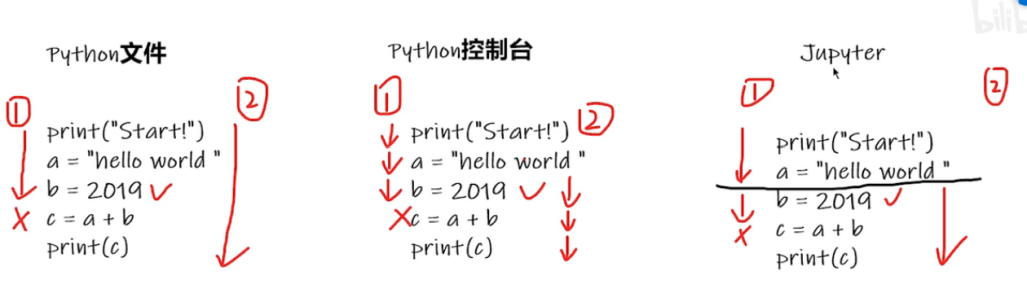
pytorch----<package 有很多分隔区

dir()打开，看见

help()说明书

1. **运行顺序**

如果代码是一块为一个整体运行的话，py把所有代码视为一个整体（有错就从头开始）；而py控制台是以每一行为一块进行执行的；jupyter可以以任意行为块来执行。



1. **pytorch如何读取数据**

Dataset Dataloader

获取数据以 为网络提供不同的数据形式

及label

访问数据集中的元素：img,label=bees\_dataset[1]

img.show()

windows路径要加//

1. **Tensorboard的使用**

add.scalar-->

scalar\_value y轴 global\_step x轴 tag\

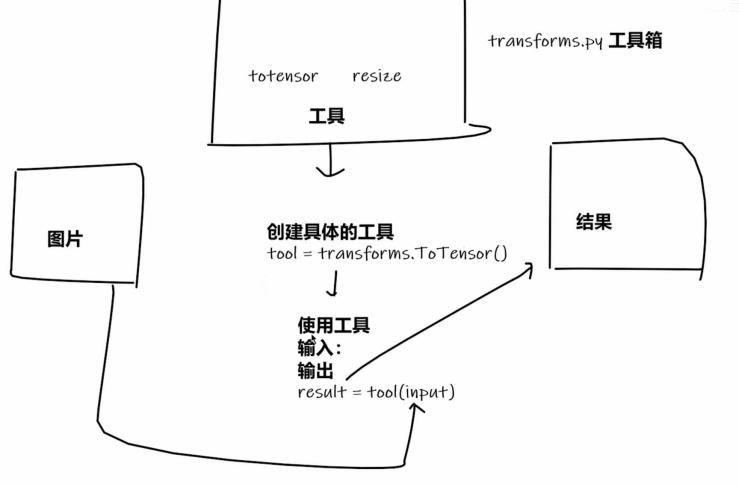
(pytorch) PS D:\作业\project1> tensorboard --logdir=Logs来打开log文件，logdir指的是当前文件夹下的文件名

add-image()的使用：tag，image-tensor,global-step

将图片格式由PIL转化为numpy-->opencv,或者numpy数组：img\_array=np.array(img)

1. **#transforms的用法**  #我们为什么需要tensor数据类型

transforms.py工具箱 :totensor,resize(工具）

1. **常见的transforms**

\*输入 -->PIL -->Image.open()

ToTenSor的使用、Normallize:输入tensor还有均值和标准差，信道中的输入归一化

Resize RandomCrop

1. **torchvision中数据集的使用**
2. **DataLoader的使用 怎么读取数据集**

test\_data = torchvision.datasets.CIFAR10(

root='./data', # 数据集下载和保存的根目录

train=False, # 是否为训练集（False表示测试集）

download=True, # 如果数据集不存在，是否自动下载

transform=torchvision.transforms.ToTensor() # 数据预处理转换

)

writer=SummaryWriter("dataloader")  
step=0  
for data in test\_loader:  
 img,targets=data  
 print(img.shape)  
 print(targets)  
 writer.ass\_images("teat\_data",img,step)  
 step=step+1  
writer.close()

