目标检测 Notebook

叶亮

2021 年 10 月 15 日

目录

1 Base

1.1 基本知识点

离散型随机变量：

期望：$E(X) = \sum_{i=1}^{n} x_i p_i$

方差：$D(X) = \sum_{i=1}^{n} [x_i - E(x)]^2 p_i$

连续型随机变量：

期望：$\int_{-\infty}^{\infty} x f(x) dx$

方差：$\int_{-\infty}^{\infty} [x - E(x)]^2 f(x) dx$

1.2 distribution

1.2.1 伯努利分布

Bernoulli distribution，又称**两点分布**或 **0-1 分布**，是一个离散型概率分布。记取值为 1 的概率为 p，取值为 0 的概率为 $q = 1 - p$, 概率质量函数为

$$f(x;p) = p^x (1-p)^{1-x} = \begin{cases} p, & if\ x = 1 \\ q = 1 - p, & if\ x = 0 \end{cases} \tag{1}$$

期望为

$$E[X] = P(X = 1) \cdot 1 + P(X = 0) \cdot 0 = p + 0 = p \tag{2}$$

方差

$$Var(X) = \sum_{i=0}^{1} (x_i - E[X])^2 f(x) = (0-p)^2(1-p) + (1-p^2)p = p(1-p) = pq \tag{3}$$

1.2.2 二项分布

二项分布 (Binomial distribution) 是 n 个独立的是/非试验中成功次数的离散概率分布，每次试验成功率为 p。单次的试验为伯努利试验，即 **n=1** 时，二项分布为**伯努利分布**。

如果变量 X 服从参数为 n 和 p 的二项分布，则记为 $X \sim b(n,p) or X \sim B(n,p)$。n 次试验中正好得到 k 次成功的概率由概率质量函数给出：

$$f(k,n,p) = Pr(X = k) = \binom{n}{k} p^k (1-p)^{n-k}\ where \binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{4}$$

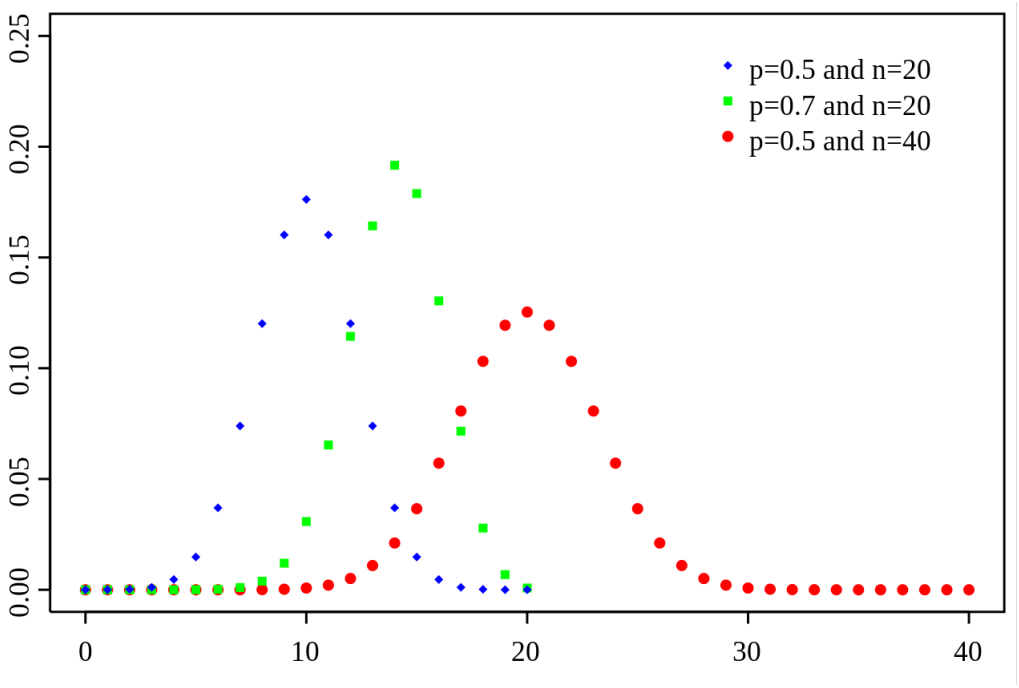


图 1: 不同参数下的二项分布

期望和方差为

$$E(x) = np \tag{5}$$

$$Var(x) = np(1-p) \tag{6}$$

1.3 cross-entropy

熵是表示**随机变量不确定性的度量**. 设 X 是一个取有限值的离散随机变量，其概率分布为

$$P(X = x_i) = p_i, i = 1, 2, ..., n \tag{7}$$

则随机变量 X 的熵定义为

$$H(X) = -\sum_{i=1}^{n} p_i \log p_i \tag{8}$$

根据定义可知，熵的大小只依赖于 X 的分布，而与 X 的取值无关，所以可以将 X 的熵记为 $H(p)$，即

$$H(p) = -\sum_{i=1}^{n} p_i \log p_i \tag{9}$$

熵越大，则说明随机变量的不确定性越大，因此需要更多的信息去验证真假。当 X 为二元变量时，例如 1，0，则熵为

$$H(p) = -p \log p - (1-p) \log (1-p) \tag{10}$$

当 $p=0$ 或 $p=1$ 时,$H(p)=0$，随机变量完全没有不确定性。当 $p=0.5$ 时，$H(p)=1$，熵取最大值，随机变量不确定性最大。

1.4 神经网络相关知识

1.4.1 卷积相关

参考论文:A guide to convolution arithmetic for deep learning.

1.4.2 感受野

感受野 (Receptive field)，用来表示网络内部的不同位置的神经元对原图像的感受范围的大小。神经元感受野的值越大表示其能接触到的原始图像范围就越大，也意味着他可能蕴含更为全局、语义层次更高的特征；而值越小则表示其所包含的特征越趋向于局部和细节。因此感受野的值可以大致用来判断每一层的抽象层次。参考博客：a-guide-to-receptive-field。

先介绍卷积的输出特征图尺寸计算公式

$$n_{out}=\lfloor\frac{n_{in}+2p-k}{s}\rfloor+1 \tag{11}$$

其中，n 为特征图尺寸大小，k 为卷积核尺寸,p 为 padding 尺寸，s 为步长。

感受野的可视化方式有两种，如图2所示，左边为常用的方式，右边为固定尺寸的可视化方式。右边通过固定尺寸的方式，可以更直观地将深层特征图的特征对应到之前特征图中的感受野中心位置。

感受野的计算公式一（常用方式）：

$$RF_{l+1}=RF_l+(k_{l+1}-1)\times features\_stride_l \tag{12}$$

其中,RF 为感受野大小，l 表示层数，$features\_stride_l=\prod_{i=1}^{l}s_i$,$l=0$ 表示输入层，$RF_0=1$,$features\_stride_0=1$.

如果包含空洞卷积的话，则公式为

$$RF_{l+1}=RF_l+(k_{l+1}-1)\times features\_stride_l\times dilation_{l+1} \tag{13}$$

感受野的计算公式二：

$$\begin{aligned} j_{out}&=j_{in}*s\\ r_{out}&=r_{in}+(k-1)*j_{in}\\ start_{out}&=start_{in}+(\frac{k-1}{2}-p)*j_{in} \end{aligned} \tag{14}$$

其中，第一行公式计算输出特征图的 jump 大小，即每个特征之间的间距。第二行计算输出特征图的感受野尺寸。第三行计算第一个输出特征的感受野的中心位置，等于第一个输入特征的中心点加上第一个输入特征与第一个卷积中心的距离 $\frac{k-1}{2}*j_{in}$，并减取 padding 空间 $p*j_{in}$。

