SSD 在 MATLAB 中的实现

1. 卷积层

1.1. 输入输出

输入：

in_array -----> 输入特征图，维度为 3（高、宽、深或原始图像的通道数）。

kernels -----> 卷积核，维度为 4（高、宽、深、卷积核个数）。

stride -----> 卷积核移动步长。

padding -----> 填充像素数。

dilation -----> 卷积核膨胀距离。

输出：

out_array -----> 输出特征图，维度为 3（高、宽、深）。

1.2. 原理

3 维卷积的基本原理

CNN 中的卷积运算主要是卷积核与图像块在 3 个维度上对应点相乘运算（严格意义上讲应为互相关运算，但在神经网络中卷积和互相关的概念是混淆的）。

卷积过程可表示为：

$$O(i,j,k)=\sum_{r}\sum_{s}\sum_{t}I(i+r,j+s,k+t)K(r,s,t) \tag{1}$$

其中，I 代表卷积层输入，K 代表卷积核。i，j，k 和 r，s，t 分别代表输入和卷积核的三个维度标号。

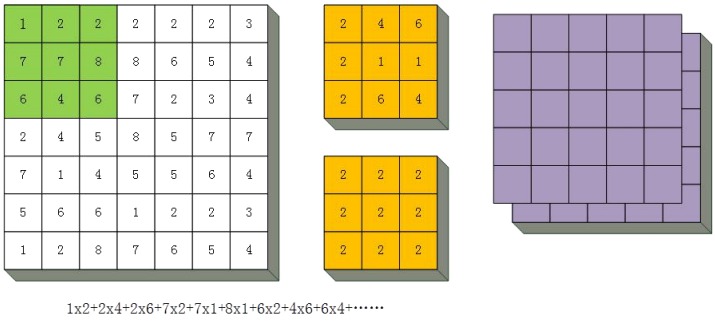

图 1 深度学习中的三维卷积运算

以上图为例，卷积层输入的尺寸为 7x7xc。该层卷积核尺寸为 3 x 3 x c x kernel_num，可以理解为有 kernel_num 个 3 维的卷积核对输入进行卷积操作。每个卷积核在输入的第一、二维组成的平面上按步长(stride)滑动，对卷积核覆盖位置进行对应点相乘、累加操作，得到 5 x 5 x 1 的结果。将每个卷积核的卷积结果按第三维进行拼接，得到 5 x 5 x kernel_num 的输出。

将上图范例推广到一般情况，若输入为 w x h x c，卷积核尺寸为 k_w x k_w x c x kernel_num 其输出为 $\frac{w-k\_w+1}{stride}$ x $\frac{h-k\_h+1}{stride}$ x kernel_num。乘运算和加运算均为 $\frac{w-k\_w+1}{stride}$ x $\frac{h-k\_h+1}{stride}$ x kernel_num x k_w x k_h x c 次。

padding

卷积后会出现输出的尺寸略小于输入尺寸的情况，而人们为了后续处理方便要将卷积层的输出尺寸和输入保持一致。因此在输入进入卷积层之前，需要对输入进行填充的操作（一般填充数字 0）。padding 表示每条边向外填充 0 的数量。

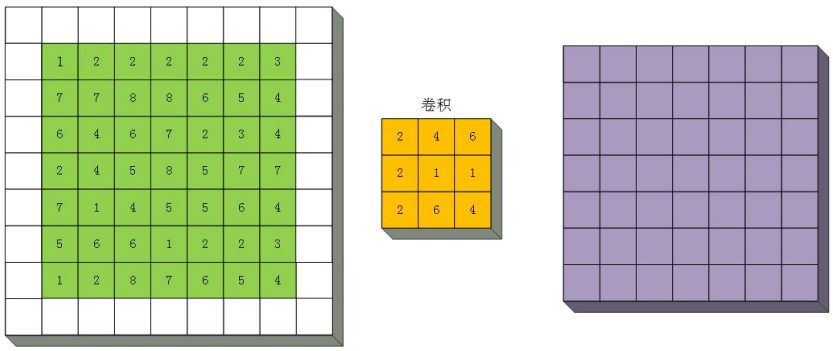

图 2 padding

以上图为例，输入为 7 x 7 时在四周填充 1 个像素的 0，可以保证输入输出大小一致。一般 padding 的值取 floor(k_w/2)、floor(k_h/2)，来保证输入输出尺寸一致。

