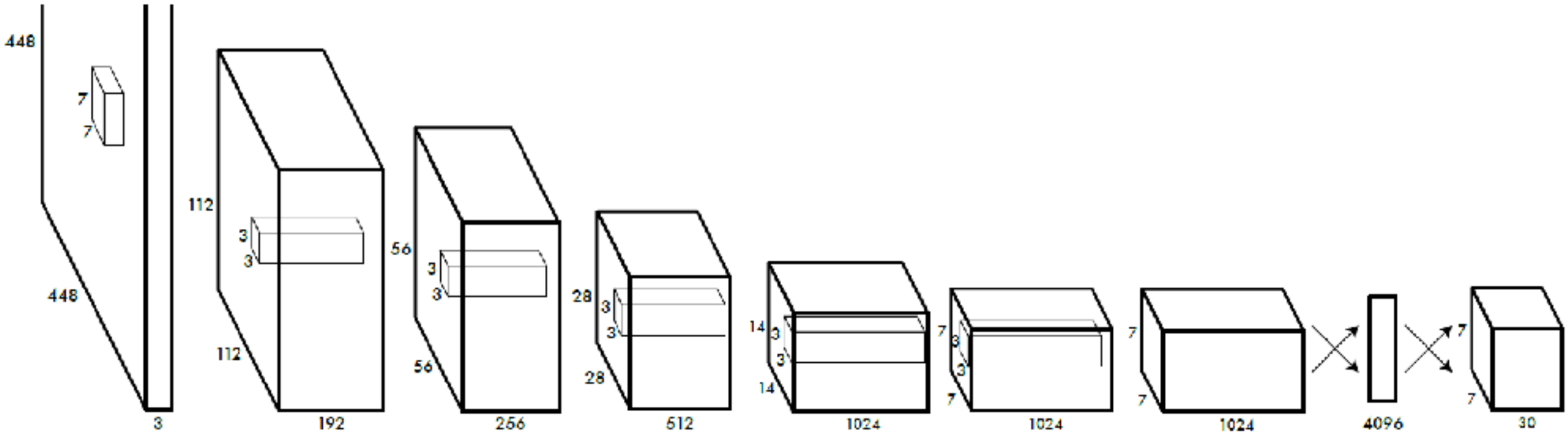
CNN应用案例

YOLO实现目标检测



Conv. Layer
7x7x64-s-2
Maxpool Layer
2x2-s-2

Conv. Layer
3x3x192
Maxpool Layer
2x2-s-2

Conv. Layers
1x1x128
3x3x256
1x1x256
3x3x512
Maxpool Layer
2x2-s-2

Conv. Layers
1x1x256
3x3x512
} ×4
1x1x512
3x3x1024
Maxpool Layer
2x2-s-2

Conv. Layers
1x1x512
3x3x1024
} ×2
3x3x1024
3x3x1024-s-2

Conv. Layers
3x3x1024
3x3x1024

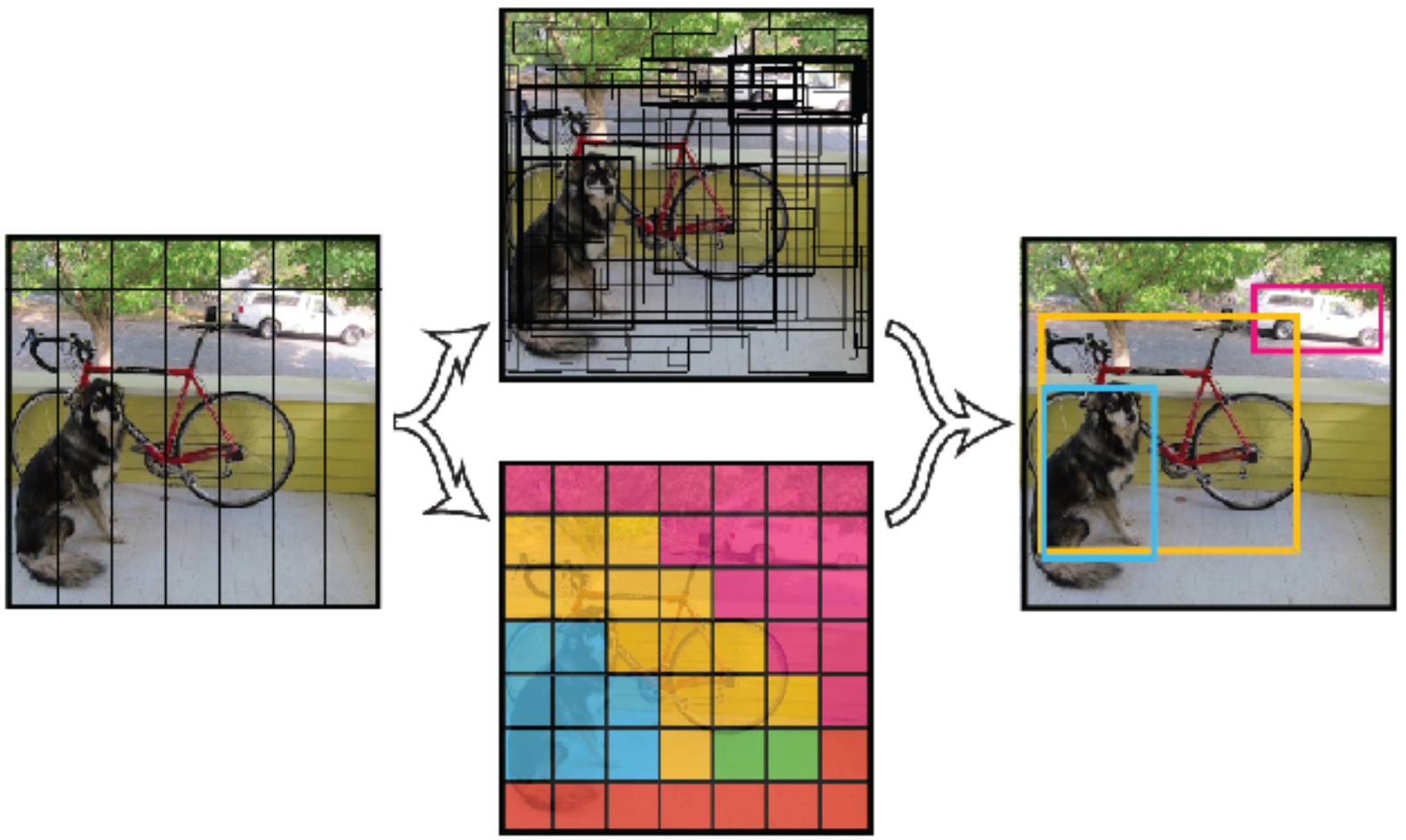
Conn. Layer

Conn. Layer

YOLO核心思想

- YOLO的核心思想就是利用整张图作为网络的输入，直接在输出层回归bounding box的位置和bounding box所属的类别。

YOLO的实现方法



论文：You Only Look Once: Unified, Real-Time Object Detection

- 将一幅图像分成SxS个网格(grid cell)，如果某个object的中心 落在这个网格中，则这个网格就负责预测这个object。

- 每个网格：
  1. 预测B个bounding box
  2. 预测C个分类

位置信息：x,y,w,h

置信度：confidence

$$\Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

- 实例应用1：YOLO实现目标检测
  - YOLO的损失函数设计

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(x_i - \hat{x}_i\right)^2 + \left(y_i - \hat{y}_i\right)^2$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2$$

坐标预测

判断第i个网格中的第j个box是否负责这个object

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i\right)^2$$

含object的box的confidence预测

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i\right)^2$$

不含object的box的confidence预测

判断是否有object中心落在网格i中

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c)\right)^2$$

类别预测

- 这个损失函数中：
  1. 只有当某个网格中有object的时候才对classification error进行惩罚。
  2. 只有当某个box predictor对某个ground truth box负责的时候，才会对box的coordinate error进行惩罚，而对哪个ground truth box负责就看其预测值和ground truth box的IoU是不是在那个cell的所有box中最大。

