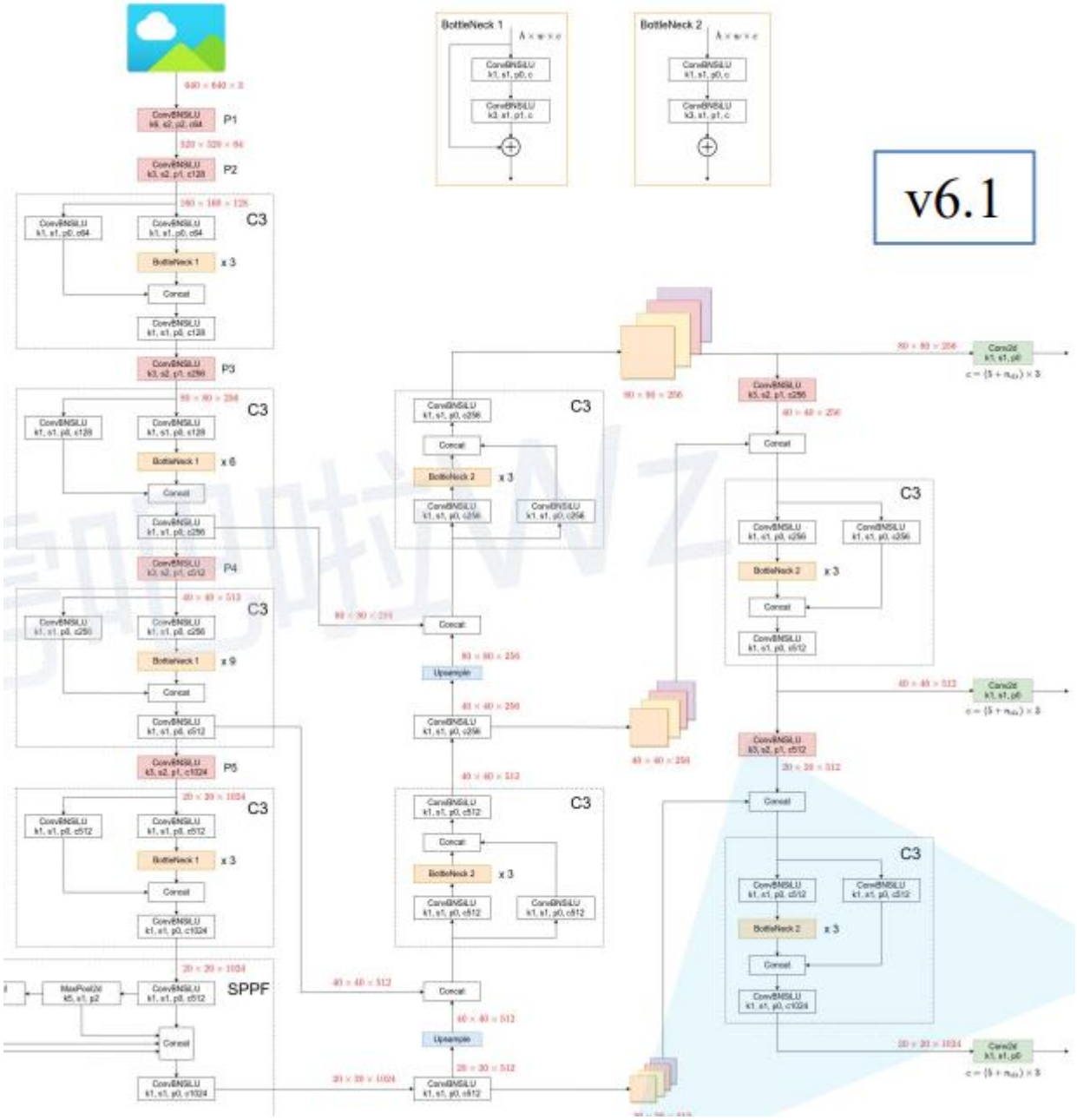
Backbone: New CSP-Darknet53

Neck: SPPF, New CSP-PAN

Head: YOLOv3 Head

Заменен модуль Focus на обычный сверточный слой 6x6.Оба работают одинаково, но послед ний более эффективен.




