

AIPND P4 通关指南



# 文档说明

1. 本文档旨在帮助AIPND VIP class的学员更好地理解神经网络章节的知识结构体系，以及帮助学

员顺利通关P4，本导学的内容包括：

（1）将课程中平行分布的各知识结构进行梳理，整理成为知识图谱（含对应知识点和章节）；

（2）神经网络训练的通用框架和流程；

（3）P4部分题意解析及避雷，其中附有补充资料链接及示例代码；

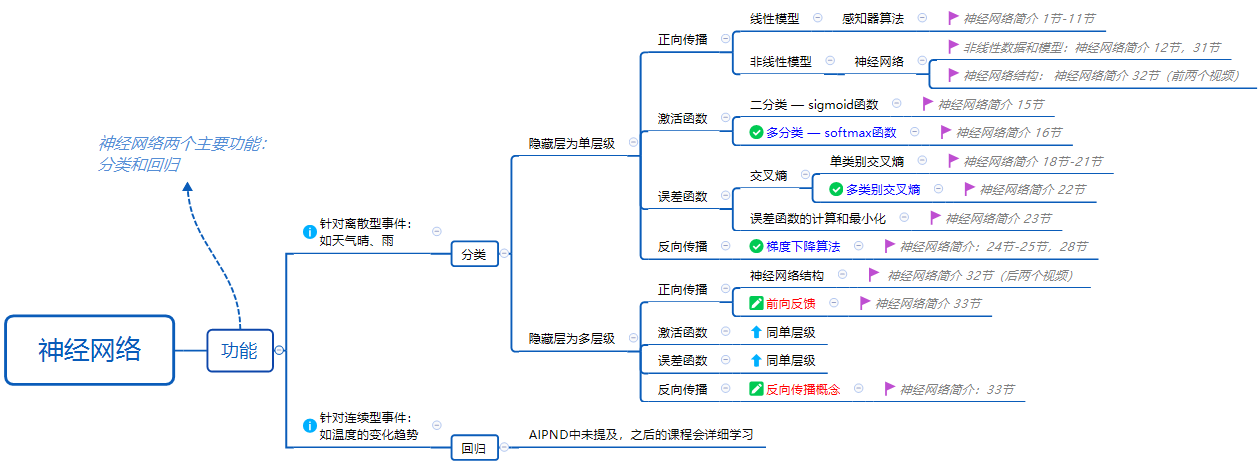
2. 本文档仅整理了P4完成过程中易出现的典型问题，以及相应的建议解决方案，由于每个人代码不同，如果学员在项目过程中遇到文档中没有涵盖的问题，或者文档中提供的方法不能解决当前出现的问题，请及时询问自己的专属助教。