- YOLO对相互靠的很近的物体，还有很小的群体 检测效果不好，这是因为一个网格中只预测了两个框，并且只属于一类。

- 对测试图像中，同一类物体出现的新的不常见的长宽比和其他情况是。泛化能力偏弱。

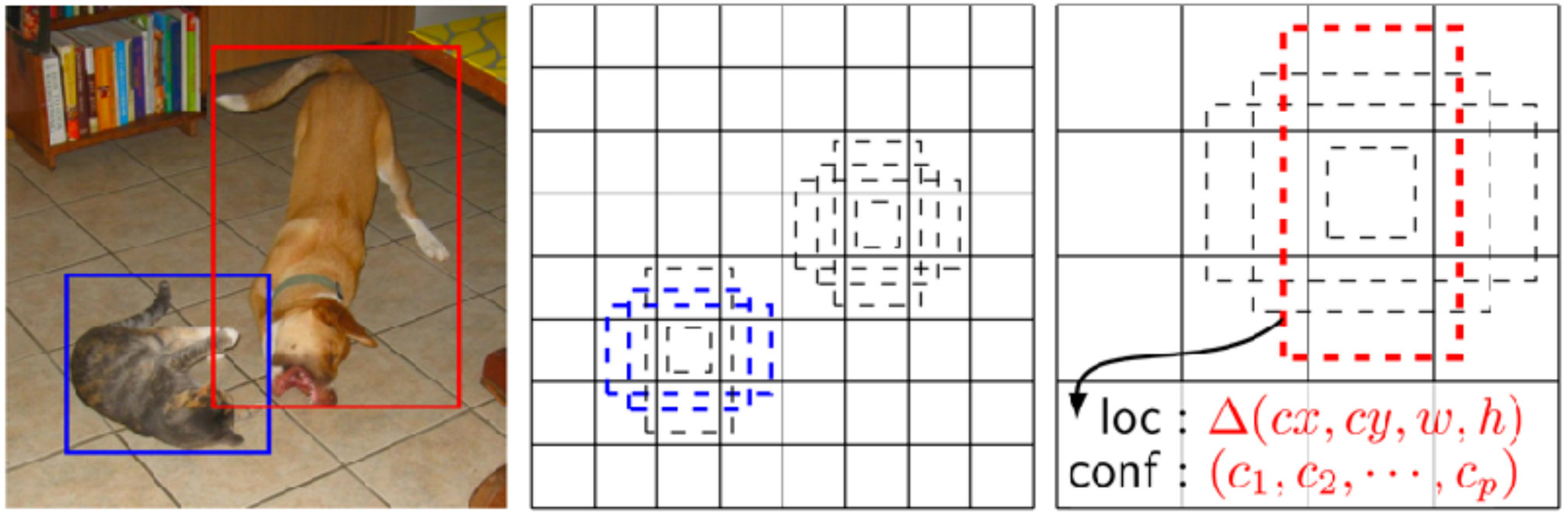
- 由于损失函数的问题，定位误差是影响检测效果的主要原因。尤其是大小物体的处理上，还有待加强。

YOLO论文代码

https://github.com/pjreddie/darknet

SSD实现物体检测

论文：《SSD:Single Shot MultiBox Detector》



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

- 由于YOLO简单粗暴的将图像进行网格划分，然后对每个网格进行处理，这样导致定位不精确等一些列问题。而基于region proposal却又定位较精确的优点，那么SSD就结合了YOLO和anchor进行检测，结果也是比yolo提高很多,速度58fps.

- 和faster的anchor不同之处在于，SSD在多个featureMap上进行处理，因为每一层featureMap的感受野不同。 faster是先提取proposal，然后在分类，而SSD值利用anchor直接进行分类和BBox回归。