第一层为输入层，即图像本身,$n=image\ size, r=1, j=1, start=0.5$。图3所示为具体计算过程。

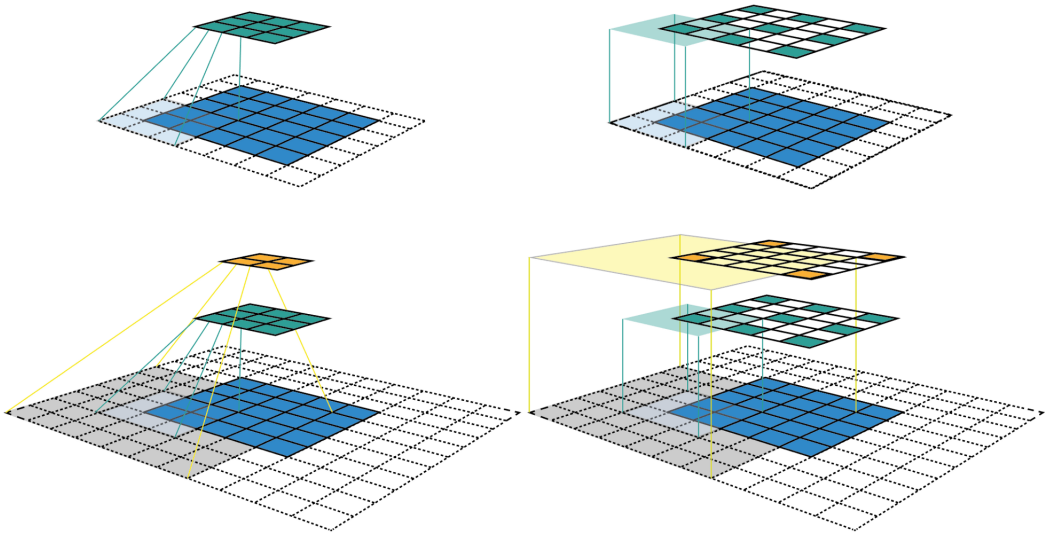


图 2: 两种感受野的可视化方式

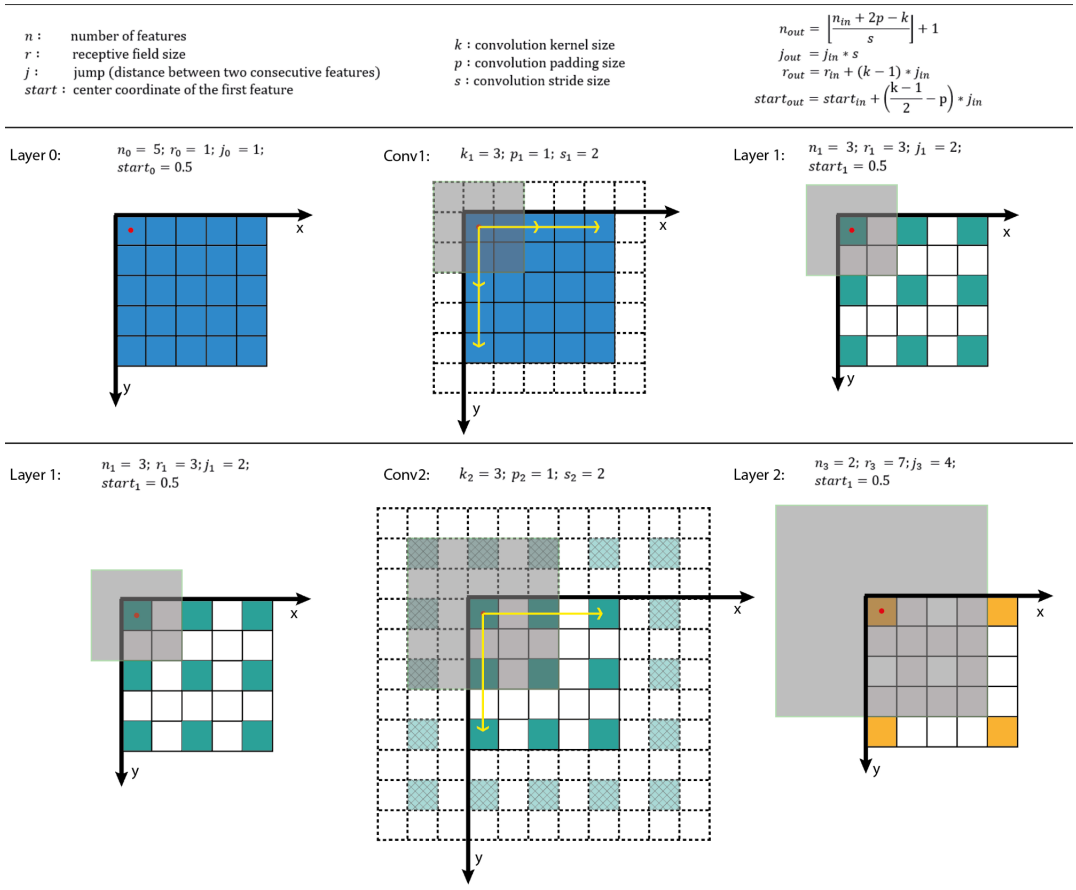


图 3: 感受野的计算过程

2 Loss

2.1 IoU loss

IoU Loss 的基本计算公式为:

$$\mathcal{L}_{IoU} = 1 - IoU \tag{15}$$

其中, IoU 为预测和 GT 框的交并比，其他改进后的版本大多式在此基础上加入额外的惩罚项来有针对性地引导模型。其优点为:

1. 可以反映预测检测框与真实检测框的检测效果。

2. 尺度不变性，也就是对尺度不敏感（scale invariant），在 regression 任务中，判断 predict box 和 gt 的距离最直接的指标就是 IoU。(满足非负性；同一性；对称性；三角不等性)。

缺点：

1. 如果两个框没有相交，根据定义，IoU=0，不能反映两者的距离大小（重合度）。同时因为 loss=0，没有梯度回传，无法进行学习训练。

2. IoU 无法精确的反映两者的重合度大小。如下图所示，三种情况 IoU 都相等，但看得出来他们的重合度是不一样的，左边的图回归的效果最好，右边的最差。

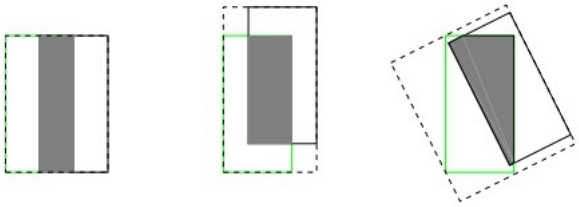


Figure 2. Three different ways of overlap between two rectangles with the exactly same IoU values, *i.e.* $IoU = 0.33$, but different $GIoU$ values, *i.e.* from the left to right $GIoU = 0.33, 0.24$ and -0.1 respectively. $GIoU$ value will be higher for the cases with better aligned orientation.

图 4: 相同 IoU 的不同情况

2.2 GIoU loss

GIoU loss 中的 IoU 部分替换成了 GIoU, 其计算方式如下:

$$GIoU = IoU - \frac{|A_c - U|}{|A_c|} \tag{16}$$

其中，A_c 表示两个框的最小闭包区域面积（同时包含预测和 GT 的最小框），然后计算闭包区域中不属于两个框的区域所占比重。特性：

1. 与 IoU 相似，GIoU 也是一种距离度量，作为损失函数的话，[公式] , 满足损失函数的基本要求；

2. GIoU 对 scale 不敏感；

3. GIoU 是 IoU 的下界，在两个框无限重合的情况下，IoU=GIoU=1；

4. IoU 取值 [0,1]，但 GIoU 有对称区间，取值范围 [-1,1]。在两者重合的时候取最大值 1，在两者无交集且无限远的时候取最小值-1，因此 GIoU 是一个非常好的距离度量指标。

5. 与 IoU 只关注重叠区域不同，GIoU 不仅关注重叠区域，还关注其他的非重合区域，能更好的反映两者的重合度。

2.3 DIoU loss

DIoU 要比 GIou 更加符合目标框回归的机制，将目标与 anchor 之间的距离，重叠率以及尺度都考虑进去，使得目标框回归变得更加稳定，不会像 IoU 和 GIoU 一样出现训练过程中发散等问题。

$$DIoU = IoU - \frac{\rho^2(b, b^{gt})}{c^2} \tag{17}$$

其中，$\rho()$ 为欧式距离计算，b 为中心点坐标，c 为最小闭包的对角线长度:$c^2 = cw^2 + ch^2$. Code optimization, the calculation of distance between two center points of bboxes. C 包含 x 和 y，以下公式为 x 计算部分，y 部分同理，最后求和.

$$\begin{aligned}(C_1 - C_2)^2) &= ((x1_{C1} + x2_{C1})/2 - (x1_{C2} + x2_{C2})/2)^2 \\ &= ((x1_{C1} + x2_{C1}) - (x1_{C2} + x2_{C2}))^2/4\end{aligned} \tag{18}$$

特性:

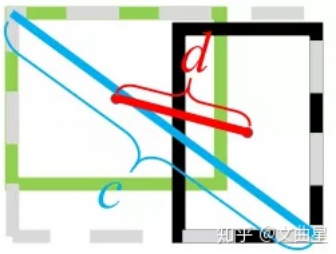


图 5: DIoU: 对 anchor 框和目标框之间的归一化距离进行了建模

1. 与 GIoU loss 类似，DIoU loss($\mathcal{L}_{DIoU} = 1 - DIoU$) 在与目标框不重叠时，仍然可以为边界框提供移动方向。

2. DIoU loss 可以直接最小化两个目标框的距离，因此比 GIoU loss 收敛快得多。3. 对于包含两个框在水平方向和垂直方向上这种情况，DIoU 损失可以使回归非常快，而 GIoU 损失几乎退化为 IoU 损失。4. DIoU 还可以替换普通的 IoU 评价策略，应用于 NMS 中，使得 NMS 得到的结果更加合理和有效。

2.4 CIoU loss

论文考虑到 bbox 回归三要素中的长宽比还没被考虑到计算中，因此，进一步在 DIoU 的基础上提出了 CIoU。其惩罚项如下面公式：

$$\mathcal{R}_{CIoU} = \frac{\rho^2(b, b^{gt})}{c^2} + \alpha v \tag{19}$$

其中，v 用来度量长宽比的相似性，定义为:$v = \frac{4}{\pi^2}(\arctan\frac{w^{gt}}{h^{gt}} - \arctan\frac{w}{h})^2$，$\alpha$ 为权重函数，定义为:$\alpha = \frac{v}{(1-IoU)+v}$。最后，CIoU loss 的梯度类似于 DIoU loss，但还要考虑 v 的梯度。在长宽在 $[0, 1]$ 的情况下，对 arctan 进行求导时 $w^2 + h^2$ 的值通常很小，会导致梯度爆炸，因此在 $\frac{1}{w^2+h^2}$ 实现时将替换成 1。在使用 pytorch 实现时，计算 α 时使用 torch.no_grad() 来避免计算 α 的损失，以此来保证合理的收敛过程。

2.5 BIoU loss

3 Activation

3.1 Sigmoid

Sigmoid 函数，其公式如下:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{20}$$

$$\sigma'(x) = \sigma(x) \cdot (1 - \sigma(x)) \tag{21}$$

在什么情况下适合使用 Sigmoid 激活函数呢?