#----->epoch

for epoch in range(2):#相当于进行了两轮  
 step=0  
 for data in test\_loader:  
 img,targets=data  
 print(img.shape)  
 print(targets)  
 writer.ass\_images("Epoch:teat\_data",img,step)  
 step=step+1

1. **怎么搭建神经网络**

Module

输入-->卷积-->非线性处理-->卷积-->非线性-->输出

convolution卷积-->COVN2D import torch.nn.functional as F

stride步长，卷积核每次移动的步长，可以单个数也可以是（ ，）控制左右和上下移动

padding填充

1. **神经网络的卷积层**

torch.nn.conv2d(in\_channels,out\_channels,kernel\_size（卷积核的大小）,tride=1,padding=0)

int int int or 元组 后面都是

1. **最大池化的使用**

torch.nn.MaxPool2d(kernel\_size,stride=None,padding=0.dilation=1,return\_indicices=False,ceil\_mode=False-->不完整就舍去)

floor、ceilling上下取整

1. **神经网络的非线性激活**

nn.Relu ---->inplace默认是False,不会用函数返回的结果对原值直接进行替换

1. **线性层及其他层介绍**

nn.Sequential 是一个容器类，用于将多个网络层按顺序组合在一起。

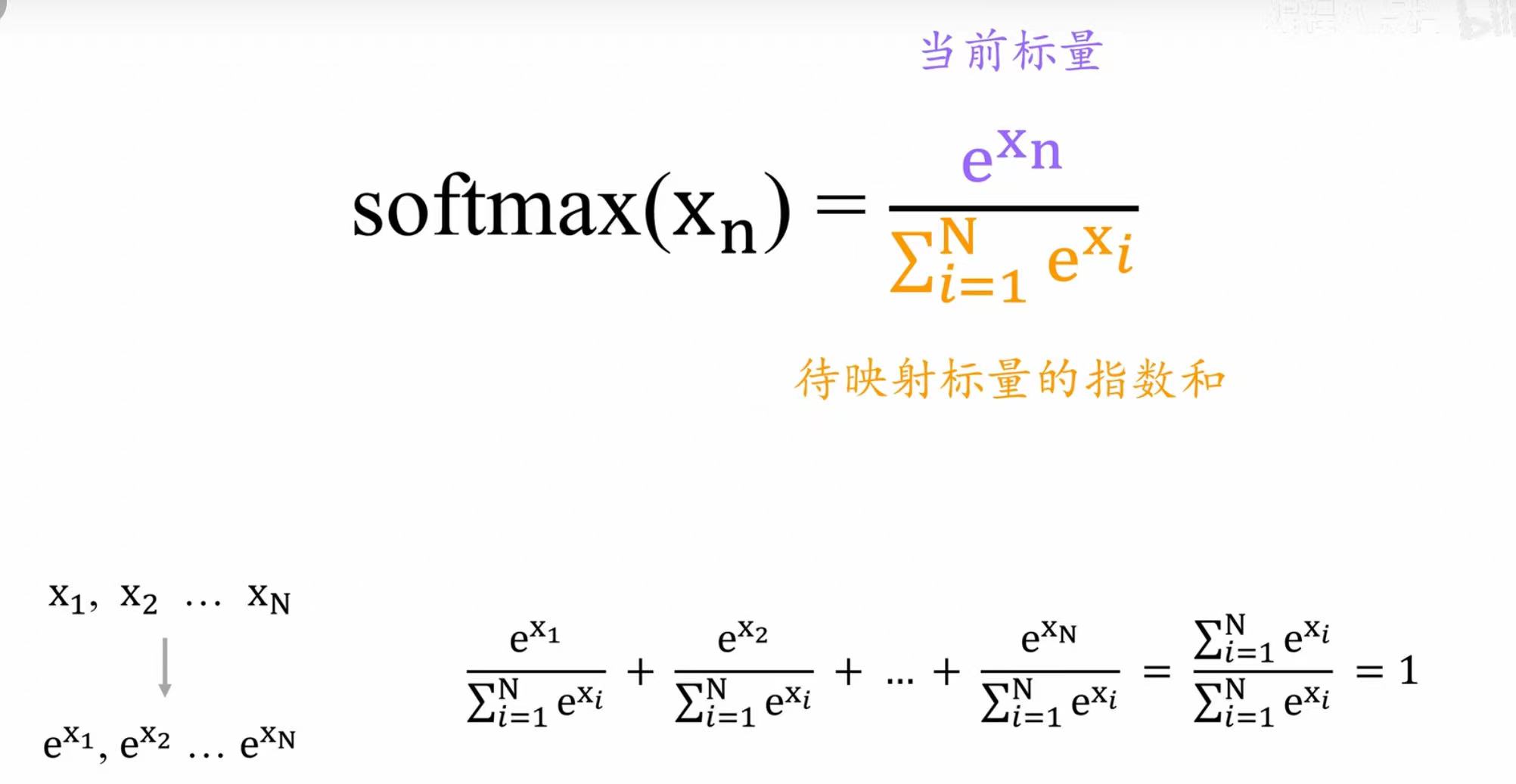
self.model1=Sequential(  
 Conv2d(3,32,5,padding=2),  
 MaxPool2d(2),  
 Conv2d(32, 32, 5, padding=2),  
 MaxPool2d(2)）