- 实例应用2：SSD实现物体检测
  - SSD检测框架

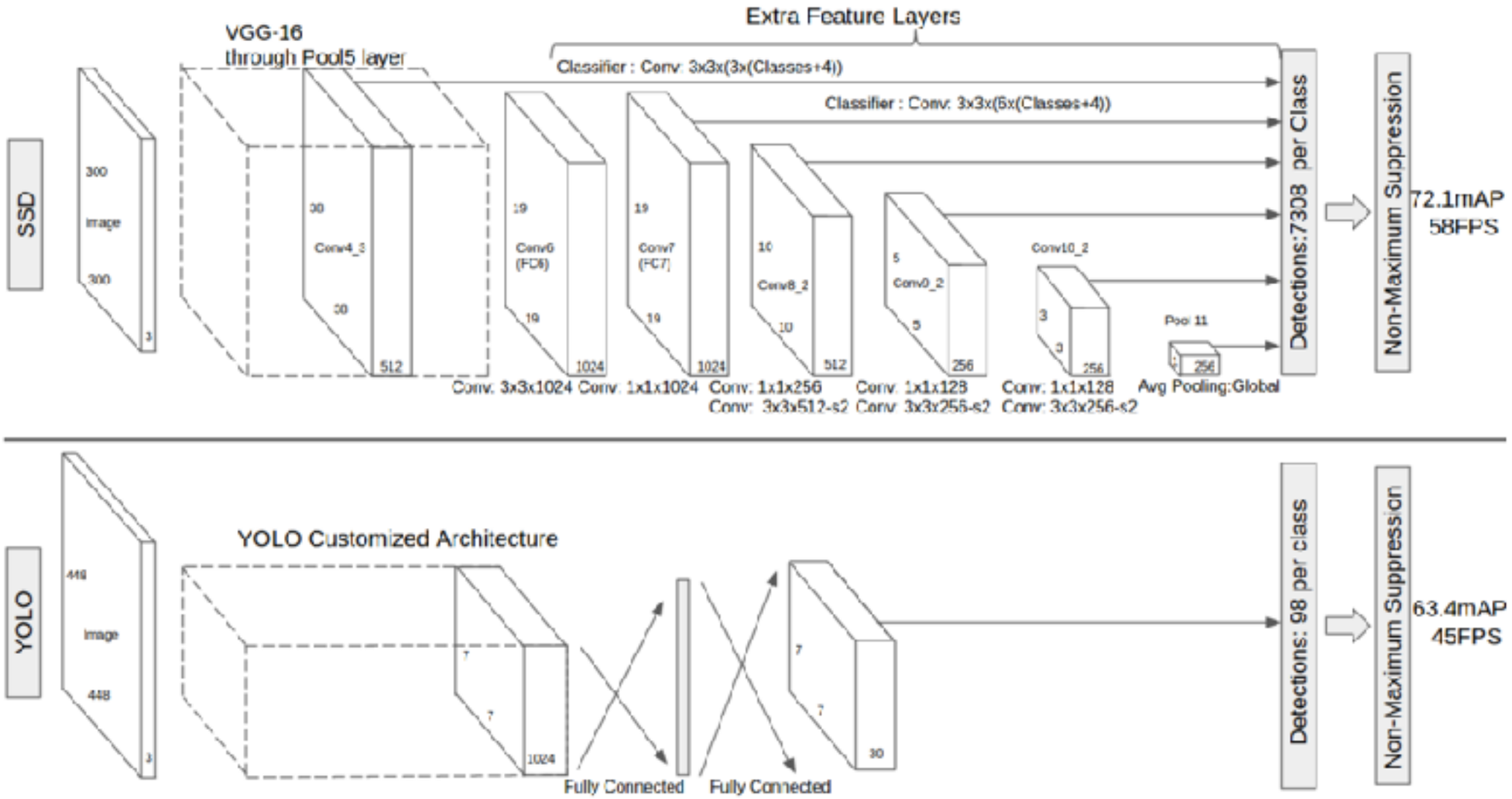


Fig. 2: A comparison between two single shot detection models: SSD and YOLO [5]. Our SSD model adds several feature layers to the end of a base network, which predict the offsets to default boxes of different scales and aspect ratios and their associated confidences. SSD with a 300×300 input size significantly outperforms its 448×448 YOLO counterpart in accuracy on VOC2007 `test` while also improving the run-time speed, albeit YOLO customized network is faster than VGG16.

SSD论文代码

https://github.com/weiliu89/caffe/tree/ssd

PixelNet原理与实现

PIXELNET:
REPRESENTATION *OF* THE PIXELS, *BY* THE PIXELS, AND *FOR* THE PIXELS.

Aayush Bansal, Xinlei Chen, Bryan Russell, Abhinav Gupta, Deva Ramanan

(a) Semantic Segmentation **(b) Surface Normal Estimation** **(c) Edge Detection**

- PixelNet探索了一般像素级预测问题的设计原则，从低级边缘检测，到中级表面正态估计，到高级的语义分割。卷积预测器，例如全卷积网络（FCN），通过卷积处理，利用相邻像素的空间冗余，已经取得了显着的成功。虽然计算效率高，这种方法在学习期间在统计学上不是有效的，因为空间冗余限制了从相邻像素学习的信息。