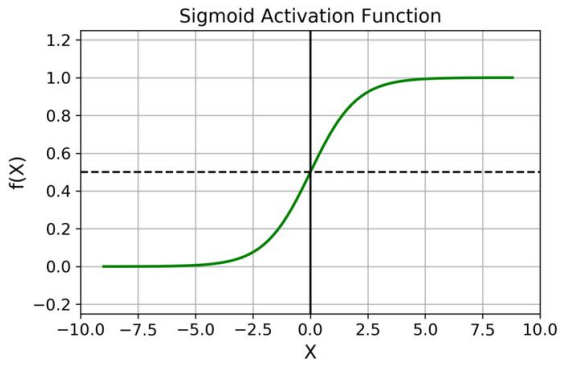


图 6: Sigmoid 函数

1. Sigmoid 函数的输出范围是 0 到 1。由于输出值限定在 0 到 1，因此它对每个神经元的输出进行了归一化；

2. 用于将预测概率作为输出的模型。由于概率的取值范围是 0 到 1，因此 Sigmoid 函数非常合适；梯度平滑，避免「跳跃」的输出值；

3. 函数是可微的。这意味着可以找到任意两个点的 sigmoid 曲线的斜率；

4. 明确的预测，即非常接近 1 或 0。

Sigmoid 激活函数有哪些缺点？

1. 倾向于梯度消失；

2. 函数输出不是以 0 为中心的，这会降低权重更新的效率；

3. Sigmoid 函数执行指数运算，计算机运行得较慢。

3.2 Tanh

双曲正切函数，其图形也是 S 形，公式如下:

$$tanh(x) = \frac{2}{1+e^{-2x}} - 1 = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} \tag{22}$$

$$tanh'(x) = 1 - tanh^2(x) \tag{23}$$

相比 sigmoid, tanh 的优势体现在:

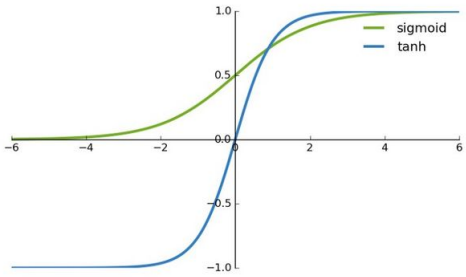


图 7: Tanh 函数

首先，当输入较大或较小时，输出几乎是平滑的并且梯度较小，这不利于权重更新。二者的区别在于输出间隔，tanh 的输出间隔为 1，并且整个函数以 0 为中心，比 sigmoid 函数更好；

tanh 函数的缺点同 sigmoid 函数的第一个缺点一样，当 x 很大或很小时，导数接近于 0 ，会导致梯度很小，权重更新非常缓慢，即梯度消失问题。

在一般的二元分类问题中，tanh 函数用于隐藏层，而 sigmoid 函数用于输出层.

3.3 ReLU

ReLU 的公式如下:

$$relu(x) = \begin{cases} x, x >= 0 \\ 0, x < 0 \end{cases} \tag{24}$$

ReLU6 的公式如下，这是为了在移动端设备 float16/int8 的低精度的时候也能有很好的数值分辨率:

$$\text{ReLU6}(x) = \min(\max(0, x), 6)$$

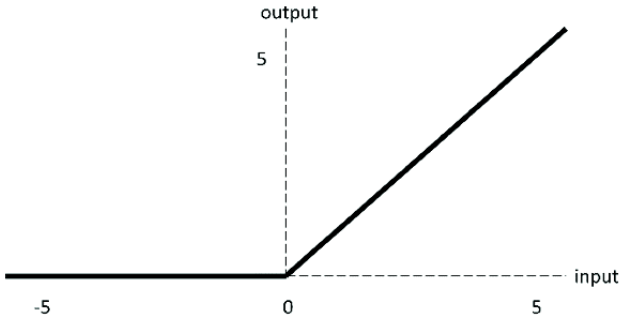


图 8: ReLU 函数

优点：输入为正时，不存在梯度饱和问题；计算速度快。

缺点：

Dead ReLU 问题。当输入为负时，ReLU 完全失效，在正向传播过程中，这不是问题。有些区域很敏感，有些则不敏感。但是在反向传播过程中，如果输入负数，则梯度将完全为零，sigmoid 函数和 tanh 函数也具有相同的问题；

输出为 0 或正数，这意味着 ReLU 函数不是以 0 为中心的函数。

3.4 Leaky ReLU & Parametric ReLU

$$LReLU(x) = \begin{cases} x, x > 0 \\ ax, x <= 0 \end{cases} \tag{25}$$

特性:

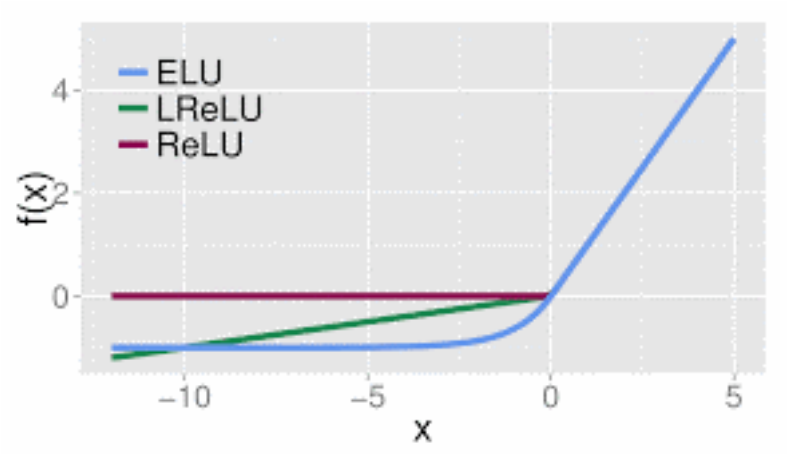


图 9: ReLU、Leaky ReLU、ELU

Leaky ReLU 通过把 x 的非常小的线性分量给予负输入（0.01x）来调整负值的零梯度（zero

gradients）问题；如果 a 是可以学习的参数，则为 **PReLU**(Parametric ReLU).

leak 有助于扩大 ReLU 函数的范围，通常 a 的值为 0.01 左右；

Leaky ReLU 的函数范围是（负无穷到正无穷）。

3.5 ELU

ELU 的提出也解决了 ReLU 的问题。与 ReLU 相比，ELU 有负值，这会使激活的平均值接近零。均值激活接近于零可以使学习更快，因为它们使梯度更接近自然梯度。

$$\mathrm{ELU}(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha * (\exp(x) - 1), & \text{if } x \leq 0 \end{cases} \tag{26}$$

ELU 具有 ReLU 的所有优点，并且：

1. 没有 Dead ReLU 问题，输出的平均值接近 0，以 0 为中心；

2. ELU 通过减少偏置偏移的影响，使正常梯度更接近于单位自然梯度，从而使均值向零加速学习；

3. ELU 在较小的输入下会饱和至负值，从而减少前向传播的变异和信息。

一个小问题是它的计算强度更高。与 Leaky ReLU 类似，尽管理论上比 ReLU 要好，但目前在实践中没有充分的证据表明 ELU 总是比 ReLU 好。

3.6 SELU

SELU 是在 ELUde 基础上引入了一个 λ 系数。当其中参数取为 $\lambda \approx 1.0507, \alpha \approx 1.6733$ 时，在网络权重服从标准正态分布的条件下，各层输出的分布会向标准正态分布靠拢。这种" 自我标准化" 的特性可以避免梯度消失和爆炸的问题。

论文提出时用于 feed-forward neural network 中，构成 Self-Normalizing Neural Networks(SNN). 其主要作用在于将神经元的值归一化到 0 均值单位方差范围。以便于网络更好的收敛。

