|  |
| --- |
| 哈尔滨工业大学(深圳) |
| **《人工智能》实验报告** |
|  |
| **实验二**  **深度学习实现花卉识别**   |  |  | | --- | --- | | 学 院: | 计算机科学与技术 | | 班 级: | 1 |   姓 名:   |  |  | | --- | --- | | 学 号: |  | | 学 期: | 2022春 | | 实验地点: |  | | 实验教师: |  |   目录  [1 实验内容 2](#_Toc102079325)  [2 实验记录 2](#_Toc102079326)  [2.1. 实验环境 2](#_Toc102079330)  [2.2. 数据集处理 3](#_Toc102079331)  [2.3. 模型构建与训练 4](#_Toc102079332)  [2.3.1. CNN模型 190110121-杨航 4](#_Toc102079333)  [（1） 模型定义 4](#_Toc102079334)  [（2） 模型训练 7](#_Toc102079335)  [2.3.2. CNN模型\_tensorflow框架（190110130\_向柳嫒） 9](#_Toc102079336)  [（1） 模型定义 9](#_Toc102079337)  [（2） 模型训练 11](#_Toc102079338)  [2.3.3. \*\*\*模型 13](#_Toc102079339)  [（3） 模型定义 13](#_Toc102079340)  [（4） 模型训练 13](#_Toc102079341)  [2.3.4. 利用VGG16网络架构进行的迁移学习（by 谢俊安） 14](#_Toc102079342)  [（1） 模型定义 14](#_Toc102079343)  [（2） 模型训练 15](#_Toc102079344)  [2.4. 实验结果 20](#_Toc102079345)  [2.5. 深度学习框架对比 24](#_Toc102079346)  [3 总结 27](#_Toc102079347)  [3.1. 问题及解决方法 27](#_Toc102079349)  [3.2. 实验的启发、总结及建议 29](#_Toc102079350)  [4 参考文献 30](#_Toc102079351)  *实验报告内容包含但不限于以下内容，如有补充请用红色\*标注。*  *注意标注每部分的作者，作者可标注在小标题上。* |

# 实验内容

*简要概述本次实验的内容，用到哪些框架。*

基于给定的数据集，在本地分别用TensorFlow、MindSpore、Pytorch框架实现花卉识别。须自己一层层实现模型的定义。

# 实验记录



## 实验环境

*阐述实验用到的环境，包括操作系统、开发软件、用到的库及版本号。*

杨航：

实验环境：Python3

操作系统：windows

开发软件：Pycharm2021

库：torch、numpy、sklearn

向柳嫒：

实验环境：Python3

操作系统：windows

开发软件：Pycharm2021

库：tensorflow、numpy、sklearn

谢俊安：

* 操作系统：MacOS Monterey12.3.1
* 开发软件：PyCharm
* 库及版本号：
  + Tensorflow 2.6.2

Cv2

Keras 2.6.0

numpy 1.19.5

matplotlib 3.3.4

scikit-learn 0.24.2

## 数据集处理

*阐述对数据集进行的预处理，包括但不限于数据集划分、数据扩充、数据重采样、数据增强等，写明处理的方法、理由及对实验实验结果的影响。*

杨航：

数据集采取随机划分的方式，使用sklearn库，按照8：2的比例将数据集划分为训练集和测试集。同时为了提高模型的收敛速度，对数据进行的标准化处理。

对所确定模型，尝试了三个不同的种子进行三次随机划分，将最后得到的在测试集上的表现取了平均，得到了最终的结果。三次随机划分所得到的准确率都较相似，表明模型具有一定稳定性。