BottleNeck 1
BottleNeck 2
v6.1
C3
SPPF
P1
P2
P3
P4
P5
Concat
Upsample

2. Реализация проекта

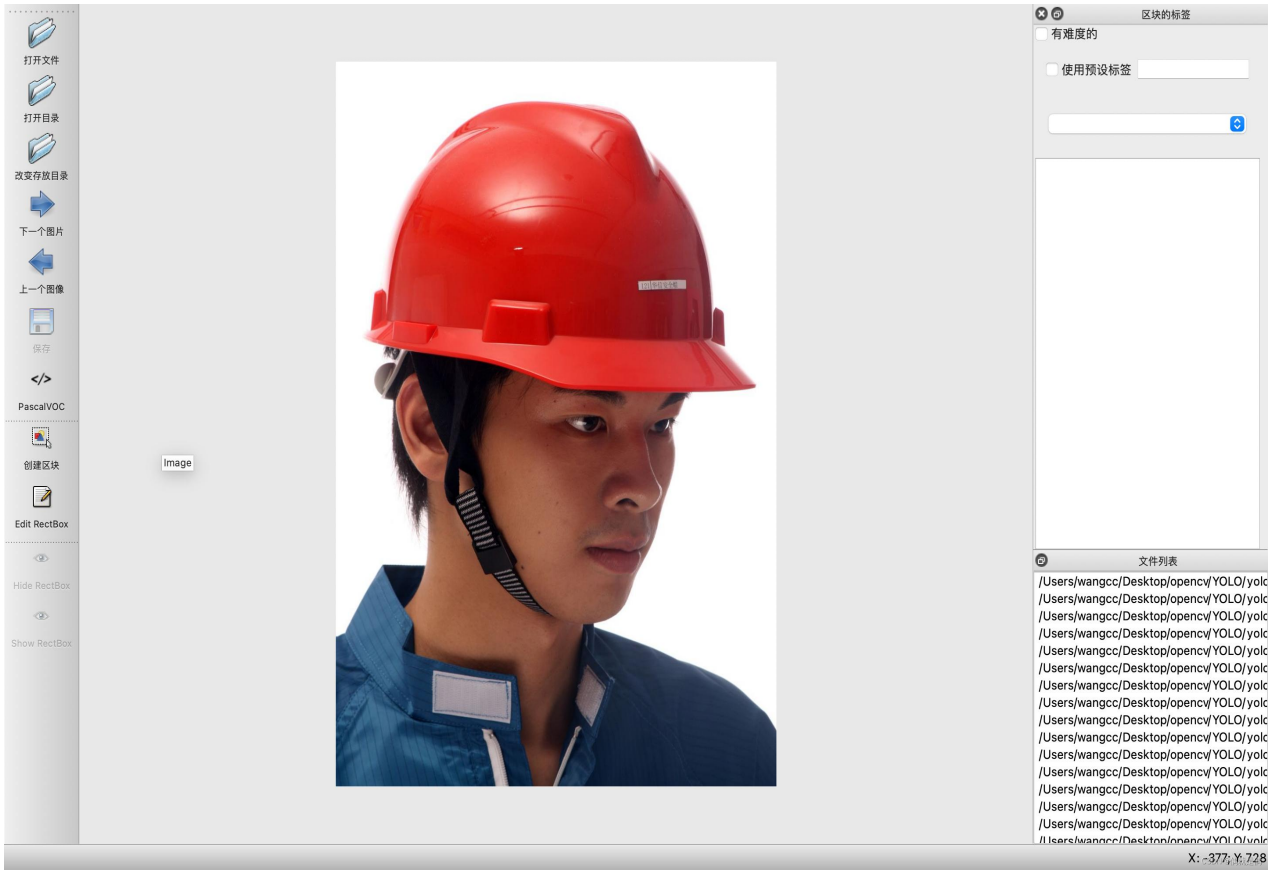
2.1Установка среды Pytorch

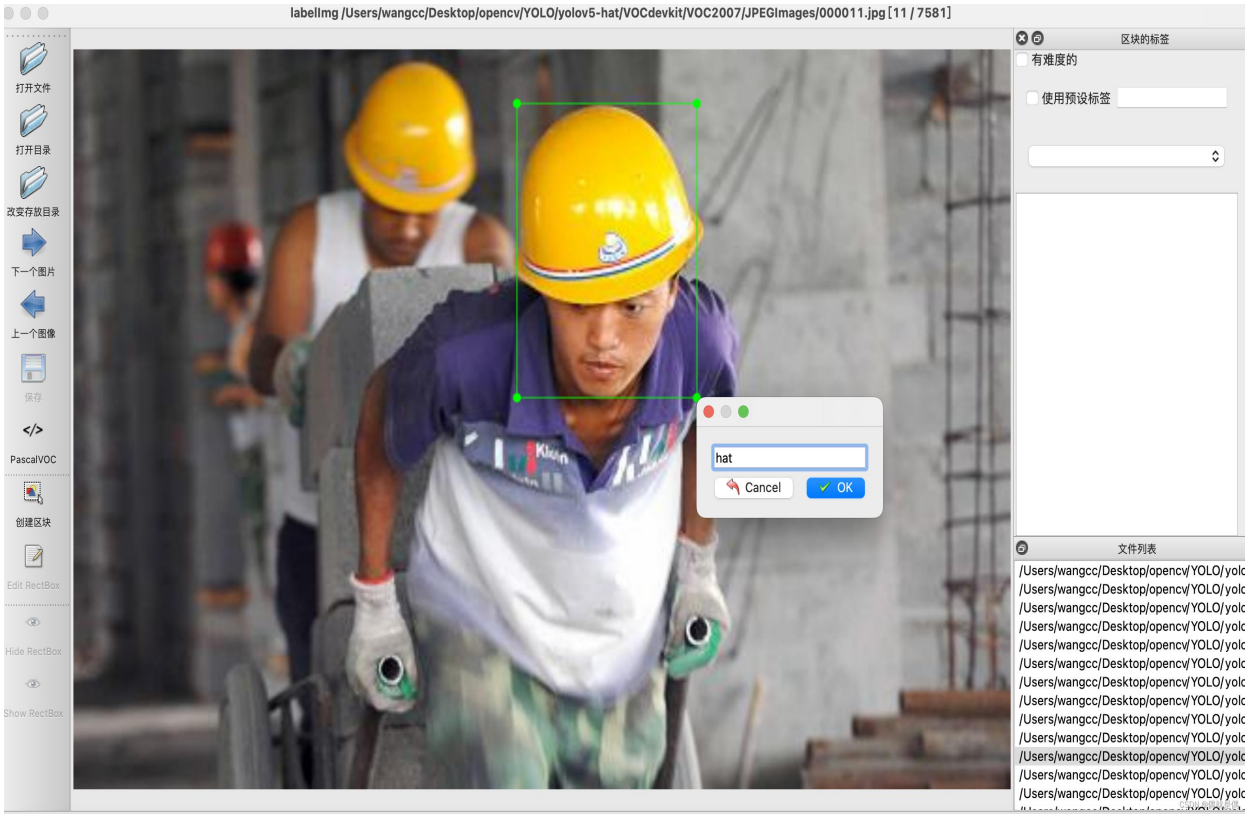
1: Сначала загрузите версию установочного пакета anaconda для Windows с официального сайта anaconda https://www.anaconda.com/.

После завершения загрузки создайте новую папку на диске D: установите ее на диск D. Запомнить! ! !