3.7 SiLU / Swish

Sigmoid Linear Unit

$$\mathrm{silu}(x) = x * \sigma(x), \text{where } \sigma(x) \text{ is the logistic sigmoid.} \tag{27}$$

Swish 的设计受到了 LSTM 和高速网络中 gating 的 sigmoid 函数使用的启发。使用相同的 gating 值来简化 gating 机制，这称为 self-gating。self-gating 的优点在于它只需要简单的标量输入，而普通的 gating 则需要多个标量输入。这使得诸如 Swish 之类的 self-gated 激活函数能够轻松替换以单个标量为输入的激活函数（例如 ReLU），而无需更改隐藏容量或参数数量。Swish 激活函数的主要优点如下：

「无界性」有助于防止慢速训练期间，梯度逐渐接近 0 并导致饱和；(同时，有界性也是有优势的，因为有界激活函数可以具有很强的正则化，并且较大的负输入问题也能解决); 平滑度在优化和泛化中起了重要作用。

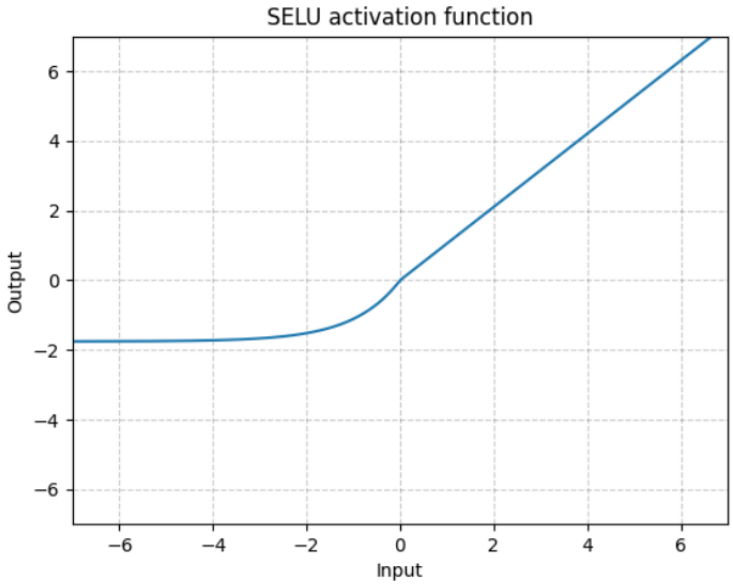


图 10: SELU

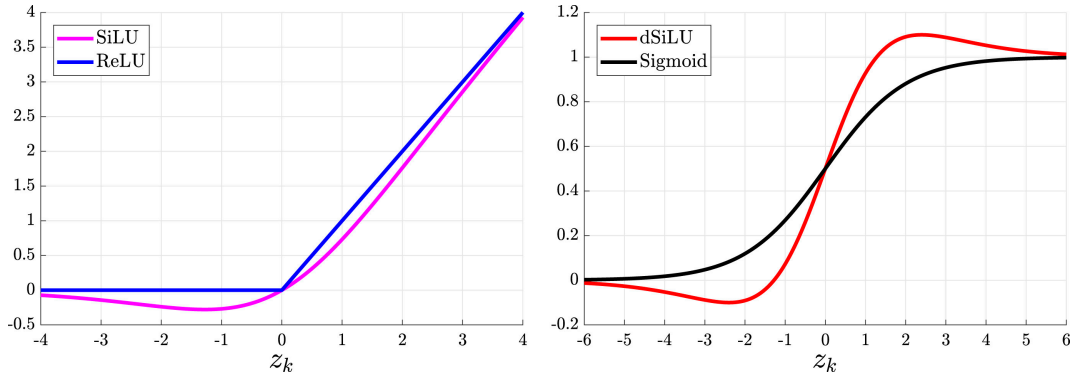


图 11: The activation functions of the SiLU and the ReLU (left panel), and the dSiLU and the sigmoid unit (right panel)

3.8 Hard swish

$$\text{Hardswish}(x) = \begin{cases} 0 & \text{if } x \leq -3, \\ x & \text{if } x \geq +3, \\ x \cdot (x+3)/6 & \text{otherwise} \end{cases} \tag{28}$$

主要是解决移动端下精度较低时的计算问题。

4 mmdetection

4.1 anchor

mmdetection 中，在 retinanet, R-CNN 等非 ssd 系列检测器的 grid anchor 生成中，在 grid 中每个 anchor 的坐标表示为 (xmin,ymin,xmax,ymax)。anchor 参数包含 strides, ratios, scales, base sizes 等. 其中 strides 为特征图的步长，ratios 为宽高比，scales 为 anchor 的缩放因子。计算过程如下:

1. 根据 featmap 生成 grid. 其中 grid 坐标为 i * featmap size for i in grid size.

2. grid 中每个 cell 根据 ratios, scales 和 base sizes 计算 anchors,$xmin = grid_i - basesize * scales * ratios, xmax = grid_i + basesize * scales * ratios$. 每个 anchor 的形式为 (xmin,ymin,xmax,ymax), 且均

具体计算方式可参考 **core/anchor/anchor_generator.py** 中部分。

4.2 coder

在 RetinaNet、SSD、Cascade R-CNN 等网络中，网络预测的 bbox 都会进行 Delta xywh 编码，即将原始的 (xmin, ymin, xmax, ymax) 进行编码, 计算 proposals 相对 GT 的距离，得到 (dx,dy,dw,dh)，来减小回归的过拟合，提升回归的稳定性。其中编码后的 x 为中心点。

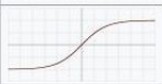
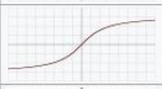
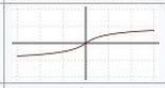
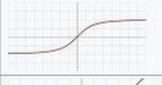
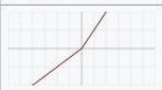
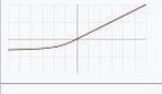
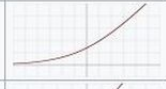
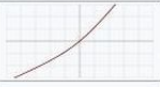
$$\begin{aligned} \delta_x &= (g_x - p_x)/p_w \quad \delta_y = (g_y - p_y)/p_h \\ \delta_w &= \log \frac{g_w}{p_w} \qquad\qquad\quad \delta_h = \log \frac{g_h}{p_h} \end{aligned} \tag{29}$$

上述公式计算得到的 δ 值通常很小，因为网络通常只对 p 进行少量微调，导致回归 loss 比分类 loss 小很多。为了提升学习的有效性，δ 通常需要经过均值和方差进行标准化.

$$\delta_x^{'} = \frac{\delta_x - \mu_x}{\rho_x} \tag{30}$$

4.3 two-stage

mmdetection 中的二阶段检测网络, 首先使用 RPN-FPN 网络, 得到多个 level 的特征图, 然后使用多个 level 的特征图生成 proposals, 并使用 NMS 进行过滤。然后利用过滤后的 proposals, 在对

Name	Plot	Equation
Identity		$f(x) = x$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Logistic (a.k.a. Sigmoid or Soft step)		$f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$ [1]
TanH		$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$
ArcTan		$f(x) = \tan^{-1}(x)$
Softsign[9][10]		$f(x) = \frac{x}{1+\lvert x\rvert}$
Inverse square root unit (ISRU)[11]		$f(x) = \frac{x}{\sqrt{1+\alpha x^2}}$
Rectified linear unit (ReLU)[12]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Leaky rectified linear unit (Leaky ReLU)[13]		$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Parameteric rectified linear unit (PReLU)[14]		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Randomized leaky rectified linear unit (RReLU)[15]		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ [3]
Exponential linear unit (ELU)[16]		$f(\alpha, x) = \begin{cases} \alpha(e^x - 1) & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases}$
Scaled exponential linear unit (SELU)[17]		$f(\alpha, x) = \lambda \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ with $\lambda = 1.0507$ and $\alpha = 1.67326$
S-shaped rectified linear activation unit (SReLU)[18]		$f_{t_l, a_l, t_r, a_r}(x) = \begin{cases} t_l + a_l(x - t_l) & \text{for } x \leq t_l \\ x & \text{for } t_l < x < t_r \\ t_r + a_r(x - t_r) & \text{for } x \geq t_r \end{cases}$ t_l, a_l, t_r, a_r are parameters.
Inverse square root linear unit (ISRLU)[11]		$f(x) = \begin{cases} \frac{x}{\sqrt{1+\alpha x^2}} & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Adaptive piecewise linear (APL)[19]		$f(x) = \max(0, x) + \sum_{s=1}^{S} a_i^s \max(0, -x + b_i^s)$
SoftPlus[20]		$f(x) = \ln(1 + e^x)$
Bent identity		$f(x) = \frac{\sqrt{x^2+1}-1}{2} + x$
Sigmoid-weighted linear unit (SiLU)[21] (a.k.a. Swish[22])		$f(x) = x \cdot \sigma(x)$ [5]
SoftExponential[23]		$f(\alpha, x) = \begin{cases} -\frac{\ln(1-\alpha(x+\alpha))}{\alpha} & \text{for } \alpha < 0 \\ x & \text{for } \alpha = 0 \\ \frac{e^{\alpha x}-1}{\alpha} + \alpha & \text{for } \alpha > 0 \end{cases}$
Sinusoid[24]		$f(x) = \sin(x)$
Sinc		$f(x) = \begin{cases} 1 & \text{for } x = 0 \\ \frac{\sin(x)}{x} & \text{for } x \neq 0 \end{cases}$
Gaussian		$f(x) = e^{-x^2}$