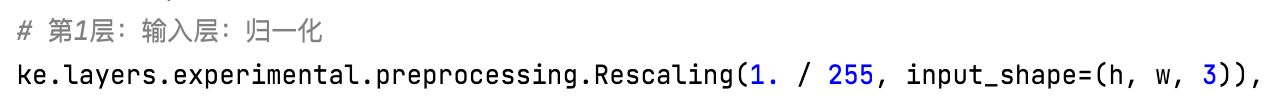
向柳嫒：

用sklearn库里的train\_test\_split函数划分数据，测试集占10%，并设置随机数种子：

图形用户界面, 文本

描述已自动生成

用keras库里的preprocessing将数据标准化处理：



处理后加快了模型的速度，且随即划分的结果都比较相似，表明模型具有一定稳定性。

谢俊安：

数据集划分为训练集:测试集=8:2，相对是比较合理的划分方法，既给予了模型足够的训练样本，也获得了充足的测试样例来验证模型的正确性。

## 模型构建与训练

*以下部分以模型为单位写，每种模型均需要包括模型定义和模型训练部分。*

### CNN模型 190110121-杨航

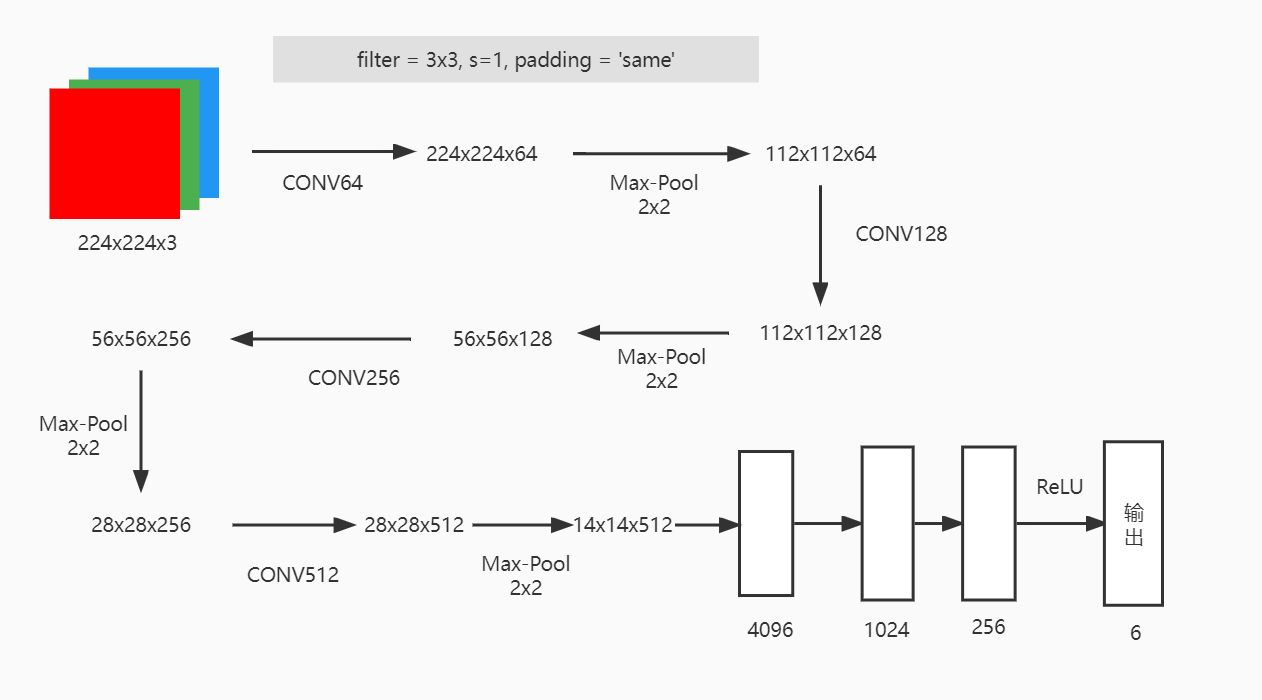
### 模型定义

*阐述本次实验使用的模型，详细说明模型结构及选型，包含但不限于卷积层、池化层、全连接层、损失函数等主要参数及选型理由，提供完整的模型结构图，截图模型定义代码。*

使用框架：pytorch

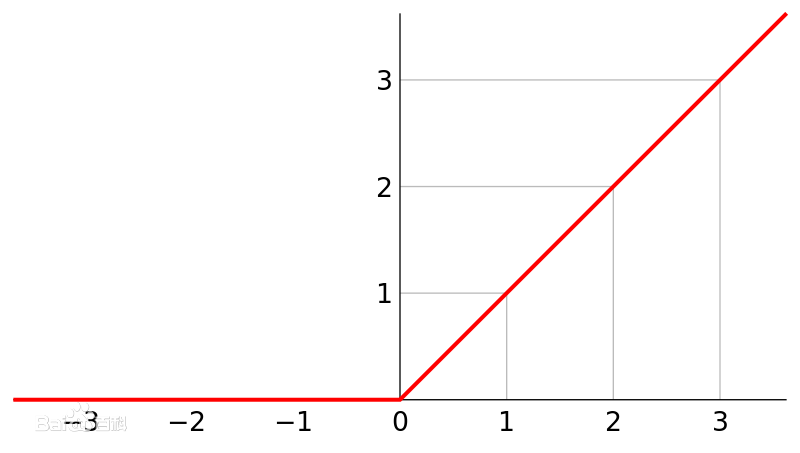
模型参考：VGG16

模型结构：