dilation

dilation 用于控制卷积核中孔的大小，形成多孔卷积核。多孔卷积增大了每个卷积核的感受野，起到与池化层类似的作用。

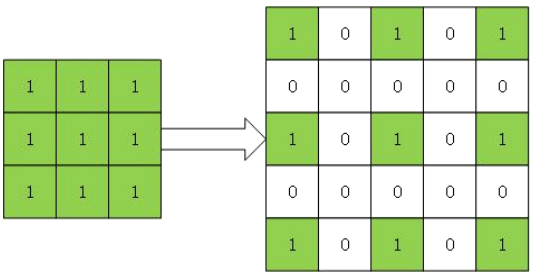

图 3 多孔卷积核

上图为 dilation 为 1 时的卷积核，与普通卷积核相比，卷积核系数间距离增大为 1，用 0 填充。

在 Matlab 实现时，采用提取输入相应位置数值的方法完成 dilation 的实现。以上图为例，实现时有两种方法：(1)扩大卷积核为 5 x 5，取输入相应位置的 5 x 5 邻域计算；(2)不扩大卷积核尺寸，取输入相应邻域中指定位置（绿色区域）进行计算。第一种方法取操作数容易，但增添了多余的计算量。程序中采用第二种方式。

1.3. im2col

为提高运算速度，Caffe 框架中使用了 im2col 的方法加快卷积计算。

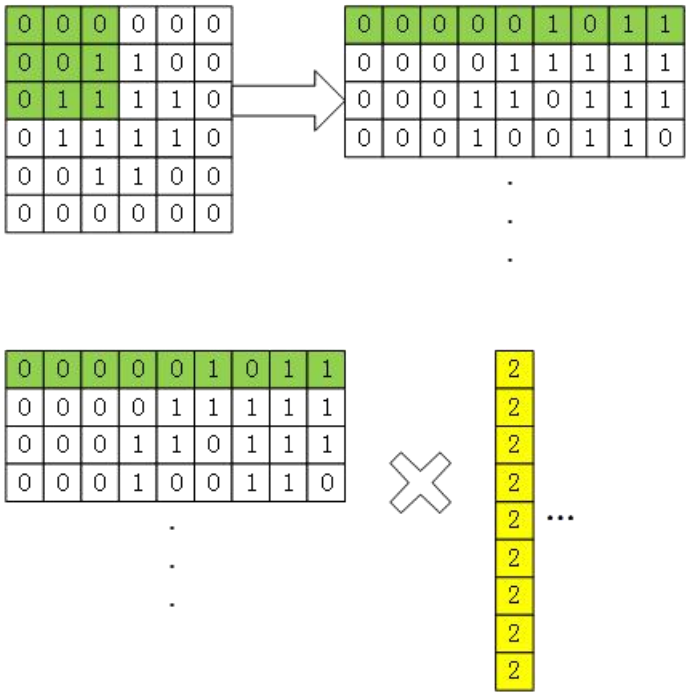

图 3 im2col

如上图所示，im2col 首先将每个滑动窗内的数据重排为行向量，列序按滑动窗位置组织为矩阵。卷积核也按照类似的方法处理，每个卷积核重排为列向量，

行序按卷积核的先后顺序组织为矩阵。两矩阵相乘结果的每一行重排后可得特征图。

2. 激活函数层(RELU)