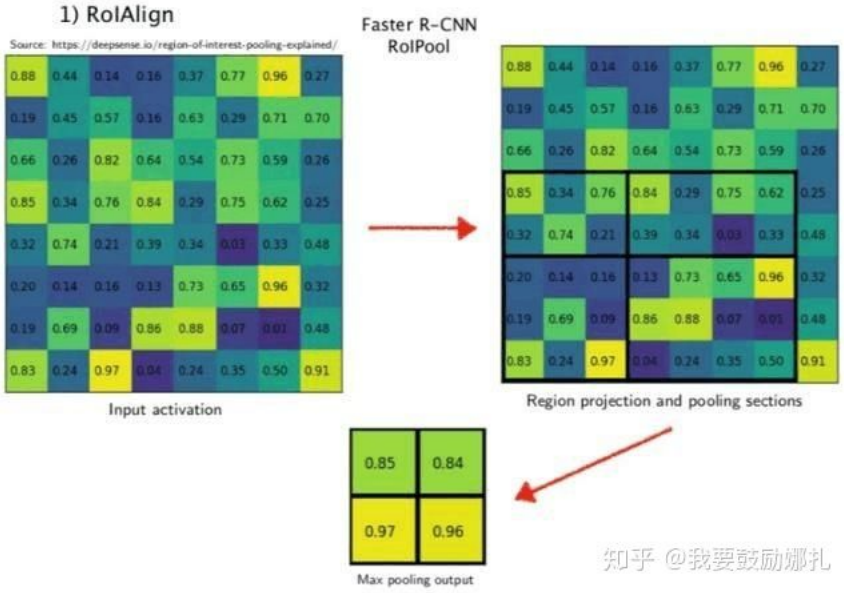


图 13: ROI-Pool 过程

应特征图上进行 ROIAlign 和 ROIPool. 在 RPN 阶段，计算 loss 时分类 loss 只计算 class-agnostic loss，即采样到的 gt targets 为正的框。

4.3.1 ROI Pool

ROI pool 根据生成的 proposals,(xyxy), 在对应的 feature maps 上进行裁剪，将对应坐标内的特征图分成 $s*s$ 块状网格，每个网络里面可能由多个值，然后使用最大池化对每个网络内的多个值进行池化，得到 $s*s$ 的尺寸输出。在对特征图进行分块时，ROI pool 使用直接取整的方式，以一个 $8*8$ 的特征图为例，输出 $2*2$ 的池化后的特征图。假设 (xyxy) 为 (0,3,7,8), 则 proposal 的原始 h,w 为 7,5. 则 1)$5/2 = 2.5$ –> 2, 剩下的为 3，则 2+3 向下取整 floor; 2)$7/2 = 3.5$ –>3，剩下的为 4，则 3+4

5 training

5.1 Optimizer

5.1.1 warmup

warmup 通常有三个方式:linear, constant, exp. 通常需要设置 warmup 的迭代数 $iter_{total}$ 和 warmup 的增加比率 ratio,

$$lr_t = lr_{constant} * ratio \tag{31}$$

$$\begin{aligned} lr_t &= lr_{const} * (1-k), \\ k &= (1 - iter_t/iter_{total}) * (1 - ratio) \end{aligned} \tag{32}$$

$$lr_t = lr_{const} * k, \tag{33}$$

$$k = ratio^{1-iter_t/iter_{total}} \tag{34}$$

6 Yolov4

Yolov4 的实现过程中,anchor 的生成以及 label 与 anchor 的对应关系的构建方法。如图14. 所示。

6.1 Anchor generation & target build

与 retinanet，rcnn 系列等 bbox 预测不同的是，yolo 模型推理得到的结果（x,y,w,h) 需要经过如下公式进行编码：

$$\begin{aligned} b_x &= \sigma(t_x) + c_x \\ b_y &= \sigma(t_y) + c_y \\ b_w &= p_w e^{t_w} \\ b_h &= p_h e^{t_h} \end{aligned} \tag{35}$$

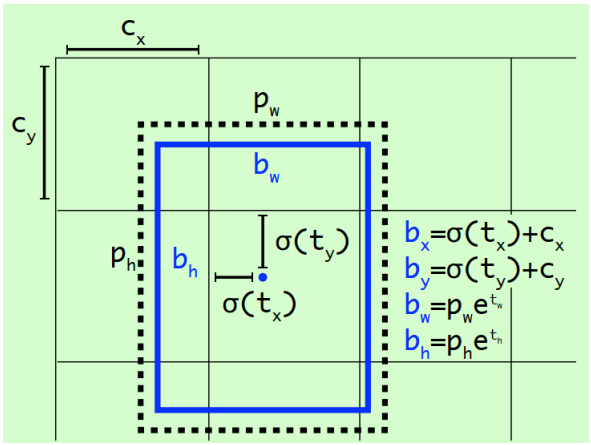


图 14: Yolo bbox prediction

7 Yolov5

7.1 Model

Focus 结构: Focus 模块中，首先将一幅图，按照隔点采样 (::2,1::2) 的方式，将一个图片分成四幅图，然后将这四幅图拼接到一块，形成 $3 * 4 = 12$ 通道的特征图，再做卷积操作，这样来实现下采样和卷积操作。

C3: CSP bottleneck with 3 convolution[1]. 其中 CSP 模块如图15所示，采用 fusion first 方式，通过 1x1 的卷积将 base layer 分成均等的两部分特征图，其中 part 1 和 part2 为 hidden channels，为最终输出通道的 1/2.C3 中的 bottleneck 为标注的残差模块，使用 1x1-3x3 和 shortcut 实现。

Conv:conv 模块中包含了 conv-bn-act 三部分，其中 act 使用 SiLU 激活函数，在特征提取的下采样过程中，没有使用 maxpool，而是采用了 s=2 的 conv 来实现。

head:yolov5 检测头与 yolov3, 4 系列一样，每一层输出的通道为 (num classes + 5(xywh,conf)) * num anchors, e.g 80 类别，三个 anchor, 则输出通道为 (80+5)*3 = 255. 对最后的预测特征图使用 sigmoid 归一化到 0-1 区间.

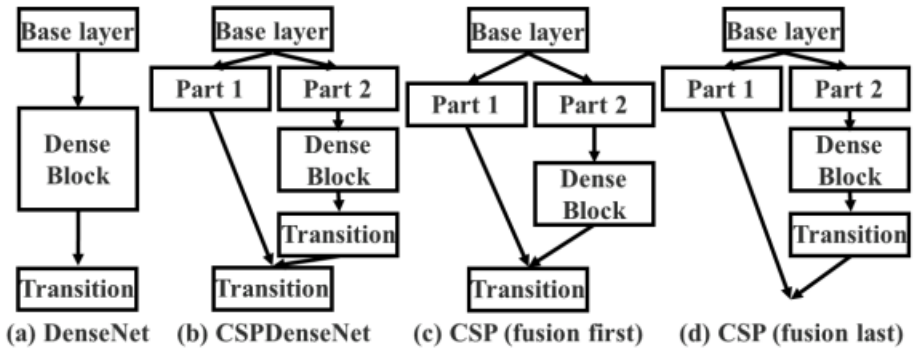


图 15: CSP 模块

7.2 Loss

7.3 Inference

在 yolov5 的推理部分中，预测 pred 的通道为 xywh,obj_conf, nc。首先，使用 conf_thres 对 obj_conf 进行过滤，保留一定数目的 bbox, 然后对所有的类别概率进行重计算 $nc\_conf = obj\_conf * nc\_conf$,

8 backbone

8.1 Regularization

8.1.1 Dropout

原理,dropout 随机丢弃神经元 (全连接中输入神经元)，实现方式为 $keep_prob$, 每个神经元生成一个随机数 $k, k < keep_prob$ 即丢弃。优点, 该方法有利于分类中泛化能力的提升.

8.1.2 Drop Connect

8.1.3 Drop block

9 Refinedet

9.1 Anchor

Refine 中的 anchor 计算。对于每一个 feature map, 首先计算其 mesh grid, 然后计算每个框的中心点 $(x, y) = (\frac{i+0.5}{feat\_size}, \frac{j+0.5}{feat\_size})$, 然后根据每个 feature map 对应的 anchor box 的大小，计算 anchor 的长和宽 $WH_{ki} = \frac{box_k}{image\_size}$, k 表示第 k 层特征图,i 表示第 i 个网格. e.g, 使用四个 feature level, $box = [32, 64, 128, 256], image\_size = 320, feat\_size = [40, 20, 10, 5], aspect\_ratio = [2, 2, 2, 2]$, 那么最终生成 $40 * 40 * 3 + 20 * 20 * 3 + 10 * 10 * 3 + 5 * 5 * 3 = 6375$ 个 anchor.