优化器：Adam

激活函数：ReLU



激活函数优势：

* 仿生物学原理
* 更有效率的梯度下降及反向传播，避免了梯度消失和梯度爆炸
* 简化了计算过程

防止过拟合：在全连接层进行随机dropout,dropout率为0.5

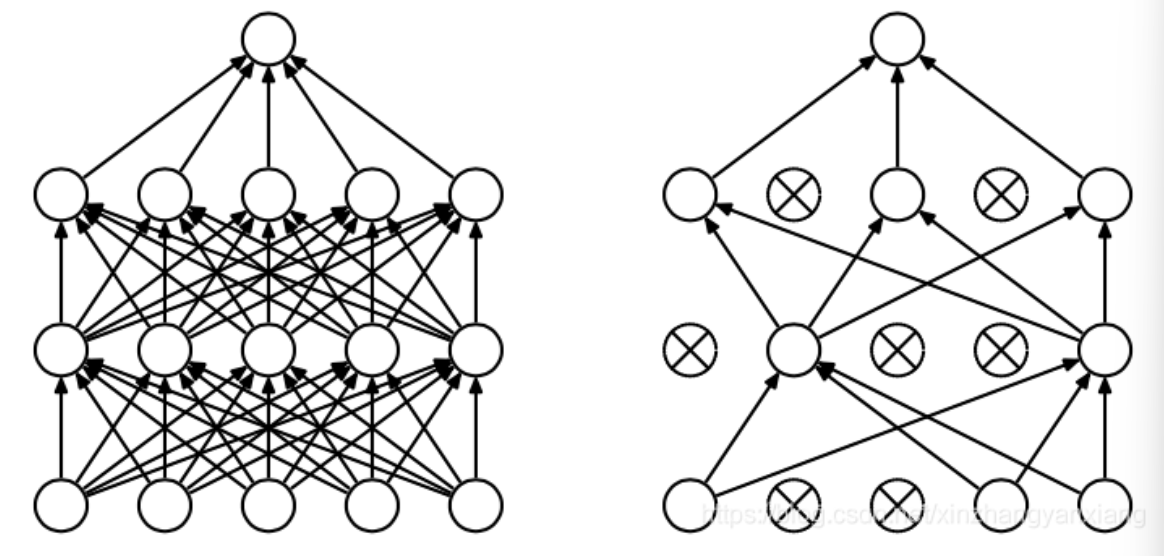
Dropout的优势及原理：

优势：

* 找到更“瘦”的网络，减少训练时间
* 防止过拟合

原理：

进行dropout前的网络如下图左边所示，dropout后的如右边所示：



对于一个有N个节点的神经网络，有了dropout后，就可以看做是2^n个模型的集合了，但此时要训练的参数数目却是不变的，这就解脱了费时的问题。

经过交叉验证，隐含节点dropout率等于0.5的时候效果最好，原因是0.5的时候dropout随机生成的网络结构最多。

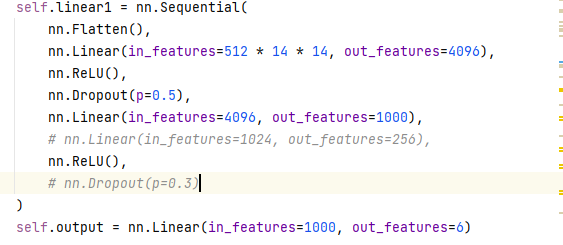
选型理由：VGG16模型能够很好的适用于分类的任务，但由于本次实验的数据量较小，因此在网络原本的基础上删除掉了一部分卷积层，将网络结构简化了许多。在选择VGG16前，还尝试过LeNet5，但由于网络较浅，在同样训练次数的表现上，效果不如VGG16。