2.1. 输入输出

输入：

in_array -----> 输入特征图，维度为 3（长、宽、深）。

输出：

out_array -----> 输出特征图，维度为 3（长、宽、深）。

2.2. 原理

RELU 可表示为：

$$f(x) = \max(x,0) \tag{2}$$

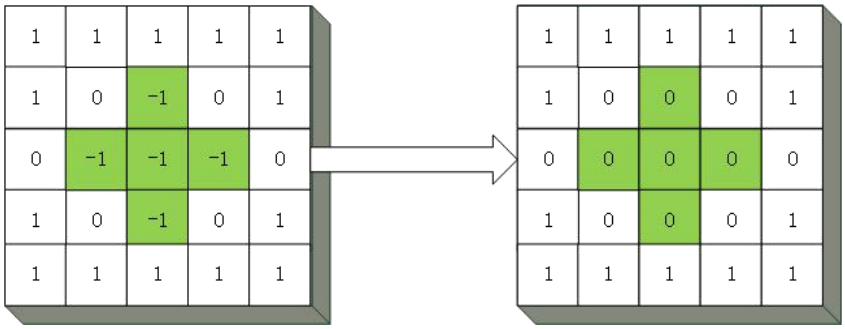

图 4 RELU

3. 池化

3.1. 输入输出

输　入：

in_array　　-----> 输入特征图（dim = 3）。

window_size　-----> 池化窗口大小。

stride　　-----> 步长。

padding　　-----> 填充像素数。

输　出：

out_array　　-----> 输出特征图（dim = 3）。

3.2. 原理

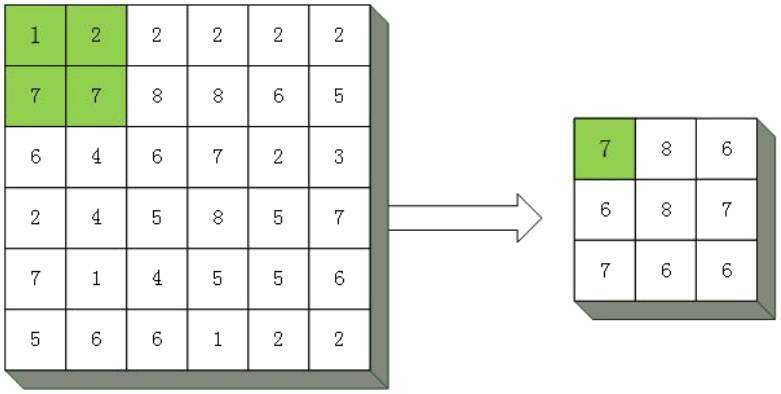

图 5 最大池化

如上图所示，池化层将每个 3 维的输入特征图按第三维拆分为若干个平面。每个平面上取窗口，滑动。上图的池化方式为最大池化，取窗口内最大值作为输出。窗口滑动的步长为 2。