9.2 Loss

计算过程分为 arm 和 odm，其中 arm 预测分别输出 $num\_anchor * 4$ 和 $num\_anchor * 2$ 通道的特征图 (坐标，是否包含目标)；odm 预测分别输出 $num\_anchor * 4$ 和 $num\_anchor * num\_classes$ 通道的特征图 (坐标, 类别数量)。每层特征图的目标数量则为 $N_i * H_i * num\_anchor$。

在 loss 计算过程中，分别计算分类 loss 和回归 loss. 其中分类使用交叉熵，回归使用 smooth l1

10 Dataset

10.1 COCO Dataset

COCO dataset 全称 Common Objects in Context. 共有 80 个类。共有三种标注类型：object instance, object keypoints, image captions。使用 json 文件进行存储。

10.1.1 object instance

"info": info, "licenses": [license], "images": [image], "annotations": [annotation], "categories": [category]

11 Detector

11.1 CornerNet, CenterNet

11.1.1 targets computation

在 anchor free 网络中，生成 gt bbox 的 heatmap target 时，相应的高斯半径的计算方式如下。一共存在三种情况：1，生成的 target 与 gt bbox 有重叠部分，其中一个 corner 在 gt box 内

部，另一个在 gt box 外部。

$$\frac{(w-r)*(h-r)}{w*h+(w+h)r-r^2} \geq iou \quad \Rightarrow \quad r^2-(w+h)r+\frac{1-iou}{1+iou}*w*h \geq 0 \tag{36}$$

$$a=1, \quad b=-(w+h), \quad c=\frac{1-iou}{1+iou}*w*h \tag{37}$$

$$r \leq \frac{-b-\sqrt{b^2-4*a*c}}{2*a} \tag{38}$$

2. 两个 corner 都在 gt box 里面

$$\frac{(w-2*r)*(h-2*r)}{w*h} \geq iou \quad \Rightarrow \quad 4r^2-2(w+h)r+(1-iou)*w*h \geq 0 \tag{39}$$

$$a=4, \quad b=-2(w+h), \quad c=(1-iou)*w*h \tag{40}$$

$$r \leq \frac{-b-\sqrt{b^2-4*a*c}}{2*a} \tag{41}$$

3. 两个 corner 都在 gt box 外部

$$\frac{w*h}{(w+2*r)*(h+2*r)} \geq iou \quad \Rightarrow \quad 4*iou*r^2+2*iou*(w+h)r+(iou-1)*w*h \leq 0 \tag{42}$$

$$a=4*iou, \quad b=2*iou*(w+h), \quad c=(iou-1)*w*h \tag{43}$$

$$r \leq \frac{-b+\sqrt{b^2-4*a*c}}{2*a} \tag{44}$$

高斯核的计算

$$\exp\frac{-(x*x+y*y)}{2*\sigma*\sigma} \tag{45}$$

11.1.2 decode heatmap

对 heatmap 进行解码，遵循如下顺序:1. nms on heatmap. 2. get topk positions from heatmap. 3. decode offset and wh size.

12 Optical Character Recognition

12.1 DBNet

Real-Time Scene Text Detection with Differentiable Binarization[2].

12.1.1 Related work

最近的 OCR 可分为两种：Regression-based and segmentation-based.

Regression-based methods 直接回归文本实例的坐标框, 使用 NMS 后处理。大部分模型达不到精准的文本框检测（非矩形，非旋转矩形等），尤其是弯曲的文本位置。

Segmentation-based methods 结合 Pixel-level prediction and post-processing 来获取文本位置。

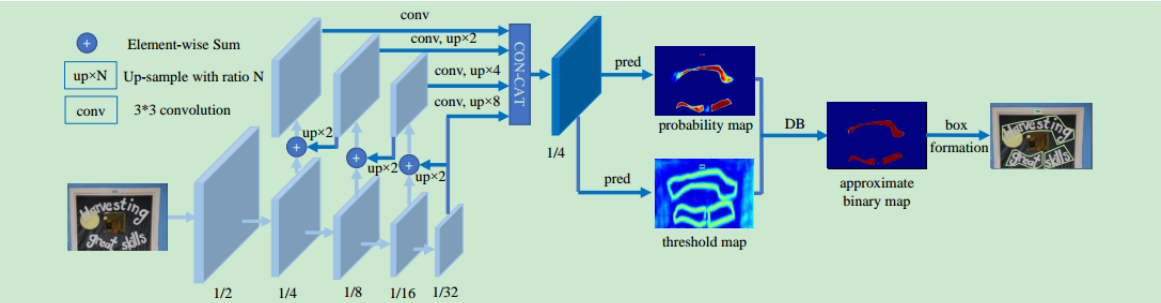


Figure 3: Architecture of our proposed method, where "pred" consists of a 3×3 convolutional operator and two de-convolutional operators with stride 2. The "1/2", "1/4", ... and "1/32" indicate the scale ratio compared to the input image.

图 16: Architecture of the DB-Net

12.1.2 Methodology

如图 Fig. 16所示，使用 FPN 作为 backbone, 在 1/4 阶段做预测，其中"pred" 由一个 3x3 卷积和两个反卷积组成。生成概率图 (P) 和阈值图 (T)，并通过 P 和 T 来计算 approximate binary map($\hat{B}$). 二值化过程可表述为如下公式：

$$B_{i,j} = \begin{cases} 1 & if\ P_{i,j} >= t, \\ 0 & otherwise. \end{cases} \tag{46}$$

Differentiable binarization 公式46不可微。所以在训练时不能通过网络对其进行优化，因此论文提出如下近似 step function:

$$\hat{B}_{i,j} = \frac{1}{1 + e^{-k(P_{i,j} - T_{i,j})}} \tag{47}$$

其中,k 表示增强因子，设置成 50.DB 提升性能可归结于梯度方向传播。以二值交叉熵为例，定义 $f(x) = \frac{1}{1+e^{-kx}}$ 为 DB 函数，其中 $x = P_{i,j} - T_{i,j}$, 正类标签的 loss l_+ 和负类标签的 loss l_- 分别为：

$$\begin{aligned} l_+ &= -\log \frac{1}{1 + e^{-kx}} \\ l_- &= -\log \left(1 - \frac{1}{1 + e^{-kx}}\right) \end{aligned} \tag{48}$$

使用链式规则可以得到如下微分：

$$\frac{\partial l_+}{\partial x} = -kf(x)e^{-kx} \tag{49}$$

$$\frac{\partial l_-}{\partial x} = kf(x) \tag{50}$$

Label Generation PSENet 生成方法：给定文本图像，每个文本区域的多边形由一系列线段进行描述：

$$G = \{S_k\}_{k=1}^{n} \tag{51}$$

其中，n 表示顶点数量，在不同的数据集中不一样。ICDAR 2015 为 4，CTW1500 为 16. 然后使用 Vatti clipping 算法将多边形 G 收缩程 G_s 得到正类区域. 其中，收缩的偏移量 D 由原多边形的周长 L 和面积 A 计算得到：

$$D = \frac{A(1-r^2)}{L} \tag{52}$$

r 为收缩比例，一般设置为 0.4. 通过类似方法，为阈值图生成标签。1. 多边形 G 通过相同的偏移 D 进行膨胀得到 G_d.2. 将 G_s 和 G_d 之间的间隙 (gap) 作为文本区域的边界，其中阈值图的标签通过计算距离 G 中最近的线段距离得到。

Optimization

Loss 表示为概率图 L_s, 二值图 L_b 和阈值图 L_t 的加权和:

$$L = L_s + \alpha \times L_b + \beta \times L_t \tag{53}$$

α 和 β 分别设置为 1.0 和 10。对 L_s 和 L_b 应用 binary cross-entropy(BCE)loss，使用 hard negative mining 来缓解正负样本的非平衡问题：

$$L_s = L_b = \sum_{i \in S_l} y_i \log x_i + (1-y_i) \log 1 - x_i \tag{54}$$

其中,S_l 为采样集，正负样本比例为 1:3.

L_t 使用 L_1 距离和来计算 loss，为膨胀后的文本多边形区域 G_d 内预测和标签的距离：

$$L_t = \sum_{i \in R_d} |y_i^* - x_i^*| \tag{55}$$

其中，R_d 为膨胀后的多边形 G_d 内的像素索引集合，y_i^* 为阈值图标签

在推理阶段，可以只用概率图或近似二值图来生成文本坐标框，其结果相似，任选一即可。论文中，为了更好的效率，使用概率图来生成文本框，这样可以移除阈值图。即，box 处理过程为三个步骤：1）从概率图/近似二值图通过固定阈值 (0.2) 来首次二值化得到二值化图；2）从二值图得到连接区域（收缩的文本区域)；3）使用偏移量 D'，Vatti clipping 算法来膨胀收缩区域。D' 计算方式为：

$$D' = \frac{A' \times r'}{L'} \tag{56}$$

其中，A' 为收缩多边形的面积；L' 为收缩多边形的周长；r' 通过经验设置为 1.5.