定义代码：

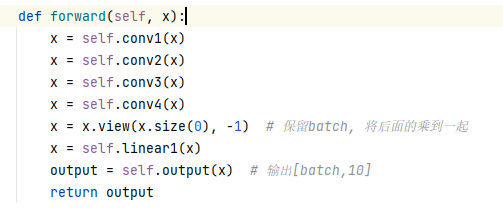
本地：



云上训练的模型线形层：



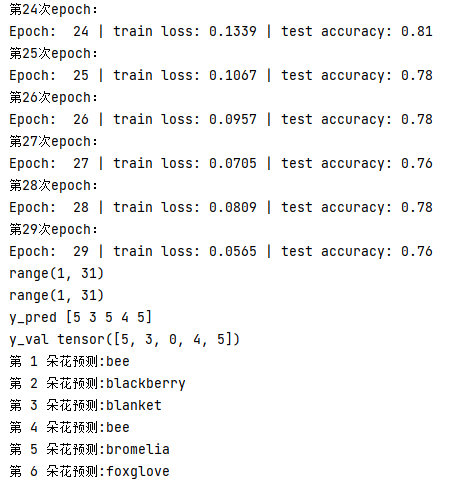
前推函数：

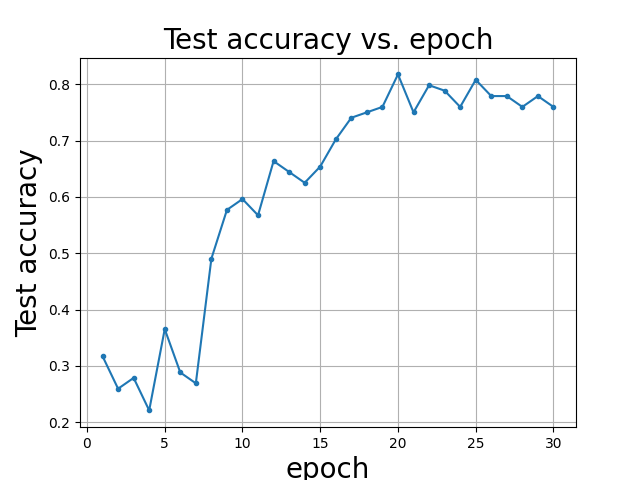
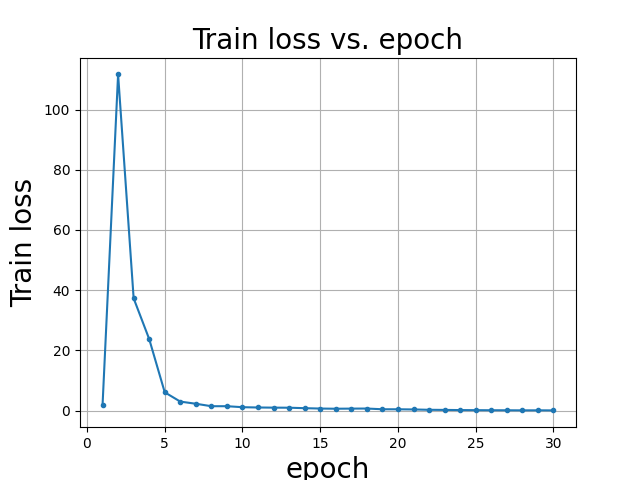


### 模型训练

*详细说明模型在训练集上的训练速度、收敛速度、收敛精度、超参数设置与调整、优化算法选择等，提供训练日志或loss曲线的截图。*

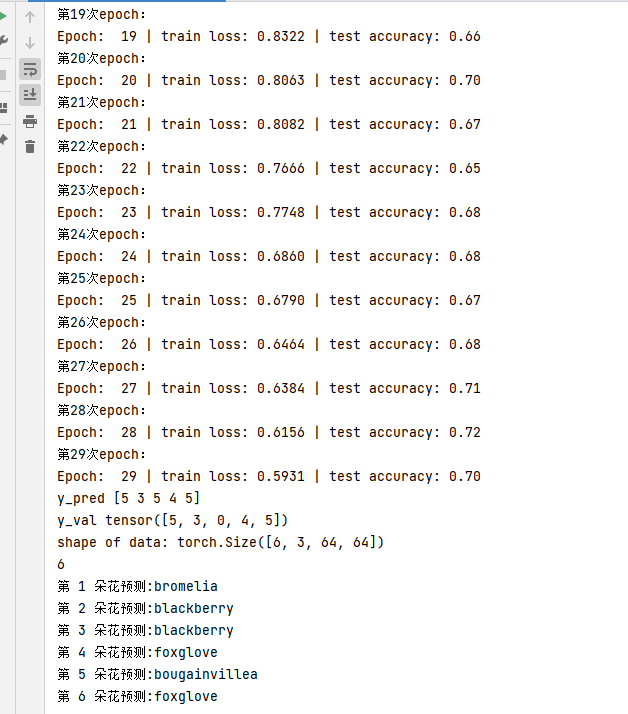
在lr固定为0.001, batch\_size为64时，训练回合为30时得到的其中一次结果如下：

  
其中loss、accuracy随着epoch的变化如下：



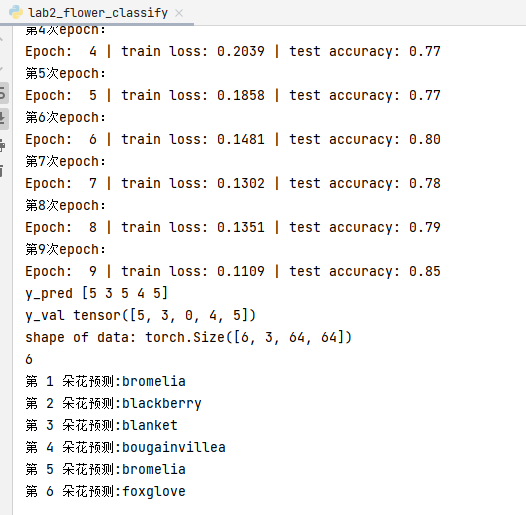
可以见到在大约20回合后，准确率便开始在0.8左右波动，分析可能为学习速率偏高。

当降低学习率/减小batch\_size：



最后在测试的六张图片上表现不佳，且准确率也低于上一表现。

同时，由于网络模型较大，运行速度偏慢，因此尝试了将前一次训练的模型参数保存下来，在下一次训练时先load进网络，调低学习率继续训练的方法。这样单次训练时间较短，并且能够通过结果实时调整参数观察结果，再继续修订模型。这样的方法每次训练回合数较低，但结果优于单次训练。下图为部分结果：



