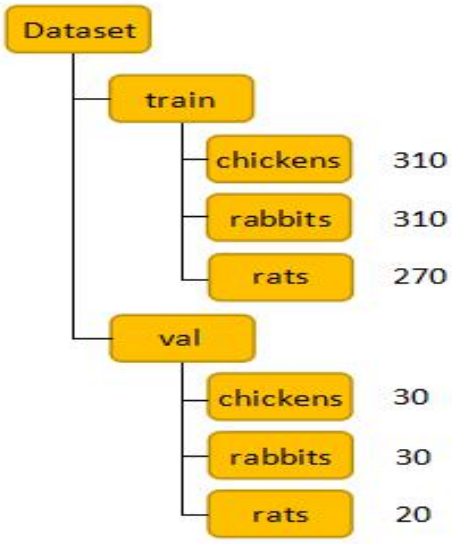
实验报告

一、 代码说明

（一） 数据集

数字为相应图片个数，原文件中图像命名为“物种+编号”，eg：rabbits 000



（二）文件名称说明

Images-rename.py：将数据集中图片乱七八糟的名字批量重命名

Classes_make_anno.py: 生成“纲”分类的训练与测试数据标签，选兔子和鸡作为数据集，预测属于哺乳纲(Mammals)还是鸟纲(Birds)

Species_make_anno.py : 生成“种”分类的训练与测试数据标签，预测是兔子、老鼠还是鸡

Multi_make_anno.py : 生成同时对“纲”、“种”进行分类的训练与测试数据标签，同时预测是兔子、老鼠 还是鸡，哺乳纲还是鸟纲

Classes_Network.py: 用于“纲”分类的网络

Species_Network.py : 用于“种”分类的网络

Multi_Network.py: 用于“纲”、“种” 多分类的网络

Classes_classification.py：程序主体，用来进行模型训练/验证，并调用训练好的模型进行预测

Species_classification.py: 程序主体，用来进行模型训练/验证，并调用训练好的模型进行预测

Classes_train_annotation.csv/Classes_val_annotation.csv: 用于纲分类的标签，由Classes_make_anno.py 生成

Species_train_annotation.csv/Species_val_annotation.csv: 用于种分类的标签，由Species_make_anno.py 生成

Multi_train_annotation.csv/Multi_val_annotation.csv: 用于纲、“种”多分类的标签，由Multi_make_anno.py 生成

二、 实验目标

完成动物纲(Classes)分类，预测该动物是属于哺乳纲(Mammals)还是鸟纲(Birds)；完成动物种(Species)分类，预测该动物是兔子、老鼠还是鸡；完成多任务分类， 同时预测该动物的“纲”和“种”

三、 实验过程

（一） 数据预处理

1 数据标签

为了得到训练标签，针对不同的任务生成相应的标签集。

Stage1 中需要将哺乳纲、鸟纲的数据分别标为 0，1 作为训练标签;

Stage2 中将兔子、老鼠、 鸡的数据分别用 0，1，2 作为标签。

2 数据读入

定义数据的变换方式及顺序，train 和 test 可不同。

加载数据的方式：torchvision.transforms 的 transforms 系列

torch.utils.data 中 DataLoader 函数

定义数据组成，如:

Stage1 中我们的数据是 sample = {'image': image, 'classes': label_classes};

Stage2 中我们的数据是 sample = {'image': image, 'species': label_species};

Stage3 中我们的数据是 sample = {'image': image, 'classes': label_class, 'species':label_species}

3 数据验证

在训练之前需要验证我们的数据集与标签集是否处理正确。可任意输出数据图片与对应标签，验证是否符合我们的设定。

（二） 模型设计

1 搭建网络：Function layer 的选择以及安放位置，可根据训练效果自行调整。 分类器数目根据分类任务需要进行设置

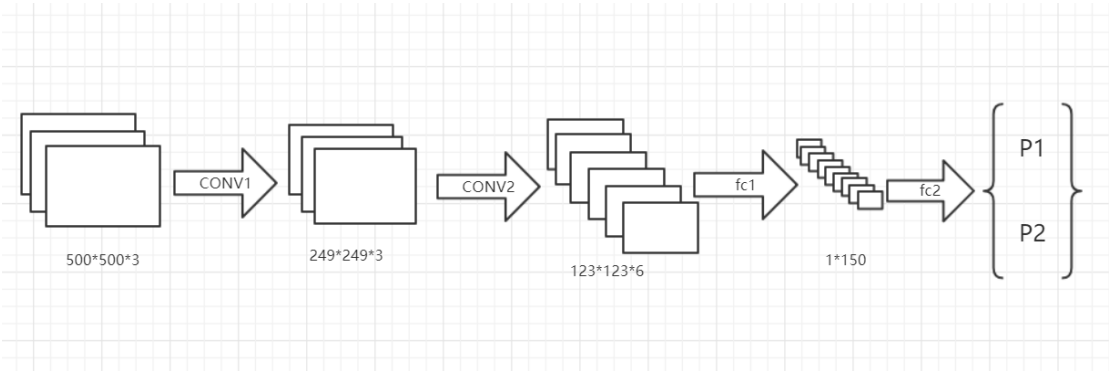


图 1 所搭建网络示意图

图注：这里我们使用了 cov 卷积+maxpoolong 池化+relu 激活的两层网络，最后使用全连接来进行 flatten 操作的方式来搭建基本的 network。最终得到两/三个类别的概率向量大小