PixelNet网络结构

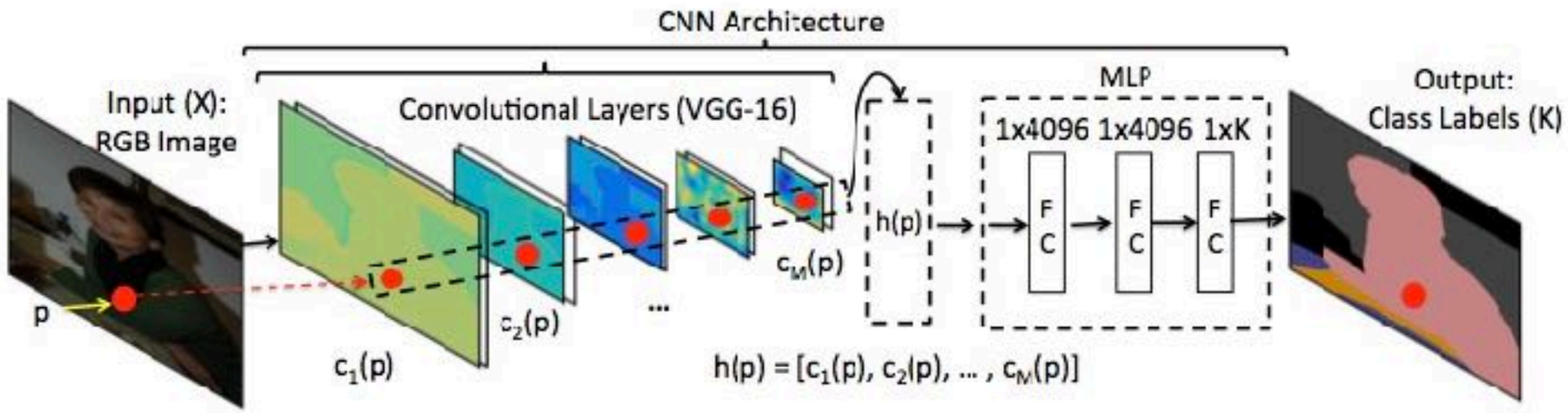


Figure 2. **PixelNet:** We input an image to a convolutional neural network, and extract hypercolumn descriptor for a sampled pixel from multiple convolutional layers. The hypercolumn descriptor is then fed to a multi-layer perceptron (MLP) for the non-linear optimization, and the last layer of MLP outputs the required response for the task. See text for more details about the use of network at training/test time.

- 像素的分层抽样可以：
- （1）在批量更新期间添加多样性，加速学习；
- （2）探索复杂的非线性预测因子，提高准确性；
- （3）为不同的像素标记任务，有效地训练最先进的模型（tabula rasa）（即“从头开始”）。我们的单一架构为PASCAL上下文数据集上的语义分割，NYUDv2深度数据集上的表面正太估计和BSDS上的边缘检测提供了最先进的结果

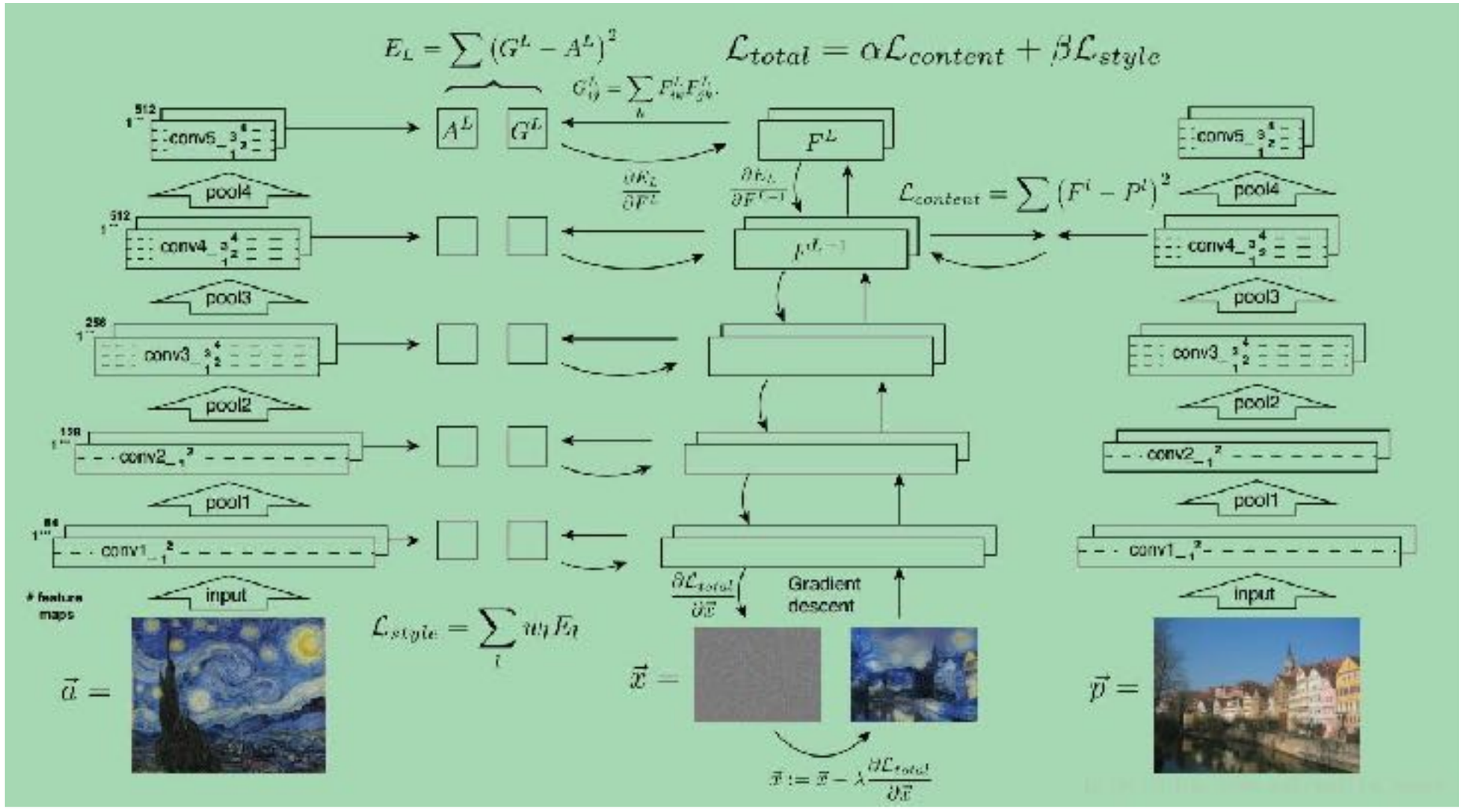
- （4）通过几何展示自我监督表示学习。使用甚至几个数据点，实现结果比以前的无监督/自我监督学习更好的结果。