祝你学习愉快，通关顺利！

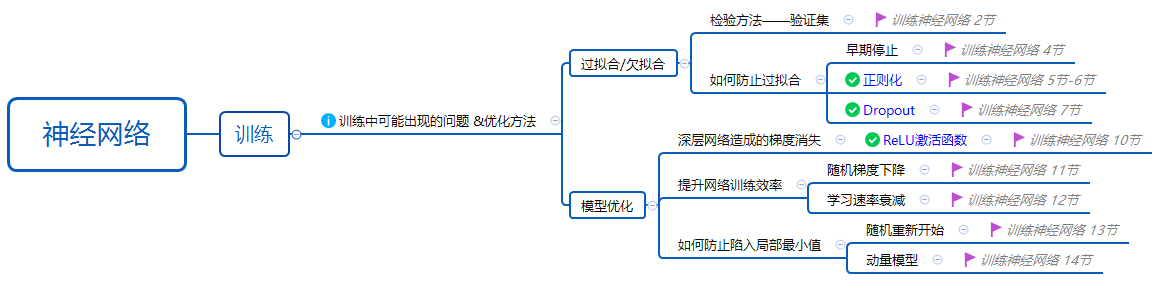
# 神经网络——知识地图

\*说明：以下三个知识地图中，蓝色和红色字体对应**P4中出现的知识点**，蓝色表示简单调用即可实现，红色表示需要自己书写框架。

其他的知识点虽不一定与P4有关，但也很重要哦！我们将在**之后的课程**中详细学习这些知识点的理论基础和应用方法。



知识地图1：神经网络的功能

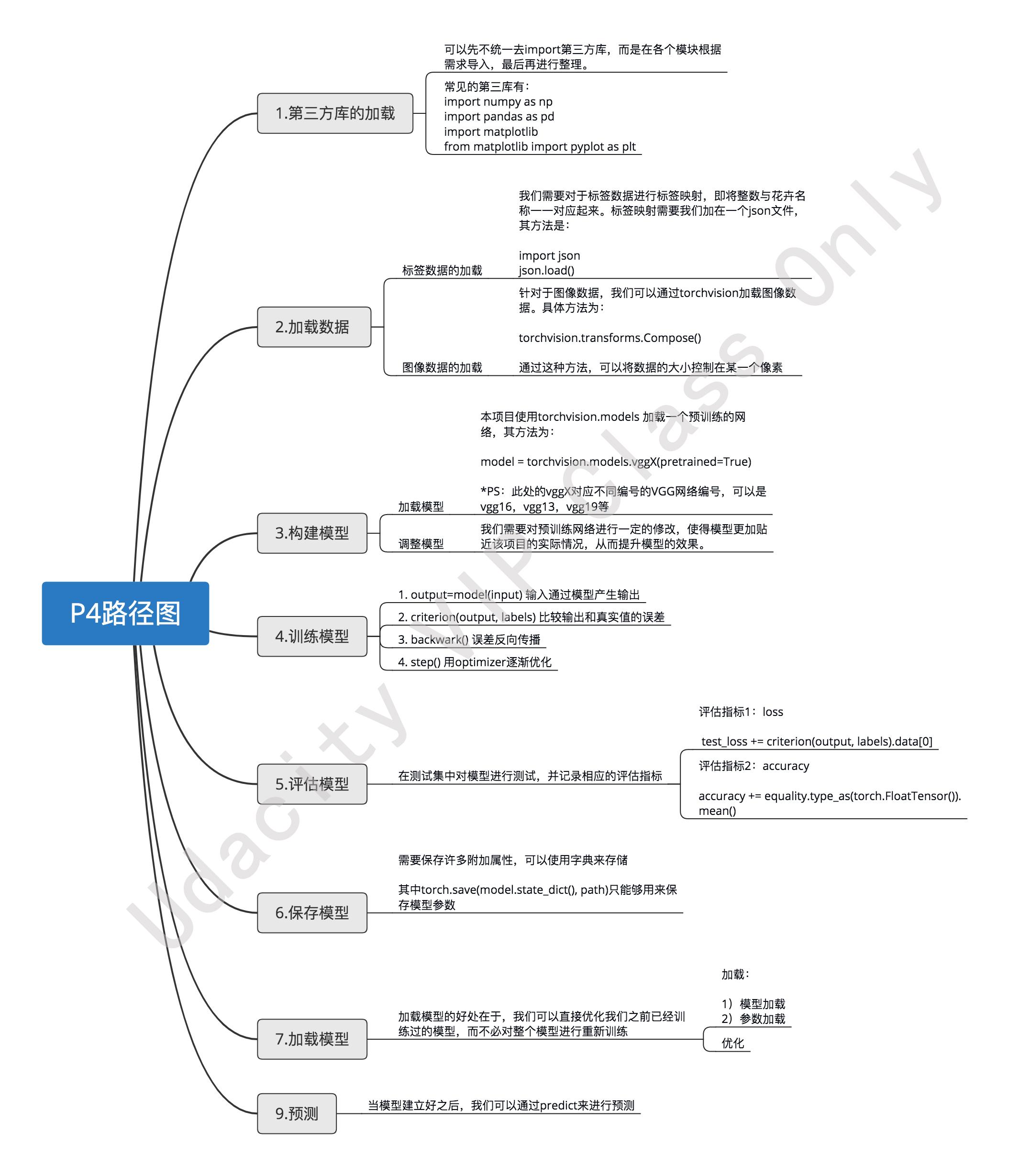


知识地图2：训练神经网络



知识地图3：实现一个符合要求的神经网络

# 如何训练一个神经网络（以P4为例）



# P4 项目解析及避雷——part 1

1. **目标**：训练一个图像分类器，能够识别102中花卉图像；
2. **数据集**：workspace中已有现成的数据集，可点击左上角jupyter查看文件夹中的数据集，共有三个数据集train，valid和test，不需要再次划分，用代码直接导入即可；
3. **项目解析**：

3.1 导入所需的软件包（import）

Tips：可以在写代码时，根据自己需要的包，边写边导入；

3.2 导入数据集

Tips：（1）主要考察对torchvision的使用，可参见pytorch[官方中文文档](https://pytorch-cn.readthedocs.io/zh/latest/)；

（2）课程中有关于导入数据集的教学课程，请参见[知识地图](#_xoiwraij6osw)；

（3）所有的数据集，需要先经过resize成为256\*256，才可以从中间裁剪出224\*224，否 则会导致图片可能裁剪不到有花朵的部分，影响训练和测试性能；

