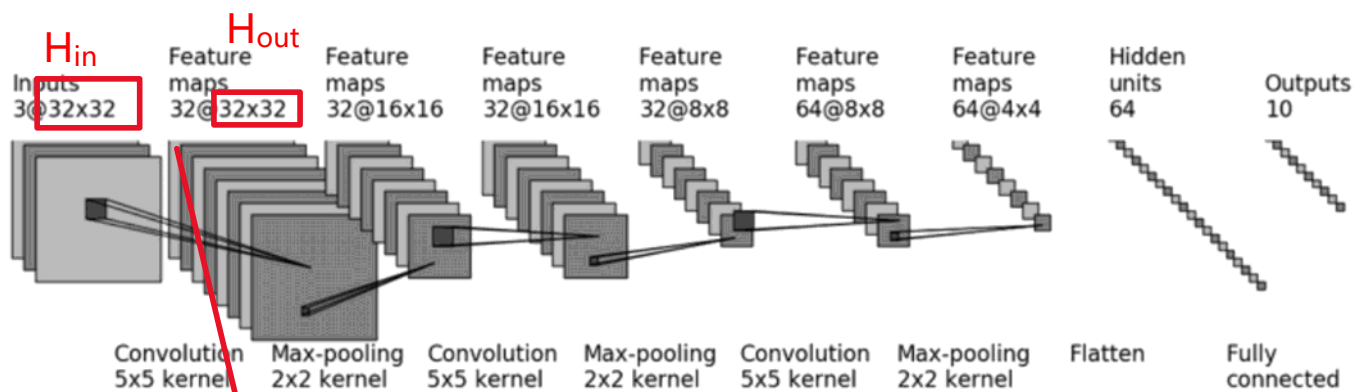


# 小实战—搭建cifar10神经网络

2025年11月23日 13:48



代入公式去算,  $H_{in}$ ,  $H_{out}$ , dilation默认1, kernel\_size5, stride默认1

padding是2

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

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

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

The screenshot shows a PyCharm IDE with a project named 'nurse\_net'. The file 'CIFAR10\_construction.py' is open, showing the construction of a neural network module named 'Solmount'. The code defines several layers: 'conv1', 'maxpool', 'conv2', 'maxpool2', 'conv3', 'maxpool3', 'flatten', 'linear', and 'linear2'. The 'forward' method is also defined. The 'Solmount' module is instantiated and printed to the console.

```
def __init__(self):
    self.conv1 = nn.Conv2d(3, 32, kernel_size=5, stride=1, padding=2)
    self.maxpool = nn.MaxPool2d(2)
    self.conv2 = nn.Conv2d(32, 32, kernel_size=5, stride=1, padding=2)
    self.maxpool2 = nn.MaxPool2d(2)
    self.conv3 = nn.Conv2d(32, 64, kernel_size=5, stride=1, padding=2)
    self.maxpool3 = nn.MaxPool2d(2)
    self.flatten = Flatten()
    self.linear = Linear(in_features=1024, out_features=64)
    self.linear2 = Linear(in_features=64, out_features=10)

    def forward(self, x):
        x = self.conv1(x)
        x = self.maxpool(x)
        x = self.conv2(x)
        x = self.maxpool2(x)
        x = self.conv3(x)
        x = self.maxpool3(x)
        x = self.flatten(x)
        x = self.linear(x)
        x = self.linear2(x)

Solmount = solmount()
print(Solmount)
```

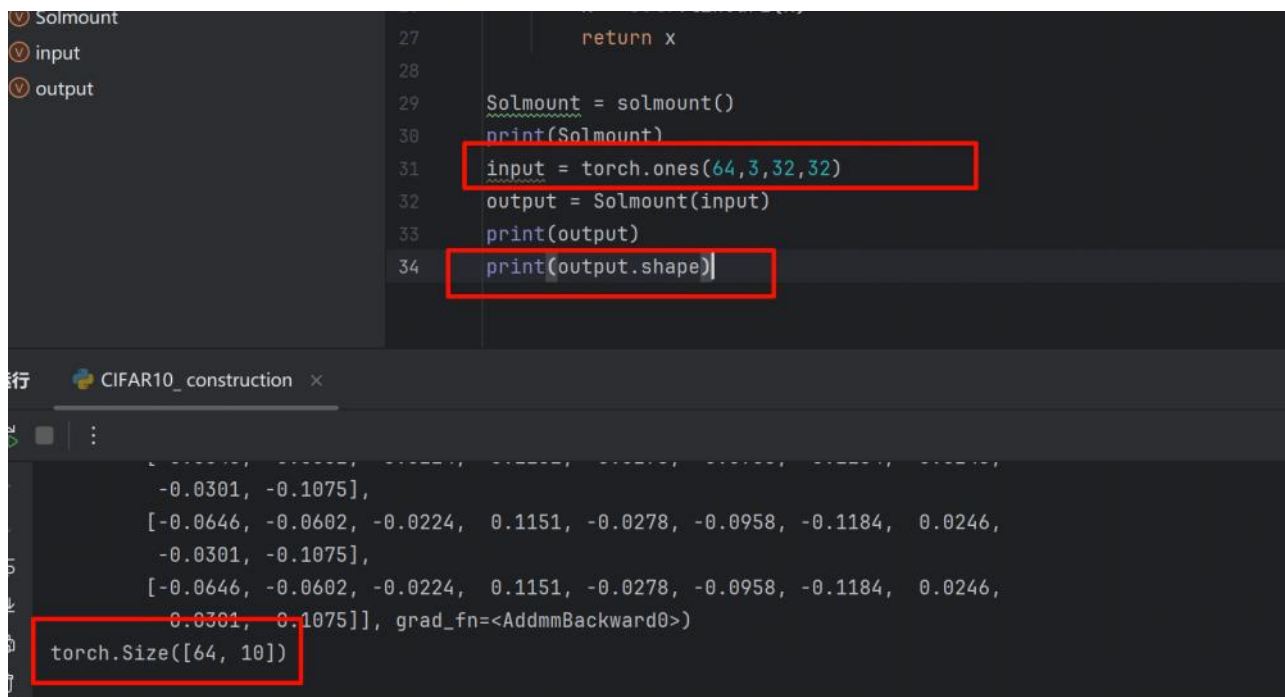
The output of the print statement is shown in the console:

```
solmount(
  (conv1): Conv2d(3, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (maxpool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(32, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (maxpool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv3): Conv2d(32, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (maxpool3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (flatten): Flatten(start_dim=1, end_dim=-1)
```

Print 以后，可以看这个网络中含有哪些层

## 验证

使用ones，创建一个想要的格式，输入进网络，看他输出来什么东西



```
27         return x
28
29     Solmount = solmount()
30     print(Solmount)
31     input = torch.ones(64,3,32,32)
32     output = Solmount(input)
33     print(output)
34     print(output.shape)
```

CIFAR10\_construction

```
[-0.0301, -0.1075],
[-0.0646, -0.0602, -0.0224,  0.1151, -0.0278, -0.0958, -0.1184,  0.0246,
-0.0301, -0.1075],
[-0.0646, -0.0602, -0.0224,  0.1151, -0.0278, -0.0958, -0.1184,  0.0246,
-0.0301, -0.1075]], grad_fn=<AddmmBackward0>)
torch.Size([64, 10])
```

这只是搭建了cifar10的基本框架，离真正训练，还有一段的距离