但由于参数文件较大，无法上传到提交平台，因此最终未采取这种方法。

### CNN模型\_tensorflow框架（190110130\_向柳嫒）

### 模型定义

* 模型：卷积神经网络（CNN）
* 模型结构图：

图示

描述已自动生成

* 模型定义代码：

本地：

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

云上训练的模型：

图形用户界面, 文本, 应用程序

描述已自动生成

* 模型选型：
* 第1层：输入层

将数据归一化

* 第2-3层：卷积层，池化层

卷积层：提取输入的不同特征。

卷积核数目为16（决定输出的depth厚度），卷积核尺寸为3\*3，激活函数为relu（加入非线性因素，提高对模型的表达能力）；

池化层：降低维度，缩减模型大小，提高计算速度。

池化窗口尺寸：2\*2。

* 第4-5层：卷积层，池化层

卷积核数目为64，卷积核尺寸为3\*3，激活函数为relu

池化窗口尺寸：2\*2

* 第6层：卷积层

卷积核数目为128，卷积核尺寸为3\*3，激活函数为relu

* 第7层：Flatten层

连接卷积层与全连接层

* 第8层：全连接层

转化成一维的向量，输出维度为128，激活函数为relu

* 第9层：输出层

输出预期结果，参数为花卉类别数：6

### 模型训练

优化器选择的是“adam”，因为经过多次实验发现adam的效果最好；

损失函数选择的是“SparseCategoricalCrossentropy”：

文本, 信件

描述已自动生成

epochs（训练回合）为20时得到的其中一次结果如下：

表格

低可信度描述已自动生成

图形用户界面, 文本, 电子邮件

描述已自动生成

可以看到20回合后，准确率达到71%，损失值约为1.3。

最后测试的六张图片全部正确，结果比较好。

Accuracy曲线截图：

图表, 折线图

描述已自动生成

随着回合数增加，训练集准确率升高至1，验证集总体趋势为上升，最终约为70%。

Loss曲线截图：

图表, 折线图

描述已自动生成

随着回合数增加，训练集丢失值降低至0，验证集丢失值起伏不定，最终约为1.3。

### \*\*\*模型

### 模型定义

*阐述本次实验使用的模型，详细说明模型结构及选型，包含但不限于卷积层、池化层、全连接层、损失函数等主要参数及选型理由，提供完整的模型结构图，截图模型定义代码。*

### 模型训练

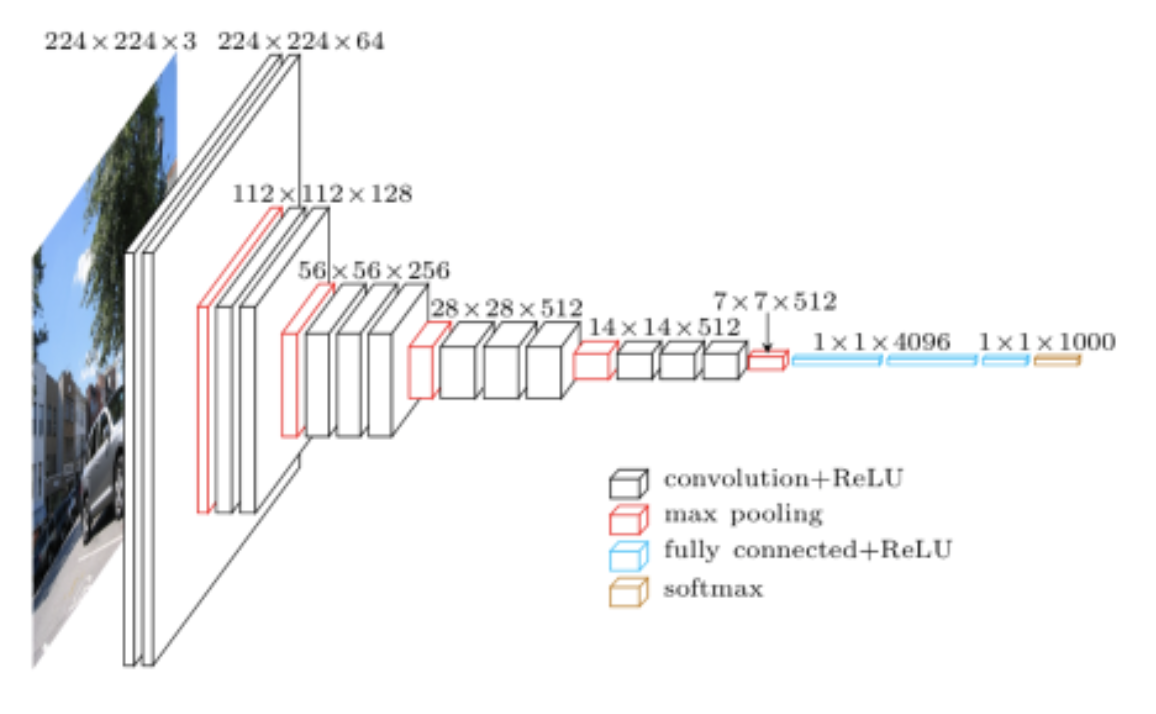
*详细说明模型在训练集上的训练速度、收敛速度、收敛精度、超参数设置与调整、优化算法选择等，提供训练日志或loss曲线的截图。*