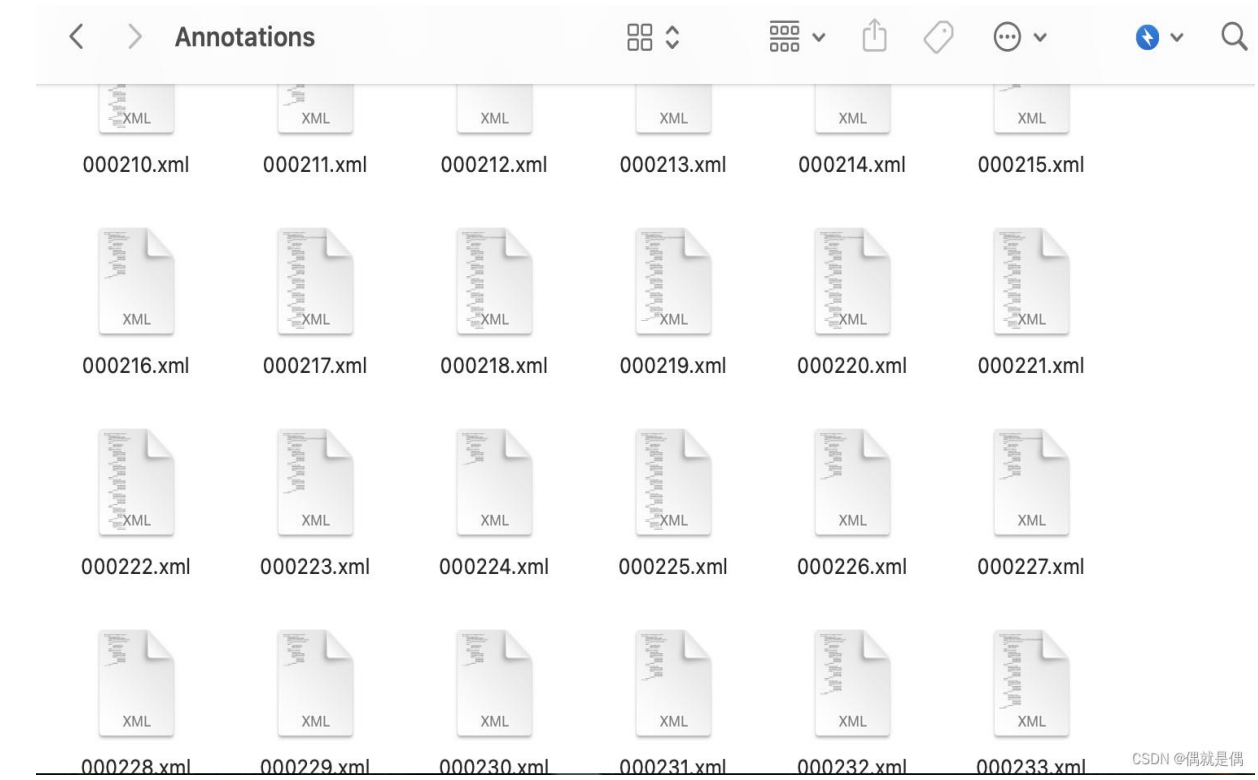
2: Установите pytorch через conda

2.2 Используйте labelimg, чтобы создать собственный набор данных для обнаружения целей глубокого обучения.





После ввода просмотрите файл этикетки в файле аннотаций.



2.3 кодовый модуль

CONV

```
class Conv(nn.Module):
    # Standard convolution
    def __init__(self, c1, c2, k=1, s=1, p=None, g=1, act=True):  # ch_in, ch_out, kernel, stride, padding, groups
        super(Conv, self).__init__()
        self.conv = nn.Conv2d(c1, c2, k, s, autopad(k, p), groups=g, bias=False)
        self.bn = nn.BatchNorm2d(c2)
        self.act = nn.SiLU() if act is True else (act if isinstance(act, nn.Module) else nn.Identity())

    def forward(self, x):
        return self.act(self.bn(self.conv(x)))

    def fuseforward(self, x):
        return self.act(self.conv(x))
```