12.1.3 Implementation Details

数据集介绍：**SynthText**，**MLT-2017 dataset**，**ICDAR 2015 dataset**，**MSRA-TD500 dataset**，**CTW1500 dataset**，**Total-Text dataset**.

训练时，使用 SynthText 预训练 100k iteration, 然后在真实样本上微调 1200epochs。batch size 16, 使用余弦学习率下降策略。其中当前迭代的学习率为 $lr_{init} \times (1 - \frac{iter}{max\_iter})^{power}$. 初始学习率为 0.007,*power* 为 0.9.weight decay of 0.0001, momentum of 0.9.

数据增强：1）随机旋转，角度区间 $(-10°, 10°)$；2）随机裁剪；3）随机翻转。所有处理图片均 resize 到 640x640。

12.2 CRNN

文本识别模型 (Text recognition model). 包含特征提取，序列建模，统一框架转录。有四个特性：1）端到端；2）处理任意长度的序列；3）不受预定义词典限制；4）公开数据上取得较好效果；

贡献：提出了 CRNN 网络（Convolutional Recurrent neural network）.1) 可直接学习标签序列；2）直接从图像数据中学习有用信息，不需要手工特征和其他预处理；3）有 RNN 的特性，可输出标签序列；4）不受序列目标长度限制；5）效果较好；6）参数较少

相关流程解析资源：wandb-text-recognition-crnn-ctc

12.2.1 Architecture

结构如图17所示. 从下往上依次为：卷积层、循环层和转录层。conv layer 用于提取特征序列，recurrent layer 用于对特征序列的每一帧做预测；transcription layer 则将每帧预测翻译为标签序列；可以使用单个 loss 端到端训练。

Feature Sequence Extraction

使用标准 backbone 进行特征提取。所有输入图像须保持同一高度，由 CNN 生成特征向量序列，每个特征向量的特征序列从左至右在特征图上按列生成（即，第 i 个特征向量为特征图的第 i 列特征的拼接结果），每一列的宽度为一个像素。特征序列如图18所示。

Sequence Labeing

a deep bidirectional Recurrent Neural Network is built. 对于特征序列 $\mathbf{x} = x_1, ..., x_T$,recurrent layer 为特征序列的每一帧 x_t 预测一个标签分布 y_t.RNN 的优点有三：1）RNN 能有效发挥捕获上下文信息的能力。如图18所示，宽的字符需要连续的几帧来做一个完全的描述。2）RNN 能将错误部分的导数反向传播给输入；3）RNN 能处理任意长度的序列。

LSTM，包含一个 cell 和三个们，输入、输出和遗忘门。与一般 LSTM 不同，基于图像的序列，前向和后向的上下文信息都有用且互补。因此，提出采用 forward 和 backward lstm 的方式

Transcription

转录是将由 RNN 生成的每帧预测转换成标签序列的过程。数学上来将，即在每帧预测条件下，找到最大概率的标签序列。实际使用中，有两种方式：lexicon-free 和 lexicon-based, 无词汇的和基于词汇的。

1. Probability of label sequence

采用 Connectionist Temporal Classification(CTC) layer 中定义的条件概率。在每帧预测 $\mathbf{y} = y_1, ..., y_T$ 条件下，概率为标签序列 $\mathbf{l}$ 定义，且忽略了 $\mathbf{l}$ 中每个元素的位置。因此，在使用负对数概率作为目标函数来训练时，可以只需要图像和对应的标签序列，省去了人工标注每一个符号位置的麻烦。

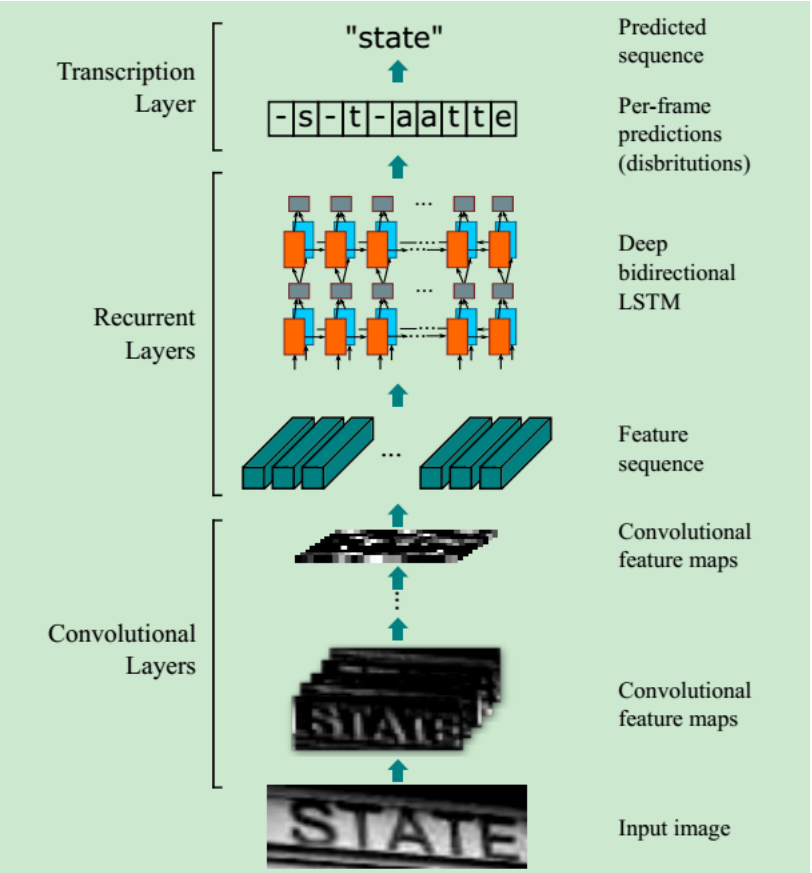


Figure 1. The network architecture. The architecture consists of three parts: 1) convolutional layers, which extract a feature sequence from the input image; 2) recurrent layers, which predict a label distribution for each frame; 3) transcription layer, which translates the per-frame predictions into the final label sequence.

图 17: CRNN 结构

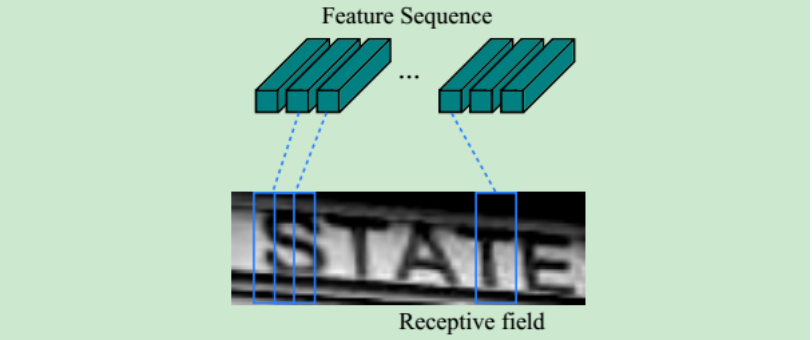


Figure 2. The receptive field. Each vector in the extracted feature sequence is associated with a receptive field on the input image, and can be considered as the feature vector of that field.

图 18: 特征序列

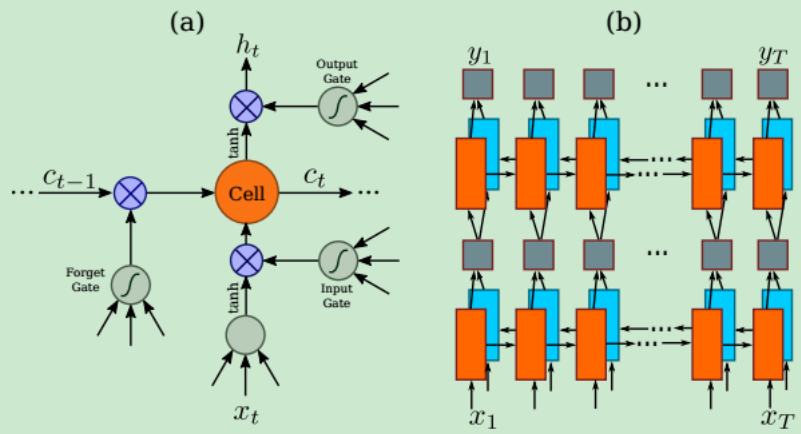


Figure 3. (a) The structure of a basic LSTM unit. An LSTM consists of a cell module and three gates, namely the input gate, the output gate and the forget gate. (b) The structure of deep bidirectional LSTM we use in our paper. Combining a forward (left to right) and a backward (right to left) LSTMs results in a bidirectional LSTM. Stacking multiple bidirectional LSTM results in a deep bidirectional LSTM.