### 利用VGG16网络架构进行的迁移学习（by 谢俊安）

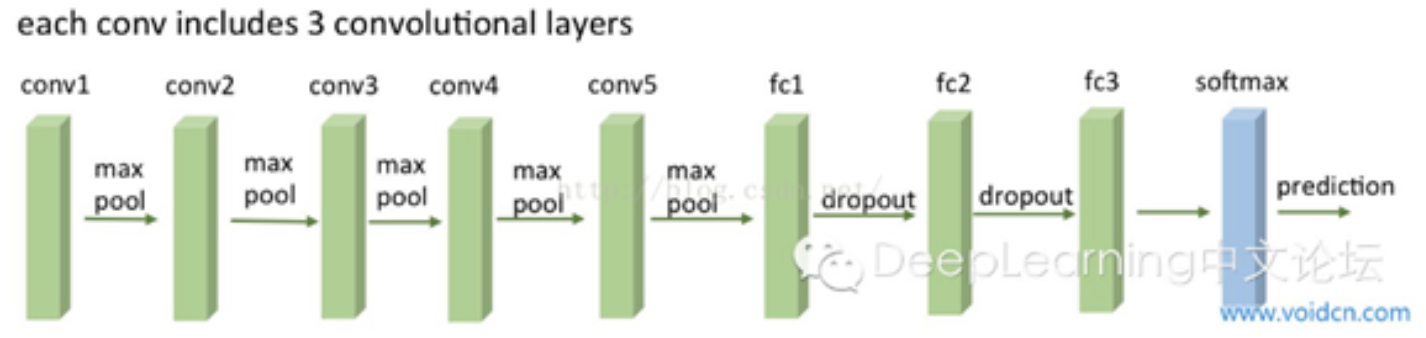
### 模型定义

本模块选用了VGG16网络架构和ImageNet预训练网络进行训练并与本小组自写的模型进行对比参照，VGG卷积神经网络模型在2014ImageNet图像分类与定位挑战赛 ILSVRC-2014中取得在分类任务第二，定位任务第一的优异成绩。

选用的变种是VGG16，这是一个拥有16层的模型，模型结构图如下：



其中具体包括5个卷积层和3个全连接层，VGG16 的第 3、4、5 块：256、512、512个 3×3 滤波器依次用来提取复杂的特征。本实验将顶层的3个全连接网络删除，并进行微调以适应本任务。模型的权重采用ImageNet数据集上预训练的权重。模型的默认输入尺寸是224x224，本任务调整为100\*100以在保持训练精度的情况下降低训练成本。