（4）valid和test两个未知的数据集，均不需要任何的翻转和变型，但是同样需要resize到256\*256，之后进行裁剪；

（5）训练集需要设置shuffle=True进行打乱，测试集和验证集不需要，保持默认False；

3.3 构建和训练分类器

Tips：（1）利用迁移学习，用[torchvision.models](https://pytorch-cn.readthedocs.io/zh/latest/torchvision/torchvision-models/)中调用训练好的模型（项目中提示为VGG），预训练后，自己根据预测目标修改模型的分类器；

（2）修改分类器时，除了需要修改输出层的节点数，还需要在输出时加上softmax或者logsoftmax的激活函数；

（3）注意将数据和模型都转移到GPU，例如：model.cuda();

（4）建议将训练集训练网络、验证集验证网络分别写成两个函数（课程中写到了一个代码中），例如：def train(epochs，model)，便于调试代码，以及增加程序的可读性；

（5）训练时，若发生了过拟合或者是精度不高的情况，可以立刻停止训练，并调整参数重新训练；

（6）网络构建和训练的课程可参见[知识地图](#_xoiwraij6osw)；

3.4 测试网络

（1）网络测试的课程可参见[知识地图](#_xoiwraij6osw)；

3.5 保存检查点

Tips：（1）需要保存许多附加属性，可以使用字典来存储，例如：torch.save({  
'epoch':epochs；...}，“保存路径”);

（2）课程中关于检查点保存的内容见[知识地图](#_xoiwraij6osw)，注意torch.save(model.state\_dict(), path)的保存方法只能保存模型参数；  
 3.6 加载检查点

Tips：（1）写成函数形式方便调用，加载的数据为之前checkpoint保存的数据；

（2）保存好模型之后，再次打开workspace时不用再对网络进行训练，直接运行加载模型的程序，将之前训练好的模型加载出来；

3.7 图像处理

Tips：（1）注意按题目要求使用thumbnail或resize方法，以保证最小的边为256像素，而非使用torchvision处理数据集，可以使用图像行与列的比值确定最小的边；

（2）从中心裁剪图像时，可以先定义图像的边界，再调用crop裁剪；

（3）PIL 图像和 Numpy 数组中，颜色通道是第三个维度，但是Pytorch要求是第一个，在使用transpose改变维度时，可以多使用print (image.shape)，输出当前图像的维度，防止出现维度错乱；

（4）输入图像的路径可以自己选定，例如'./flowers/train/1/image\_06735.jpg'，利用PIL中的Image.open加载图像，调用自己写的预处理函数和项目中给的imshow函数，显示预处理后的图像结果；

3.8 类别预测

Tips：（1）可能在之前的代码中，image被转化为numpy的数组形式，若需要转回tensor，可以使用torch.from\_numpy(image).type(torch.FloatTensor)；  
 （2）如果需要使用CPU，或者要将数据集迁移到CPU，直接.cpu()；

3.9 检查运行状况

Tips：（1）cat\_to\_name.json 文件中的数据是以字典的形式存储的；

（2）主要考察利用matplotlib进行可视化的能力；

# P4 项目解析及避雷——part 2

1. **目标**：利用argparse灵活调参；
2. **数据集**：和part 1中相同；
3. **项目解析**：

Tips：（1）利用argparse模块，可以动态地灵活地更改网络中的参数，一个示例框架如下：

argparse中定义learning rate=0.001，

parser = argparse.ArgumentParser(Description=“”)

parser.add\_argument("--learning\_rate",type = float,default = 0.001, help = "learning rate" )，

args = parser.parse\_args()

则在代码中，学习率lr写为：

optimizer = torch.optim.Adam(model.classifier.parameters(), lr=args.learning\_rate)  
在0.001不合适时，直接改写argparse中的值，即可改变原代码中的学习速率；

（2）关于argparse命令行的应用，课程中的教学可参见AIPND的【python入门】章节

——Lab图像分类；

（3）关于argparse的[补充材料](http://wiki.jikexueyuan.com/project/explore-python/Standard-Modules/argparse.html)，其中有关于argparse的详细说明（中文）；

（4）定位参数和可选参数的定义不同，详情参见补充材料，例如：项目train.py中，

Data\_dir是定位参数，其余为可选参数；

（5）定义完argparse模块后，可以输出查看定义的参数是否正确，例如print(args.learning\_rate);

（6）终端运行代码时，可参考如下步骤：

* ls查找文件夹
* cd 文件夹名 （进入当前文件夹）
* python train.py 定位参数（如data\_dir的名称）——gpu（判断GPU是否开启）