使用和修改

PyTorch 提供了很多内建的模型可供我们使用，并且分为预训练/未训练版本 (Models and pre-trained weights — Torchvision 0.22 documentation) 为了让这些内建模型适配我们的任务，常用的做法是保持前面的层不变，只修改最后输出的全连接层：

PYTHON

```python
import torch
import torchvision.models
# 需要从网上下载模态，其实下载的是模型各层的参数
resnet18_true = torchvision.models.resnet18(pretrained=True,
progress=True)
# 只会简单定义模型结构并初始化参数，不用下载
resnet18_false = torchvision.models.resnet18(pretrained=False)

# 现在我想要修改 Resnet18 模型，让它的输出层输出 10 个类别，而不是 1000 个类别
# 替换最后的 FC 层
resnet18_true.fc = torch.nn.Linear(512, 10)
print(resnet18_true)
```

保存和加载

虽然放在网络模型的笔记里，但是其实正常训练后的模型也是一样保存和加载的。

保存的方式有两种：

1. 保存模型结构和参数
2. 用字典形式，仅保存模型参数 官方推荐第二种方式，占用空间更小，而且可以避免兼容性问题

对应的加载方式也有两种：

1. 直接加载模型
2. 先定义模型结构，再加载参数 需要注意的是，对于第一种加载方法，需要确保加载模型的文件内，可见模型的结构定义，否则会报错类不存在。

示例代码：`model_save.py`：

PYTHON

```python
import torch
import torchvision.models
# 需要从网上下载模态，其实是模型各层的参数
resnet18_true = torchvision.models.resnet18(pretrained=True,
progress=True)
# 只会简单定义模型结构并初始化参数，不用下载
resnet18_false = torchvision.models.resnet18(pretrained=False)

# 方式1：保存模型结构和参数
torch.save(resnet18_false, "resnet18_false1.pth")
# 方式2：保存模型参数，以字典的形式（官方推荐只保存参数，占用空间更小，且避免兼容性问题）
torch.save(resnet18_false.state_dict(), "resnet18_false2.pth")
```

`model_load.py`:

PYTHON

```python
import torch
import torchvision

# torch 中包含 resnet18 类的定义，所以没问题
# 但是如果是自定义的模型结构，就需要在本文件中引入/定义对应类，再加载，否则使用时会报未定义的错
# 加载方式1
resnet18_false1 = torch.load("resnet18_false1.pth")
print(resnet18_false1)
# 加载方式2
resnet18_false2 = torchvision.models.resnet18(pretrained=False)
resnet18_false2.load_state_dict(torch.load("resnet18_false2.pth"))
print(resnet18_false2)
```