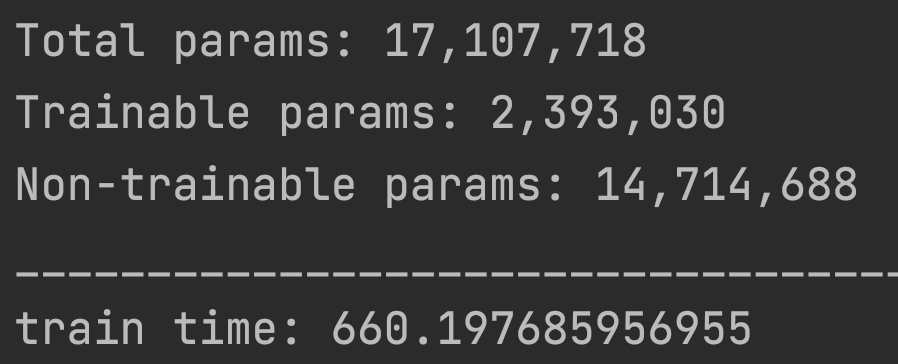


特别地，VGG 的的优越性是通过依次采用多个 3×3 卷积，模仿出更大的感受野效果。模型定义如下：



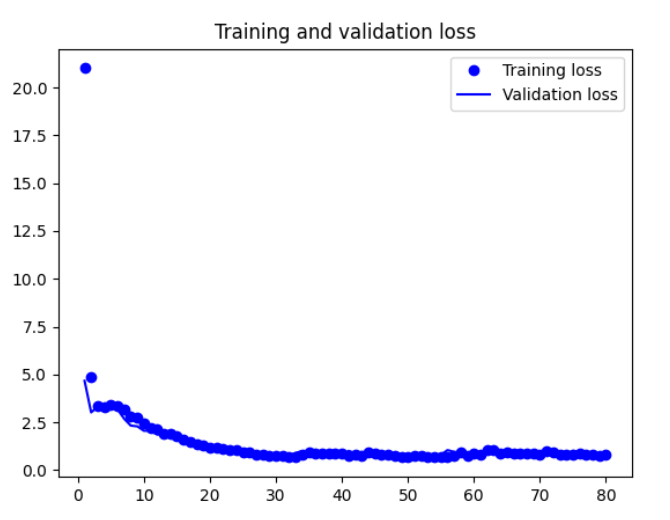
### 模型训练

* 1. 训练速度：

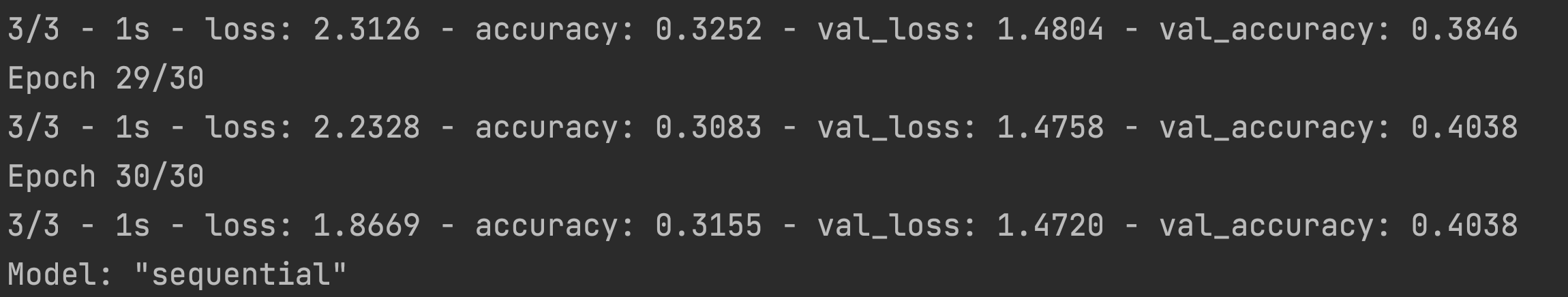


由图可知，模型对精度为100\*100的图片训练80个epoch共耗费660s，训练速度较快。

* 1. 模型基本在epochs=30时已经收敛

****

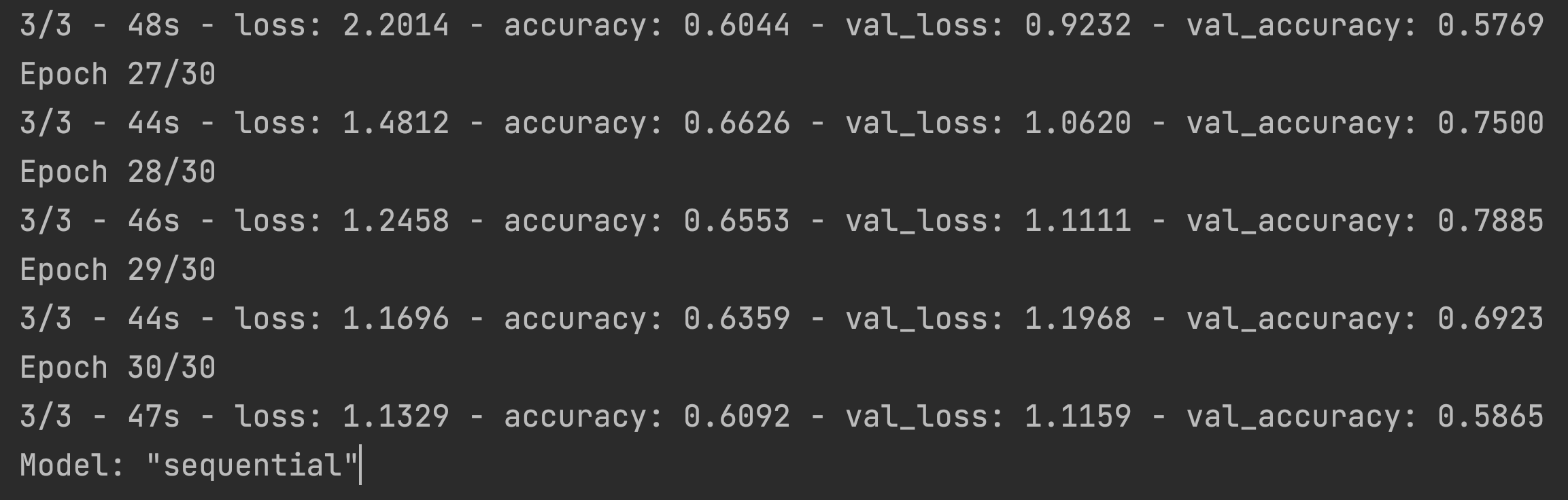
* 1. 超参数：
     1. 最终选择：图片像素为100\*100、epochs=80、batch\_size=200、优化器Adam学习率learning\_rate=0.01、添加l2正则化。
     2. 调整过程：
        1. 图片像素过低导致的精度过低：



分析：如图显示的为32\*32像素的图片样本，个人猜想原因是图片像素被压缩时，物件本身的特征也被平坦化了，导致模型的训练效果不佳。尝试将输入图片像素调整为224\*224.