PixelNet实现代码

https://github.com/aayushbansal/PixelNet

- 实例应用4：利用卷积神经网络做图像风格结合

利用卷积神经网络做图像风格结合



算法流程图

模型原理

给定一张风格图像 a 和一张普通图像 p，风格图像经过VGG-19 的时候在每个卷积层会得到很多 feature maps, 这些 feature maps 组成一个集合 A，同样的，普通图像 p 通过 VGG-19 的时候也会得到很多 feature maps，这些feature maps 组成一个集合 P，然后生成一张随机噪声图像 x, 随机噪声图像 x 通过VGG-19 的时候也会生成很多feature maps，这些 feature maps 构成集合 G 和 F 分别对应集合 A 和 P, 最终的优化函数是希望调整 x 让 随机噪声图像 x 最后看起来既保持普通图像 p 的内容, 又有一定的风格图像 a 的风格。

内容表达

在建立目标函数之前，我们需要先给出一些定义：在CNN 中，假设某一 layer 含有 N_l 个 filters，那么将会生成 N_l 个 feature maps，每个 feature map 的维度为 M_l ，M_l 是 feature map 的 高与宽的乘积。所以每一层 feature maps 的集合可以表示为 $F^l \in R^{N_l \times M_l}$ ，F^l_{ij} 表示第 i个 filter在 position j 上的 activation。

所以，我们可以给出 content 的 cost function:

$$L_{content}\ (p, x, l) = \frac{1}{2} \sum_{ij} (F^l_{ij} - P^l_{ij})$$

风格表达

为了建立风格的representation，我们先利用 Gram matrix 去表示每一层各个 feature maps 之间的关系，$G^l \in R^{N_l \times N_l}$，$G^l_{ij}$ 是 feature maps i,j 的内积：

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk}$$

利用 Gram matrix，我们可以建立每一层的关于 style 的 cost：

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G^l_{ij} - A^l_{ij})^2$$

结合所有层，可以得到总的cost

$$L_{style}(a,x) = \sum_{l=0}^{L} w_l E_l$$

最后将 content 和 style 的 cost 相结合，最终可以得到：

$$L_{total}(p,a,x) = \alpha L_{content}(p,x) + \beta L_{style}(a,x)$$

α,β 表示权值，在建立 $L_{content}$ 的时候，用到了 VGG-19 的 conv4_2 层，而在建立 L_{style} 的时候，用到了VGG-19 的 conv1_1, conv2_1, conv3_1, conv4_1 以及 conv5_1。

实现代码

https://github.com/ckmarkoh/neuralart_tensorflow