# 深度学习快速入门

08 卷积层

POET

2024 年 2 月 13 日

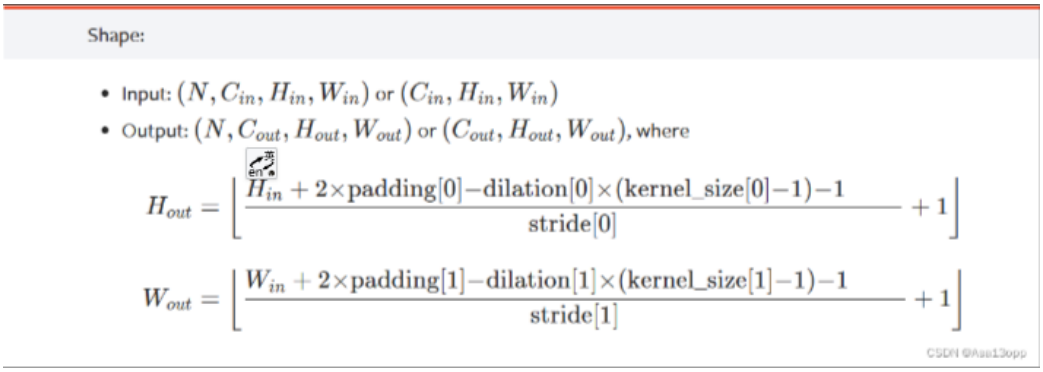
# 1 卷积原理

- Conv1d 代表一维卷积，Conv2d 代表二维卷积，Conv3d 代表三维卷积。
- kernel_size 在训练过程中不断调整，定义为 3 就是 3 * 3 的卷积核，实际我们在训练神经网络过程中其实就是对 kernel_size 不断调整。

```
torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0,

dilation=1, groups=1, bias=True, padding_mode='zeros', device=None, dtype=None)

常用参数:

    in_channels (int) - 输入图像中的通道数

    out_channels (int) - 由卷积产生的通道数

    kernel_size (int or tuple) - 卷积核的大小

    stride (int or tuple, optional) - 卷积的步幅 Default: 1

    padding (int, tuple or str, optional) - 添加到输入四个边的Padding Default: 0

    padding_mode (str, optional) - , , or . Default: 'zeros''reflect''replicate''circular''zeros'

    dilation (int or tuple, optional) - Spacing between kernel elements. Default: 1
```

```
20
21      groups (int, optional) - Number of blocked
            connections from input channels to output
            channels. Default: 1
22
23      bias (bool, optional) - If , adds a learnable
            bias to the output. Default: TrueTrue
```

可以根据输入的参数获得输出的情况，如下图所示：



Shape:

- Input: $(N, C_{in}, H_{in}, W_{in})$ or $(C_{in}, H_{in}, W_{in})$
- Output: $(N, C_{out}, H_{out}, W_{out})$ or $(C_{out}, H_{out}, W_{out})$, where

$$H_{out} = \left\lfloor \frac{H_{in} + 2 \times \text{padding}[0] - \text{dilation}[0] \times (\text{kernel\_size}[0] - 1) - 1}{\text{stride}[0]} + 1 \right\rfloor$$

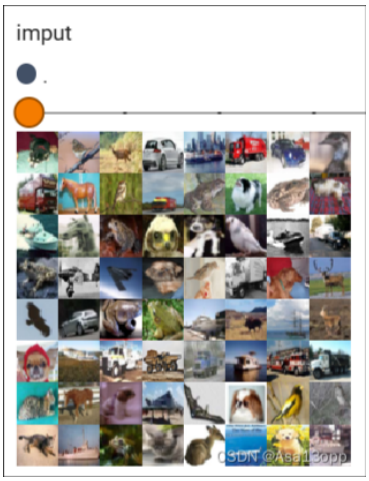
$$W_{out} = \left\lfloor \frac{W_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor$$

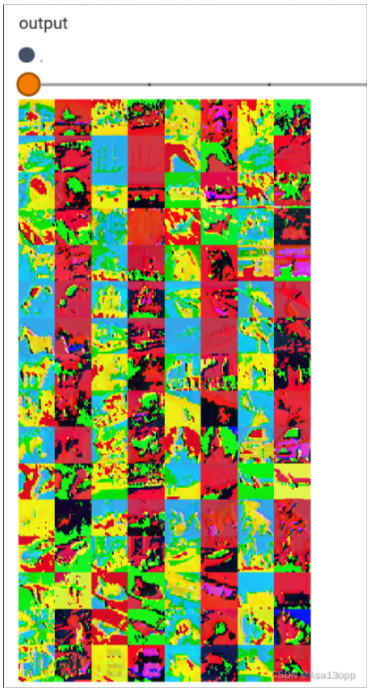


## 2 搭建卷积层并处理图片

```
1   import torch
2   from torch import nn
3   import torchvision
4   from torch.nn import Conv2d
5   from torch.utils.data import DataLoader
6
7   #搭建卷积层
8   dataset = torchvision.datasets.CIFAR10("./dataset
        ",train=False,transform=torchvision.transforms.
        ToTensor(),download=True)
9   dataloader = DataLoader(dataset, batch_size=64)
```

```
10      class Tudui(nn.Module):
11          def __init__(self):
12              super(Tudui, self).__init__()
13              self.conv1 = Conv2d(in_channels=3,
                    out_channels=6,kernel_size=3,stride=1,
                    padding=0) # 彩色图像输入为3层，我们想
                    让它的输出为6层，选3 * 3 的卷积
14
15          def forward(self,x):
16              x = self.conv1(x)
17              return x
18
19      #卷积层处理图片
20      tudui = Tudui()
21      writer = SummaryWriter("logs")
22      step = 0
23      for data in dataloader:
24          imgs, targets = data
25          output = tudui(imgs)
26          print(imgs.shape)    # 输入为3通道32×32的64张图
                片
27          print(output.shape)    # 输出为6通道30×30的64张
                图片
28          writer.add_images("input", imgs, step)
29          output = torch.reshape(output,(-1,3,30,30))
              #-1为占位符，让计算机自行计算后填入
30      # 把原来6个通道拉为3个通道，为了保证所有维度总数不
            变，其余的分量分到第一个维度中
31          writer.add_images("output", output, step)
32          step = step + 1
```


imput


output