1、dir（功能包）🡪 输出功能包模块

help（功能包. 模块）->如何使用

若输出的带有前后双下划线，说明是函数了，此时应该用help查看使用说明，注意help查看函数帮助不需要加函数的括号

在pycharm里也可以ctrl+鼠标左键

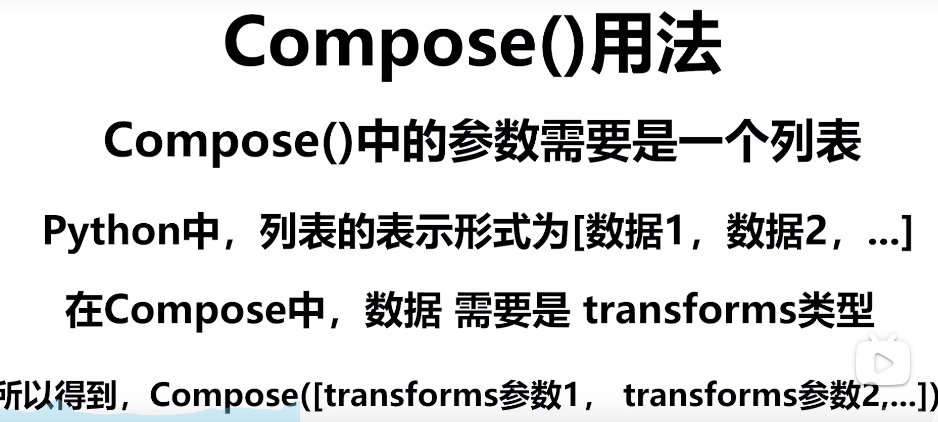
2、Dataset类、Dataloader类：数据集读取

3、Tensorboard类、Transforms类：训练过程



归一化：

*output[channel] = (input[channel] - mean[channel]) / std[channel]*

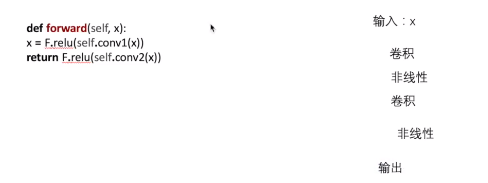


作用：将多个transforms步骤整合到一起

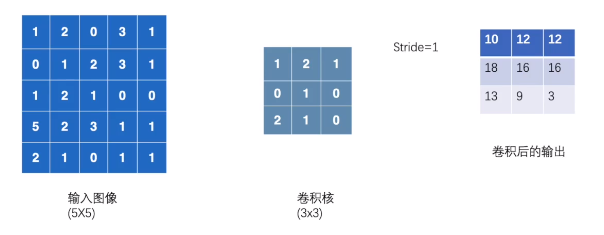
4、torchvision数据集

官网-》DOCS下

5、神经网络



1、卷积层：



对应位置相乘相加，在移动卷积核

**channels 该如何理解？**

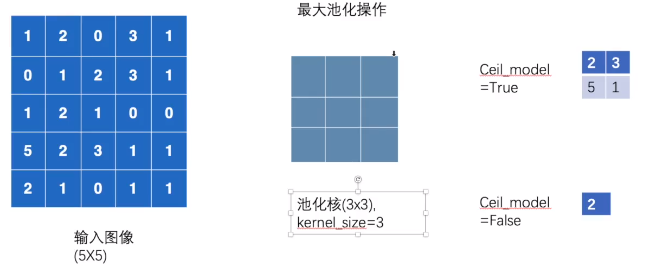
**一般的RGB图片，channels 数量是 3 （红、绿、蓝）；而monochrome图片，channels 数量是 1**