FOCUS

```
class Focus(nn.Module):
    # Focus wh information into c-space
    def __init__(self, c1, c2, k=1, s=1, p=None, g=1, act=True):  # ch_in, ch_out, kernel, stride, padding, groups
        super(Focus, self).__init__()
        self.conv = Conv(c1 * 4, c2, k, s, p, g, act)
        # self.contract = Contract(gain=2)

    def forward(self, x):  # x(b,c,w,h) -> y(b,4c,w/2,h/2)
        return self.conv(torch.cat([x[..., ::2, ::2], x[..., 1::2, ::2], x[..., ::2, 1::2], x[..., 1::2, 1::2]], 1))
        # return self.conv(self.contract(x))
```

BOTTLENECKCSP

```
lass BottleneckCSP(nn.Module):
    # CSP Bottleneck https://github.com/WongKinYiu/CrossStagePartialNetworks
    def __init__(self, c1, c2, n=1, shortcut=True, g=1, e=0.5):  # ch_in, ch_out, number, shortcut, groups, expansion
        super(BottleneckCSP, self).__init__()
        c_ = int(c2 * e)  # hidden channels
        self.cv1 = Conv(c1, c_, 1, 1)
```

```
            self.cv2 = nn.Conv2d(c1, c_, 1, 1, bias=False)
            self.cv3 = nn.Conv2d(c_, c_, 1, 1, bias=False)
            self.cv4 = Conv(2 * c_, c2, 1, 1)
            self.bn = nn.BatchNorm2d(2 * c_)  # applied to
cat(cv2, cv3)
            self.act = nn.LeakyReLU(0.1, inplace=True)
            self.m = nn.Sequential(*[Bottleneck(c_, c_, shortcut,
g, e=1.0) for _ in range(n)])

        def forward(self, x):
            y1 = self.cv3(self.m(self.cv1(x)))
            y2 = self.cv2(x)
            return self.cv4(self.act(self.bn(torch.cat((y1, y2),
dim=1))))
```

bottleneck

```
    class Bottleneck(nn.Module):
        # Standard bottleneck
        def __init__(self, c1, c2, shortcut=True, g=1, e=0.5):  #
ch_in, ch_out, shortcut, groups, expansion
            super(Bottleneck, self).__init__()
            c_ = int(c2 * e)  # hidden channels
            self.cv1 = Conv(c1, c_, 1, 1)
            self.cv2 = Conv(c_, c2, 3, 1, g=g)
            self.add = shortcut and c1 == c2

        def forward(self, x):
            return x + self.cv2(self.cv1(x)) if self.add else
self.cv2(self.cv1(x))
```

SPP

```
class SPP(nn.Module):
    # Spatial pyramid pooling layer used in YOLOv3-SPP
    def __init__(self, c1, c2, k=(5, 9, 13)):
        super(SPP, self).__init__()
        c_ = c1 // 2  # hidden channels
        self.cv1 = Conv(c1, c_, 1, 1)
        self.cv2 = Conv(c_ * (len(k) + 1), c2, 1, 1)
        self.m = nn.ModuleList([nn.MaxPool2d(kernel_size=x,
stride=1, padding=x // 2) for x in k])
```

```
def forward(self, x):
    x = self.cv1(x)
    return self.cv2(torch.cat([x] + [m(x) for m in self.m], 1))
```

2.4Отображение результатов

При ношении шлема система может хорошо определить, носит ли человек шлем, и отобразит ь соответствующую вероятность в верхнем конце поля идентификации.

Вывод

Хотя можно определить, носит ли человек шлем или нет, это только первый шаг.Позже, в соответствии с результатом идентификации, необходимо сопоставить соответствующую фактическую прикладную систему, например, как передавать информацию в систему управления. систему, реализовать посадочное приложение соответствующего проекта и разработать элементы, соответствующие работам, которые можно использовать. Действительно решить потребности жизни людей

Список литературы

[1] T. Liu, Y. Liu, Z. Tang and J.-N. Hwang. Adaptive ground plane estimation for moving camera-based 3D object tracking. IEEE Int. Workshop Multimedia Signal Processing, 2017.

[2] J. Sochor, J. Špaňhel and A. Herout. BoxCars: Improving fine-grained recognition of vehicles using 3-D bounding boxes in traffic surveillance. IEEE Trans. Intelligent Transportation Syst., 2018.