2 训练、测试网络

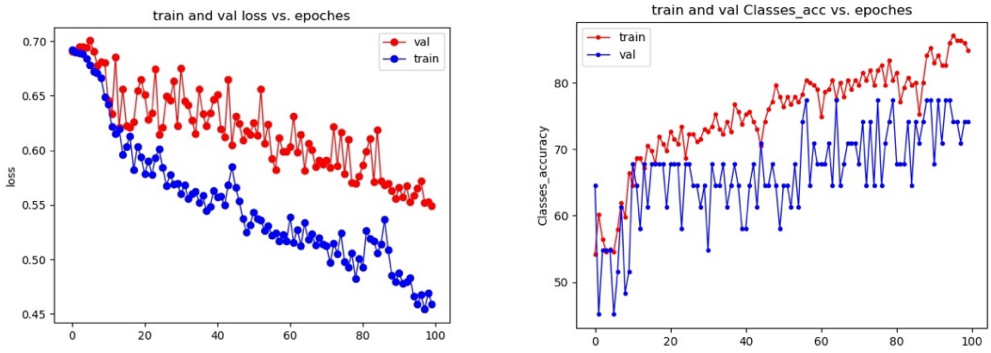
3 调参：可调整 Lr，step 等参数对比训练结果，找到最优方案。 这里我们使用 Adam 方式进行梯度下降与 optimizer （也可以使用 SGD 的方法）

4 评估：记录 loss 和 accuracy，并根据测试集中的 accuracy 记录最优模型

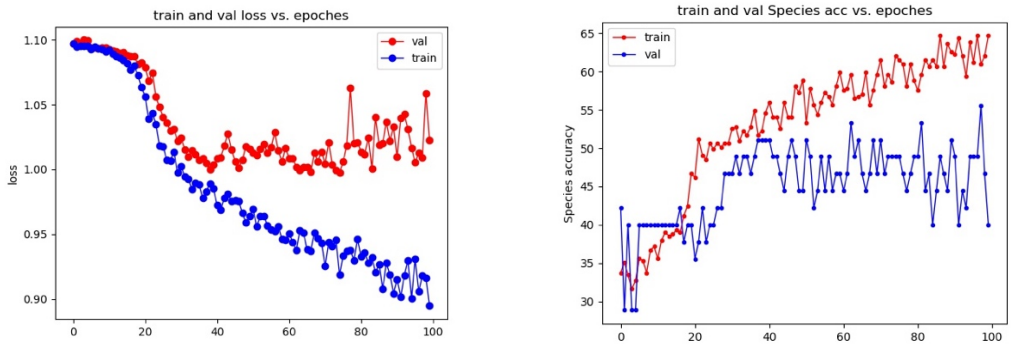
Stage1、Stage2 单任务训练时，loss 就是单一分类任务的 loss; Stage3 多任务训练时，将每个任务的 loss 进行线性加权作为训练的 loss

5 可视化：用训练好的模型对一些数据进行预测，直观地看看训练好的预测器效果如何。

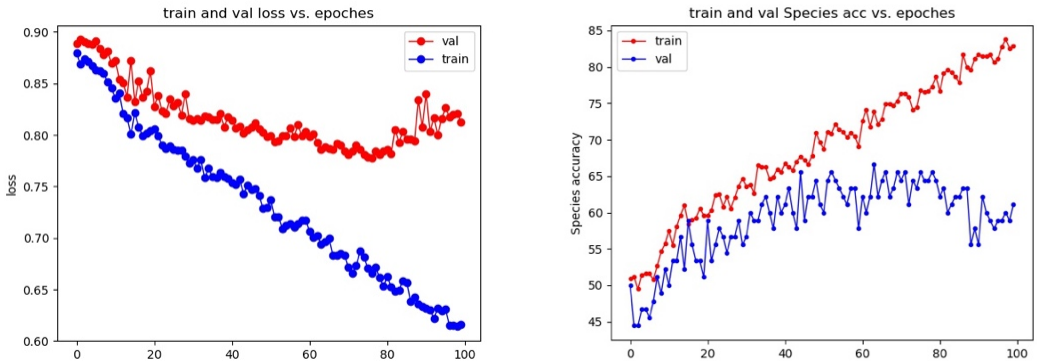
四、 实验结果与分析



Stage 1 对 classes 的二分类结果



Stage 2 对 species 的三分类结果



Stage 3 对 classes 以及 species 的三分类结果

在 loss 值的处理上，是使用了将 classes 与 species 的 loss 进行相加求平均的方式。在同样的网络之中，二分类问题的准确率明显大于三分类问题，并随着类别的增加而下降 Acc 中，我们判断将两个任务都判断准确的作为正确的结果，使用判断正确的/总数据集大小来计算。

由于数据集过小，所以非常容易出现过拟合的情况，此时 train 的 loss 会停滞不变，比如 stage2 和 stage3 中，可以通过数据增广、增加 epoch 并渐小学习率，调整 batch size 的方式来进行优化。

说明：该数据集大小有限，这里不再进行参数的调试。但很适合小白使用此案例来进行 pytorch 图像分类的入门。

多分类模型的搭建，具体可参考：

https://www.analyticsvidhya.com/blog/2019/01/build-image-classification-model-10-minutes/?utm_source=blog&utm_medium=multi-label-image-classification