问题1. 数据在网络中的维度顺序是什么？

答：数据在网络中的维度顺序是batch、channel、height、width。

问题2. nn.Conv2d()中参数含义与顺序？

答：nn.Conv2d()中参数：

in\_channels 输入的通道数

out\_channels 输出的通道数

kernel\_size 卷积核大小

stride=1 卷积移动的步长，默认值是1

padding=0 padding的宽度，默认值为0

dilation=1 卷积时的膨胀系数，默认值为1

groups=1 组卷积的组数，默认值为1

bias=True 卷积是否设置偏置值，默认设置

padding\_mode='zeros' padding的方式，默认填充0值。

问题3. nn.Linear()是什么意思？参数含义与顺序？

答：nn.Linear()是线性变换的意思，即神经网络中的全连接层。

nn.Linear()的参数：

in\_features 输入维度

out\_features 输出维度

bias=True 是否设置偏置值，默认是。

问题4. nn.PReLU()与nn.ReLU()的区别？示例中定义了很多nn.PReLU()，能否只定义一个PReLU？

答：nn.ReLU()只截取输入大于0的部分，nn.PReLU()输入大于0的部分保持不变的同时，还将小于0的部分乘以一个权重。

不能只定义一个PReLU，因为各个PReLU的权重需要训练成不同的值。

问题5. nn.AvgPool2d()中参数含义？还有什么常用的pooling方式？

答：nn.AvgPool2d()中的参数：

kernel\_size 池化核的大小

stride=None 池化核移动的步长，默认等于池化核的大小

padding=0 padding的宽度，默认值为0

ceil\_mode=False 输出的特征图大小是否向上取整，默认为否

count\_include\_pad=True 池化计算的时候是否将padding的值计算在内，默认是

divisor\_override=None 如果指定，它将用作除数，否则将使用池化区域的大小。

问题6. view()的作用？

答：view()的作用是将输入的torch.Tensor改变形状(size)并返回。返回的Tensor与输入的Tensor必须有相同的元素,相同的元素数目，但形状可以不一样，即view起到的作用是reshape，view的参数的是改变后的shape。

问题7.如何设置GPU？

答：设置GPU并检查使用情况可以通过以下4个步骤进行：

（1）模型设置

cuda\_gpu = torch.cuda.is\_available()

if(cuda\_gpu):

net = torch.nn.DataParallel(net).cuda()

（2）输入数据设置

.cuda()将tensor专程cuda

inputs, labels = Variable(inputs.cuda()), Variable(labels.cuda())

（3）输出设置

.cpu将cuda转成tensor

outputs = net(Variable(images.cuda()))

\_, predicted = torch.max(outputs.data, 1)

countall = countall + len(labels.cuda())

for j in range(len(labels)):

if(classes[labels.cuda()[j]] == classes[predicted.cuda()[j]]):

countok = countok + 1

print("countall:",countall,", countok:",countok,",Acc:",countok/countall)

（4）最后，查看nvidia-smi中，gpu被调用。

问题8.配合后续周学习，loss都有哪些。分别有什么作用(常用的即可)？

答：常用的loss有以下这些：

L1Loss：计算预测值与真实值之差的绝对值，主要用于回归问题。

MSELoss：计算预测值与真实值之差的平方，主要用于回归问题。

CrossEntropyLoss：计算预测值与真实值的LogSoftmax再求负对数似然损失，主要用于分类问题。

NLLLoss：计算预测值与真实值的负对数似然损失，主要用于分类问题。

BCELoss：计算二分类交叉熵损失，主要用于二分类问题。

BCEWithLogitsLoss：计算二分类交叉熵损失时加入sigmoid，主要用于二分类问题。

SmoothL1Loss：在预测值与真实值相差小于1时计算0.5倍的MSELoss，预测值与真实值相差大于1时计算L1Loss，主要用于回归问题。

问题9.如何设置优化器？

答：使用torch.optim工具包下的优化器类实例化为优化器对象，在训练过程中传入优化器对象作为变量。

问题10.配合第8周内容，常用的优化器有哪些？

答：常用的优化器有Adadelta、Adagrad、Adam、Adamax、Nadam、RMSprop、SGD、NAG等。

问题11. print的格式化如何实现的？

答：print的格式化有两种方式：

1. 使用%作为格式转换符开始的标志，在print传入%需要转换的参数。

示例：print('%s' % 'hello world')

1. 使用format函数来自定义格式，用{}作为格式转换占位符。

示例：print('{} {}'.format('hello','world'))

问题12. optimizer.zero()与optimizer.step()的作用是什么？

答：optimizer.zero()将梯度初始化为零，optimizer.step()更新所有参数。

问题13. model.eval()产生的效果？

答：model.eval()将模型置为测试模式，不启用 BatchNormalization 和 Dropout。

问题14. model.state\_dict()的目的是？

答：model.state\_dict()的目的是以字典形式获得模型的各个参数。

问题15.何时系统自动进行bp？

答：PyTorch 中的 autograd 包提供了这一功能。当使用自动求导时，神经网络的前向传播就定义一个计算图；图中的节点将是 Tensors，边表示从输入张量产生输出张量的函数。通过图反向传播可以轻松计算梯度。

用 Variable 对象来包装 Tensors。 一个 Variable 表示计算图中的一个节点。如果 x 是一个 Variable，那么 x.data 则是一个 Tensor，x.grad 是另一个用来保存 x 关于某个标量值的梯度。PyTorch 的 Variable 拥有和 PyTorch 的 Tensors 一样的 API：几乎所有 Tensors 上的操作都可以在 Variable 上运行；不同之处在于使用 Variables 定义了一个计算图， 允许自动计算梯度。

问题16.如果自己的层需要bp，如何实现？如何调用？

答：Function与Module都可以对pytorch进行自定义拓展，使其满足网络的需求。

Function一般只定义一个操作，因为其无法保存参数，因此适用于激活函数、pooling等操作；Module是保存了参数，因此适合于定义一层，如线性层，卷积层，也适用于定义一个网络。

Function需要定义三个方法：\_\_init\_\_, forward, backward（需要自己写求导公式）；Module：只需定义\_\_init\_\_和forward，而backward的计算由自动求导机制构成。

可以不严谨的认为，Module是由一系列Function组成，因此其在forward的过程中，Function和Variable组成了计算图，在backward时，只需调用Function的backward就得到结果，因此Module不需要再定义backward。

Module不仅包括了Function，还包括了对应的参数，以及其他函数与变量，这是Function所不具备的。

问题17. Finetune时，有时还要固定某些层不参与训练，请回答如何freeze某些层。

答：冻结某些层参数的训练有3种方法：

1. 通过切断某个模块的梯度流，但是这个会导致该模型前面的所有模型也没有梯度。不推荐这种方法。
2. 通过设置某个模块的所有变量的requires\_grad=False。

示例代码：

for param in model.parameters():

param.requires\_grad = False

1. 在优化器内设置需要进行梯度更新的变量。

示例代码：

trainable\_vars = list(model.stgcn.weight\_model.parameters())+ \

list(model.stgcn.fcn.parameters())+ \

list(model.stgcn.data\_bn.parameters())+ \

list(model.stgcn.dim\_map.parameters())+ \

list(model.aux\_cls.parameters())

opt = torch.optim.SGD(trainable\_vars , lr=1e-4, momentum=0.9)