深度学习网络训练中，需要转移到GPU上的数据是：

(1)模型参数

(2)数据样本和其标签

(3)损失函数(若损失函数中含参数)

**单显卡训练**

**.cuda()**

net = Net()

net = net.cuda()*#默认应用0号显卡*

criterion = torch.nn.CrossEntropyLoss()

criterion = criterion.cuda()

for iteration, batch in enumerate(train\_loader):

image, label = batch

image = image.cuda()

label = label.cuda()

output = net(input)

loss = criterion(output,label)

**.to(device)**

device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")*#默认应用0号显卡*

data = data.to(device)*#该语句执行的次数等于最开始读取数据时的tensor变量个数，由这些变量计算所衍生的变量也在显卡上*

net = net.to(device)

单显卡训练网络和CPU训练网络保存方式相同

**多显卡训练**

**torch.nn.DataParallel(module, device\_ids=None, output\_device=None, dim=0)**

**参数解释：**

module：需要并行处理的模型

device\_ids：并行处理的设备，默认使用所有的cuda

output\_device：输出的位置，默认输出到device\_ids[0]

在执行DataParallel时，

前向过程：

1. 分发mini-batch到每个device上，

2. 将模型复制到每个device上，

3. 模型对mini-batch进行forward计算结果，

4. 将每个device上的计算结果汇总到device\_ids[0]上

反向传播过程：

1. 在device\_ids[0]上根据计算的loss计算梯度，

2. 将device\_ids[0]上计算的梯度值分发到每个device上，

3. 每个device上进行梯度更新，

4. 将每个device上更新的梯度值汇总到device\_ids[0]上

os.environ["CUDA\_VISIBLE\_DEVICES"] = '1,2'*#1号显卡是主卡*

input = input.cuda()

net = torch.nn.DataParallel(net, device\_ids =[0,1])*#0对应1号显卡，1对应2号显卡，必须含0，因为1号显卡是主卡；若不设置device\_ids，则默认使用所有Python环境可检测到的显卡*

net = net.cuda()

output = net(input)

**多显卡训练网络保存和加载**

net = torch.nn.Linear(10,1)*#先构造一个网络*

net = torch.nn.DataParallel(net, device\_ids=[0,3])*#包裹起来*

torch.save(net.module.state\_dict(),’Model/net.pth’)*#需要在保存时多调用一个net.module*

*#加载网络*

new\_net = torch.nn.Linear(10,1)

new\_net.load\_state\_dict(torch.load(’Model/net.pth’))