<https://blog.csdn.net/sscc_learning/article/details/79814146>

2、池化层（最大池化）

池化核默认的步长为池化核的大小

ceil\_mode：不足池化核的数目时，是否保留数据（true保留）

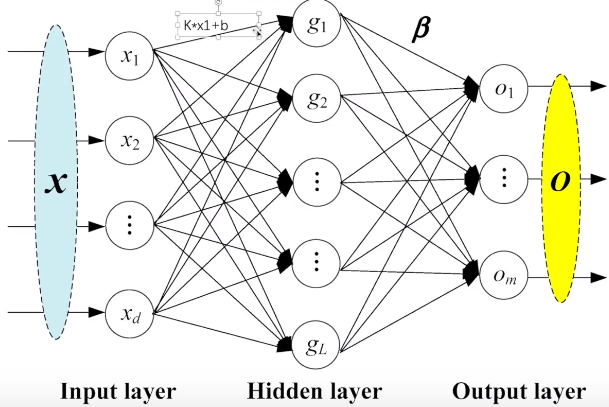


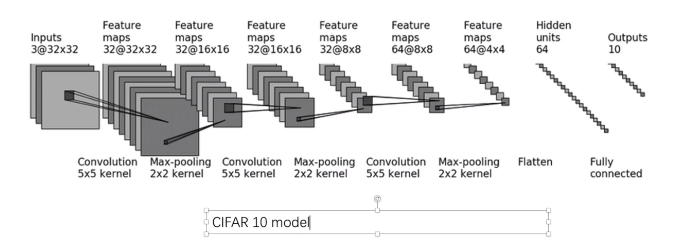
作用：提取目标关键数据，减少计算量

3、非线性化

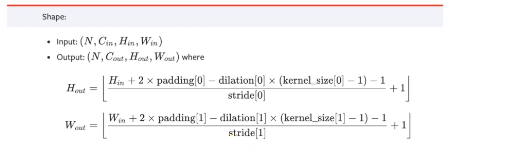
作用：引入非线性特征，训练出更符合的神经网络

4、线性层





第一层卷积：32x32---》32x32需要计算stride和padding（以下公式计算）



Flatten：之后会有64x4x4个数据，在经过线性层，到达Hidden units

5、损失函数：（越小越好）Loss Function

作用：



*反向传播，产生梯度信息可以进行优化*

6、优化器（基于5）

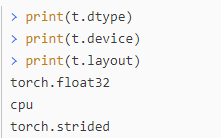
lr：学习速率

【常识】

1、PyTorch中的张量就是torch.Tensor的Python类的一个实例。

每一个torch.Tensor有这些属性:

t = torch.Tensor()

（数据类型，设备，张量在内存中的存储方式）

**1. 直接生成张量**

data = [[1, 2], [3, 4]]

x\_data = torch.tensor(data)

**2. 通过Numpy数组来生成张量**

np\_array = np.array(data)

x\_np = torch.from\_numpy(np\_array)

**3. 通过已有的张量来生成新的张量**

x\_ones = torch.ones\_like(x\_data) #*创建x\_data大小的1张量*

x\_rand = torch.rand\_like(x\_data, dtype=torch.float) #*创建x\_data大小的随机张量*

**4. 通过指定数据维度来生成张量**

shape = (2,3,) #元组类型, 描述张量的维数

rand\_tensor = torch.rand(shape)

ones\_tensor = torch.ones(shape)

zeros\_tensor = torch.zeros(shape)

**1. 张量的索引和切片**

ones\_tensor[:,1] = 1 *#第0列所有值1*

**2. 张量的拼接**

t1 = torch.cat([zeros\_tensor, zeros\_tensor, zeros\_tensor], dim=1) *# 连接 dim（1/0）：按行/列 拼接*

**3. 张量的对应元素相乘- -两种写法**

zeros\_tensor = zeros\_tensor.mul(zeros\_tensor)

zeros\_tensor1 = zeros\_tensor \* zeros\_tensor *#对应元素相乘*

**4. 张量的矩阵乘法（要满足可乘性）- - 两种写法**

zeros\_tensor = zeros\_tensor.matmul(zeros\_tensor.T)

zeros\_tensor2 = zeros\_tensor @ zeros\_tensor.T *#矩阵乘法*

**Tensor与Numpy的转化**

1、torch转np

n = t.numpy()

2、np转torch

t = torch.from\_numpy(n)