[3] K.-H. Lee, J.-N. Hwang and S.-I Chen. Model-based vehicle localization based on three-dimensional constrained multiple-kernel tracking. IEEE Trans. Circuits Syst. Video Technol., 25(1):38-50, 2014.

[4] A. Milan, S. Roth and K. Schindler. Continuous energy minimization for multitarget tracking. IEEE Trans. Pattern Analysis & Machine Intelligence, 36(1):58-72, 2014.

[5] A. Milan, K. Schindler and S. Roth. Multi-target tracking by discrete-continuous energy minimization. IEEE Trans. Pattern Analysis & Machine Intelligence, 38(10):2054-2068, 2016.

[6] X. Liu, W. Liu, H. Ma and H. Fu. Large-scale vehicle re identification in urban surveillance videos. Proc. IEEE Int. Conf. Multimedia Expo, 2016.

[7] L. Yang, P. Luo, C. C. Loy and X. Tang. A large-scale car dataset for fine-grained categorization and verification. Proc. IEEE Conf. Comput. Vis. Pattern Recogn., 3973–3981, 2015.

[8] C. N. E. Anagnostopoulos, I. E. Anagnostopoulos, I. D. Psoroulas, V. Loumos and E. Kayafas, E. License plate recognition from still images and video sequences: A survey. IEEE Trans. Intelligent Transportation Syst., 9(3):377-391, 2008.

[9] NVIDIA AI City Challenge – Data and Evaluation. [Online] Available: https://www.aicitychallenge.org/?page_id=9.

[10] C.-T. Chu, J.-N. Hwang, H.-Y. Pai and K.-M. Lan. Tracking human under occlusion based on adaptive multiple kernels with projected gradients. IEEE Trans. Multimedia, 15:1602- 1615, 2013

[11] Z. Tang, J.-N. Hwang, Y.-S. Lin and J.-H. Chuang. Multiple kernel adaptive segmentation and tracking (MAST) for robust object tracking. Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing, 1115-1119, 2016.

[12] Z. Tang, G. Wang, T. Liu, Y.-G. Lee, A. Jahn, X. Liu, X. He and J.-N. Hwang. Multiple-kernel based vehicle tracking using 3D deformable model and camera self-calibration. arXiv preprint arXiv:1708.06831, 2017.

[13] Z. Tang, Y.-S. Lin, K.-H. Lee, J.-N. Hwang, J.-H. Chuang and Z. Fang. Camera self-calibration from tracking of moving persons. Proc. Int. Conf. Pattern Recognition, 260- 265, 2016.

[14] Z. Tang, R. Gu and J.-N. Hwang. Joint multi-view people tracking and pose estimation for 3D scene reconstruction. Proc. IEEE Int. Conf. Multimedia Expo, 2018.

[15] J. Redmon and A. Farhadi. YOLO9000: Better, Faster, Stronger. arXiv preprint arXiv: 1612.08242, 2016.

[16] B. Caprile and V. Torre. Using vanishing points for camera calibration. Int. J. Computer Vision, 4(2):127-139, 1990.

[17] Google Maps. [Online] Available: https://maps.google.com/.

[18] P. Larranaga and J. A. Lozano. Estimation of distribution algorithms: A new tool for evolutionary computation. Springer Science & Business Media, 2nd edition, 2002.

[19] L. Wen, D. Du, Z. Cai, Z. Lei, M.-C. Chang, H. Qi, J. Lim, M.-H. Yang, S. Lyu. UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking. arXiv preprint arXiv:1511.04136, 2015.

[20] G. Wang, J.-N. Hwang, K. Williams and G. Cutter. Closed loop tracking-by-detection for ROV-based multiple fish tracking. Int. Conf. Pattern Recognition Workshop Comput. Vis. Analysis Underwater Imagery, 7-12, 2016.

[21] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich. Going deeper with convolutions. Proc. IEEE Conf. Comput. Vis. Pattern Recogn., 2015.