4. 归一化层

归一化层在第三个维度上对数据进行归一化。

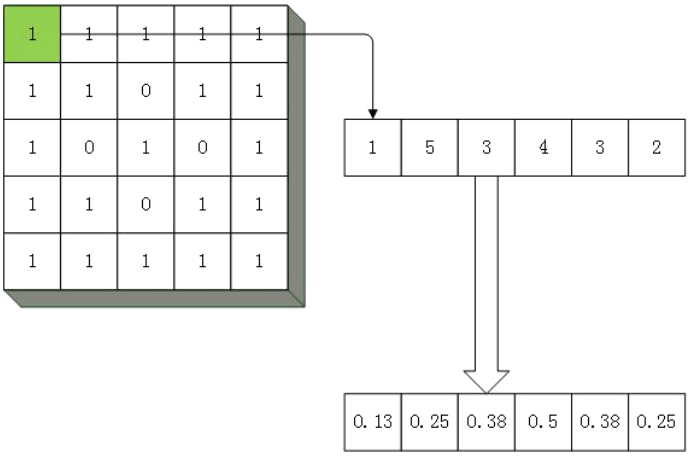

图 6 Norm3d

以上图为例，对特征图每个位置进行归一化。首先，取对应位置的向量，计算 2 范数，采用以下公式进行归一化：

$$x' = \frac{x}{\|x\|_2} \tag{3}$$

5. Softmax 层

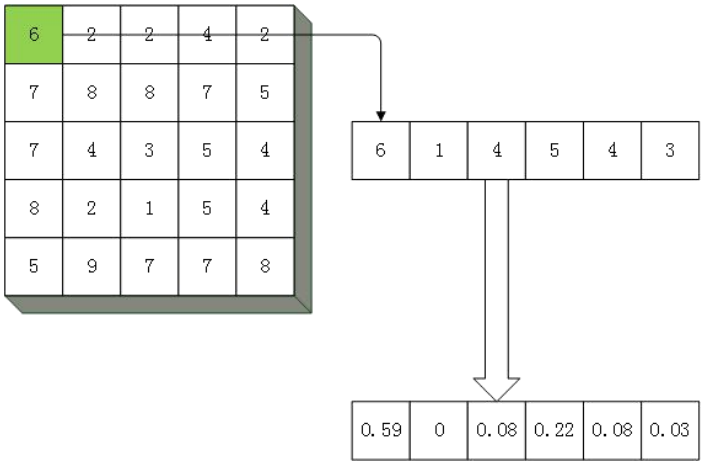

图 7 Softmax

SSD 对每个 Bounding Box 属于哪一类进行打分，但打分的范围是 $(-\infty,+\infty)$。因此，需要将分数归一化为概率。

Softmax 根据以下公式来进行归一化：

$$x_i' = \frac{e^{x_i}}{\sum_k e^{x_k}} \tag{4}$$

6. Prior Box 生成

6.1. 输入输出

输入：

scale -----> 特征图对应尺度。

aspect_ratio -----> 特征图 Box 对应长宽比。

feature_size -----> 特征图大小。

输出：

priorbox -----> 输出 Prior Box。

6.2. 原理

Prior Box 是物体回归所用的候选框。SSD 网络通过找到与物体最小 jaccard 距离的 Prior Box 进行调整，如下图所示：

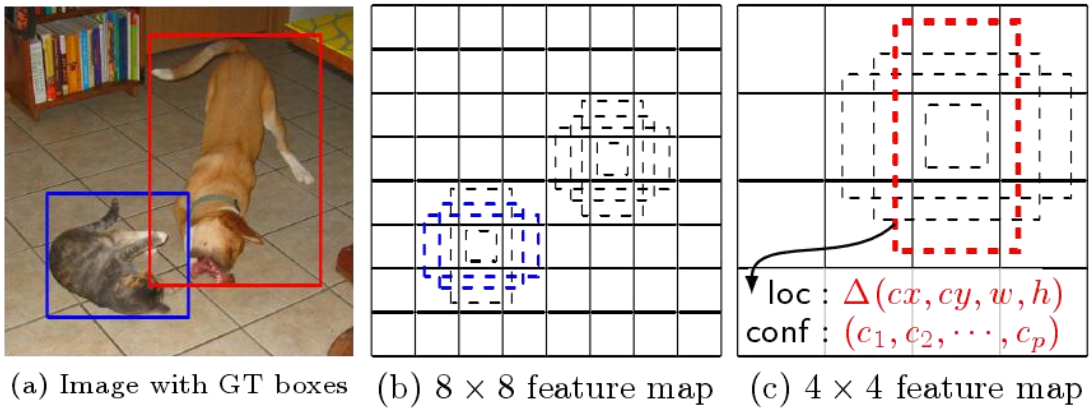

图 6 Prior Box

以 conv7_2 层特征图为例，特征图尺寸为 5 x 5，对应将原图划分为 5 x 5 个位置。如下图所示：

图 7 conv7_2 网格划分

每个位置对应长宽比不同的 6 个 Prior Box。其中心为 $\left((i-0.5)\times\frac{1}{5},(j-0.5)\times\frac{1}{5}\right)$（归一化后）。

该特征图下，长宽比为 1:1 的 Prior Box 长和宽均为 scale=0.54。2:1 的 Prior Box 长为 $scale\times\sqrt{2}$，宽 $scale/\sqrt{2}$。1:2 的 Prior Box 长为 $scale/\sqrt{2}$，宽 $scale\times\sqrt{2}$。1:3 和 3:1 的 Prior Box 类似。在这 5 种长宽比的基础上，增加一个略大与该尺度的 Prior Box，长宽均为 $\sqrt{s_k s_{k+1}}$，其中 s_k 代表特征图尺度，s_{k+1} 代表下一层特征图对应的尺度。

由中心位置和长宽可计算左上角、右下角点位置。计算得出的结果按(lx,ly,rx,ry)在第三维进行排列。

7. Prior Box 调整

由 Prior Box 的信息和回归框精修的结果可得最终目标的 Bounding Box。

设 Prior Box 中心为(p_cx,p_cy)，长宽为(p_w,p_h)。网络输出的回归框调整值为(l_x,l_y,l_w,l_h)。

最后按以下公式得到的 Bounding Box 中心点和长、宽。

$$cx = p\_cx + l\_x \times p\_w$$

$$cy = p\_cy + l\_y \times p\_w$$

$$w = p\_w \times e^{l\_w}$$

$$h = p\_h \times e^{l\_h}$$