-->x=self.model1(x)来使用

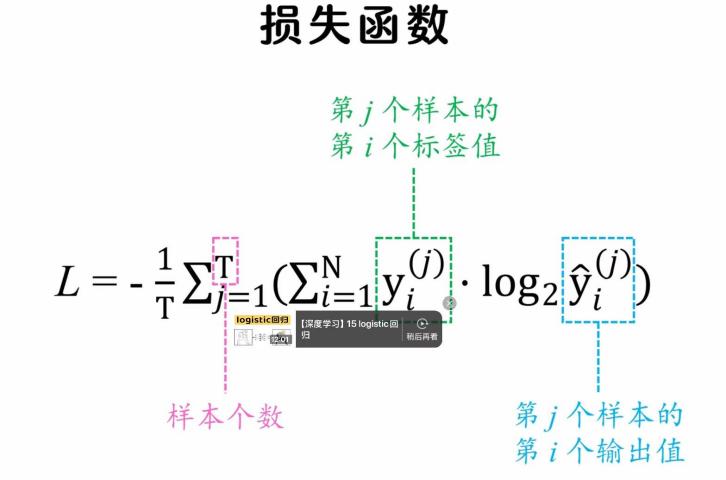
1. **损失函数与反向传播**

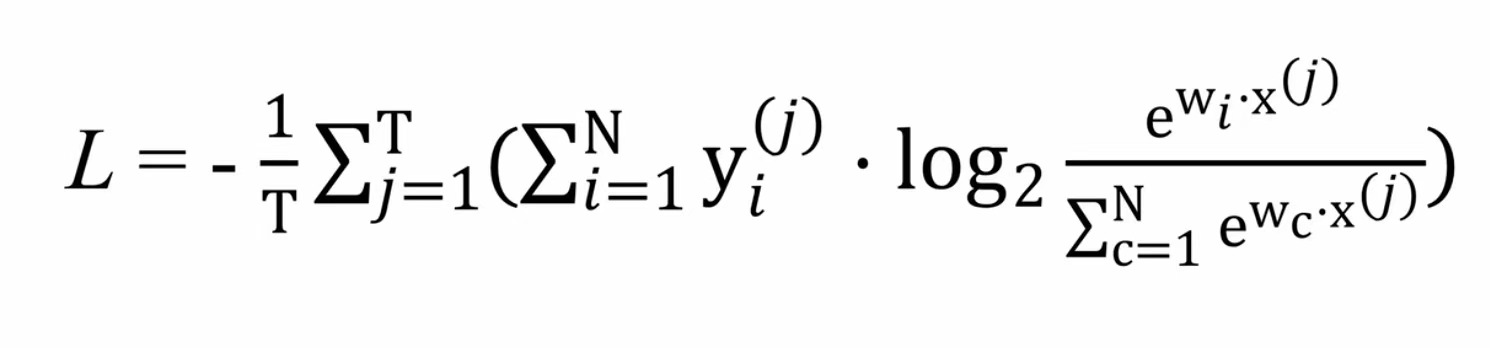
L1Loss 平均绝对损失；MSE（L2Loss）均方差。

softmax分类（多分类）：每个元素非负而且和为一（概率），映射后概率最高的即为分类结果。

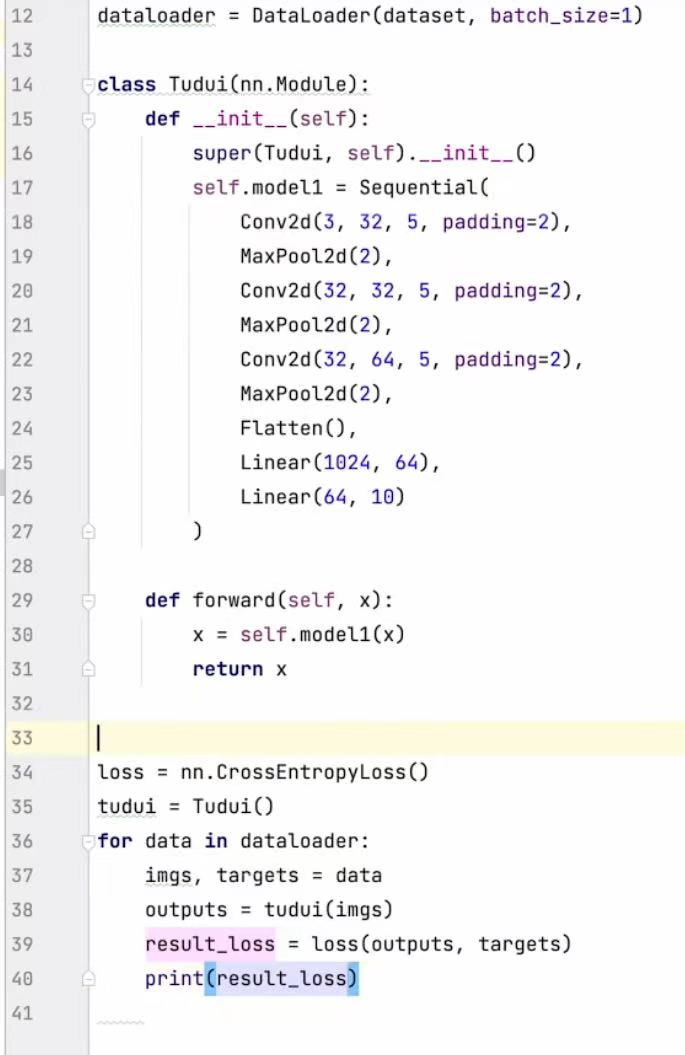


————>+交叉熵



即为总的损失函数

应用如下：



1. **优化器的选择**

lr（learningrate学习速率）

1. **模型的加载和保存**
2. **完整的模型训练套路**

准备数据集-->准备dataloader-->建立模型-->设置损失函数-->设置优化器-->设置训练中的参数-->循环多次训练（算出误差放入优化器里优化，然后输出）-->测试，计算出误差，以指标反馈-->保存模型