图 19: bidirectional LSTM

公式描述如下：输入序列 $\mathbf{y} = y_1, ...y_T$,T 为序列长度。$y_t \in \mathcal{R}^{|\mathcal{L}'|}$ 为集合 $\mathcal{L}' = \mathcal{L} \cup$ 的概率分布，其中 $\mathcal{L}$ 包含任务中的所有标签（e.g. all English characters）。一个 sequence-to-sequence 映射函数 $\mathcal{B}$ 在序列 $\boldsymbol{\pi} \in \mathcal{L}'^T$ 上定义，其中 T 为长度。$\mathcal{B}$ 将 $\boldsymbol{\pi}_i$ 映射到 $\mathbf{l}$, 通过首先移除重复的标签，然后移除'blank'（空）。比如，$\mathcal{B}$ 将"–hh-e-l-ll-oo–"(-表示 blank) 映射为"hello". 则，条件概率定义为由 $\mathcal{B}$ 映射为 $\mathbf{l}$ 的所有 $\boldsymbol{\pi}$ 的概率之和:

$$p(\boldsymbol{l}|\mathbf{y}) = \sum_{\boldsymbol{\pi}:\mathcal{B}(\boldsymbol{\pi})=(l)} p(\boldsymbol{\pi}|\mathbf{y}) \tag{57}$$

其中，$p(\boldsymbol{\pi}|\mathbf{y} = \prod_{t=1}^{T} y_{\pi_t}^t, y_{\pi_t}^t$ 为 t 时刻拥有标签 π_t 的概率。

2. Lexicon-free transcription

在这种情况下，拥有最大概率的 $\mathbf{l}^*$ 将作为预测。序列近似为 $\mathbf{l}^* \approx \mathcal{B}(\arg\max_{\pi} p(\boldsymbol{\pi}|\mathbf{y}$. 即，在每个时间 t 取最大概率的标签 π_t, 并将结果映射到 $\mathbf{l}^*$

3. Lexicon-based transcription

该模式下，每一个测试样本会绑定一个词库 $\mathcal{D}$. 标签序列通过选择词典中拥有最大条件概率的序列为作为识别结果。$\mathbf{l}^* = argmax_{\mathbf{l}\in\mathcal{D}} p(\mathbf{l}|\mathbf{y})$。但是，对于大型词库，进行穷举搜索会非常耗时。对于该问题，注意到在使用 lexicon-free 时其预测的标签序列与 GT 的编辑距离相近。因此，可以限制搜索范围在一个最近邻候选区域 $\mathcal{N}_\delta(\mathbf{l}')$, 其中 δ 为最大编辑距离，$\mathbf{l}'$ 为 lexicon-free 下转录的序列:

$$\mathbf{l}^* = \arg\max_{\mathbf{l}\in\mathcal{N}_\delta(\mathbf{l}')} p(\mathbf{l}|\mathbf{y}) \tag{58}$$

Table 1. Network configuration summary. The first row is the top layer. 'k', 's' and 'p' stand for kernel size, stride and padding size respectively

Type	Configurations
Transcription	-
Bidirectional-LSTM	#hidden units:256
Bidirectional-LSTM	#hidden units:256
Map-to-Sequence	-
Convolution	#maps:512, k:2 × 2, s:1, p:0
MaxPooling	Window:1 × 2, s:2
BatchNormalization	-
Convolution	#maps:512, k:3 × 3, s:1, p:1
BatchNormalization	-
Convolution	#maps:512, k:3 × 3, s:1, p:1
MaxPooling	Window:1 × 2, s:2
Convolution	#maps:256, k:3 × 3, s:1, p:1
Convolution	#maps:256, k:3 × 3, s:1, p:1
MaxPooling	Window:2 × 2, s:2
Convolution	#maps:128, k:3 × 3, s:1, p:1
MaxPooling	Window:2 × 2, s:2
Convolution	#maps:64, k:3 × 3, s:1, p:1
Input	W × 32 gray-scale image

图 20: 模型参数

候选区域 $\mathcal{N}_\delta(\mathbf{l}')$ 可使用 BK 树数据结构来有效寻找。其时间复杂度为 $O(log|\mathcal{D}|$, 其中 $|\mathcal{D}|$ 为词库大小。

Network Training

将训练集表示为 $\mathcal{X} = I_i, \mathbf{l}_{ii}$, 其中 I_i 为图像，$\mathbf{l}_i$ 为标签序列。目标函数为:

$$\mathcal{O} = -\sum_{I_i, \mathbf{l}_i \in \mathcal{X}} \log p(\mathbf{l}_i | \mathbf{y}_i) \tag{59}$$

使用 SGD 来训练。在 recurrent layer, 使用 Back-Propagation Through Time(BPTT) 来计算误差微分。

相关参数如图20所示.

12.3 Why you should try the Real Data for the STR

Why you should try the Real Data for the Scene Text Recognition.

OCR 用途：数字化之前的实体文本，帮助盲人阅读，车牌识别等。

文本识别模型分为 4 步：transformation, feature extraction, sequence modelling and prediction. 本文使用 TPS for transofrmation, ResNeXt 作特征提取，convolutional encoder-recurrent decoder from YAMTS with 2D attention 作序列建模和预测。

12.3.1 Architecture

使用了 ASTER [3] 的图像矫正方法 (rectification) , 目前该模块已成为标注模块。使用 ResNext 作特征提取，使用 [4] 的文本检测头，具体结构如图 21所示

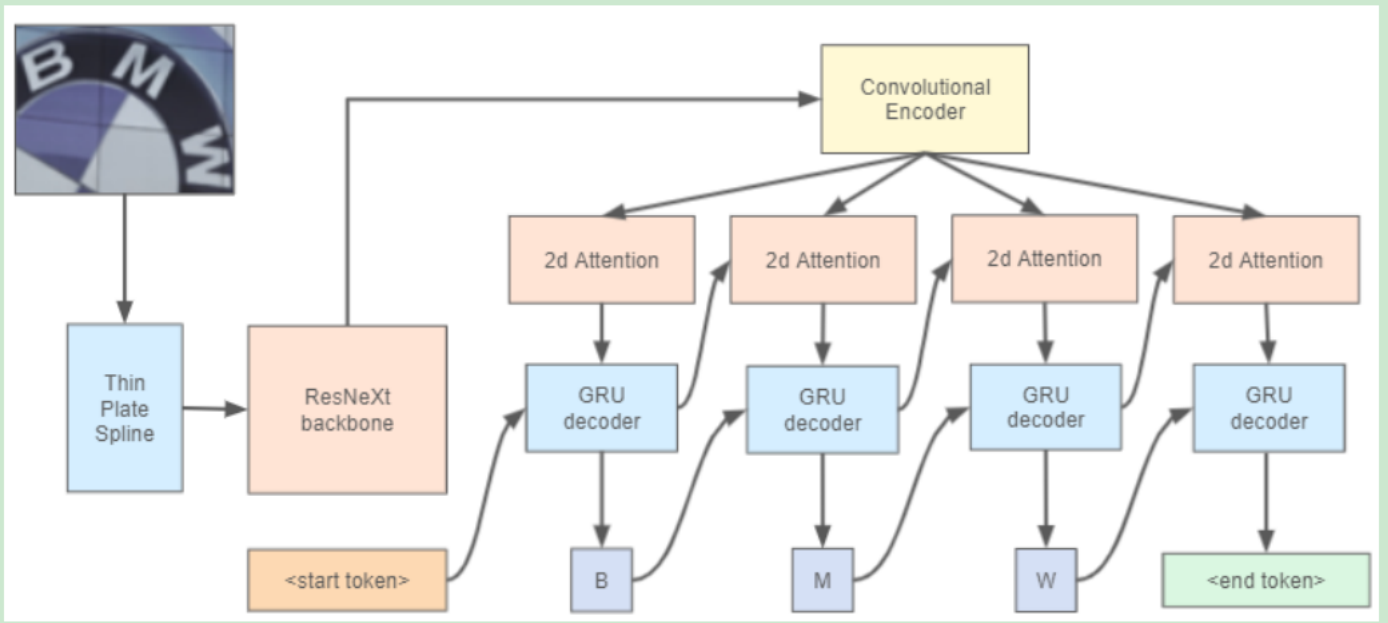


Fig. 1. Proposed text recognition model architecture.

图 21: 网络结构

参考文献

[1] C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "Cspnet: A new backbone that can enhance learning capability of cnn," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020, pp. 390–391.

[2] M. Liao, Z. Wan, C. Yao, K. Chen, and X. Bai, "Real-time scene text detection with differentiable binarization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 11 474–11 481.

[3] B. Shi, M. Yang, X. Wang, P. Lyu, C. Yao, and X. Bai, "Aster: An attentional scene text recognizer with flexible rectification," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 9, pp. 2035–2048, 2018.

[4] I. Krylov, S. Nosov, and V. Sovrasov, "Open images v5 text annotation and yet another mask text spotter," *arXiv preprint arXiv:2106.12326*, 2021.