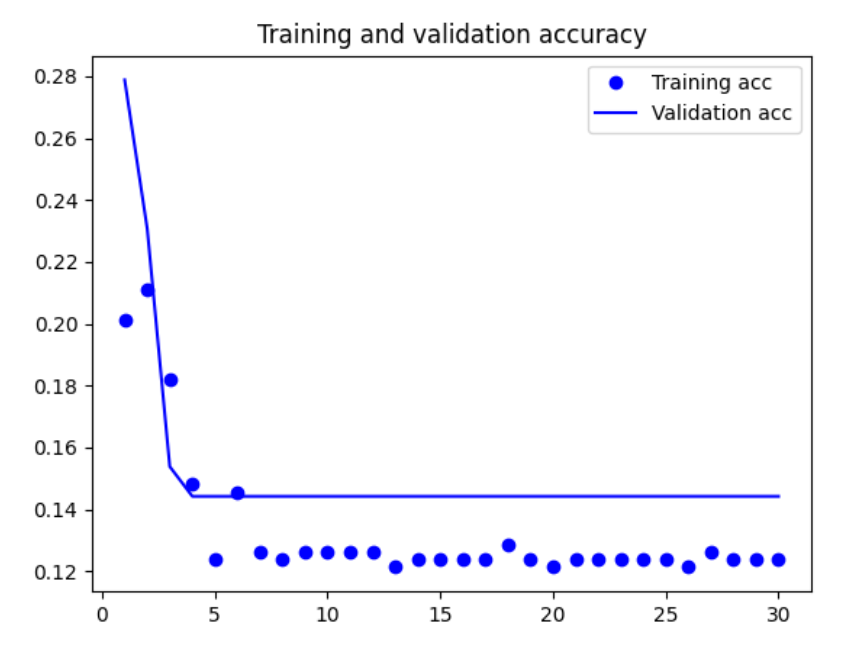
* + - 1. 对原有模型的理解不通透

具体描述：上述问题经调整图片像素之后有了小幅改进，但结果仍不理想（如下图）

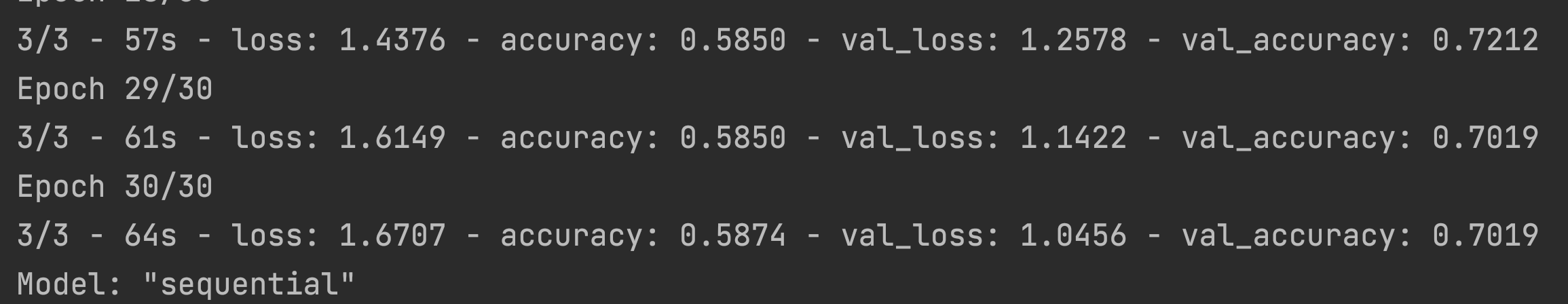


分析原因：没有理解使用的VGG16模型的特性，原模型的最后一个全链接层使用的激活函数是softmax 而不是rula

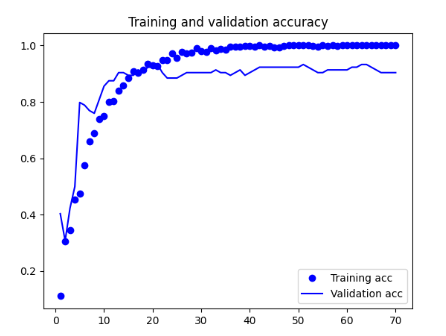
* + - 1. 越训练效果越差



原因及解决办法：样本像素太低，花种的特征已经消失。提高图片像素有很大提升。

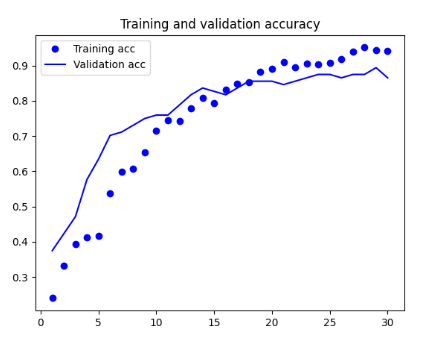


* + - 1. 训练次数过高



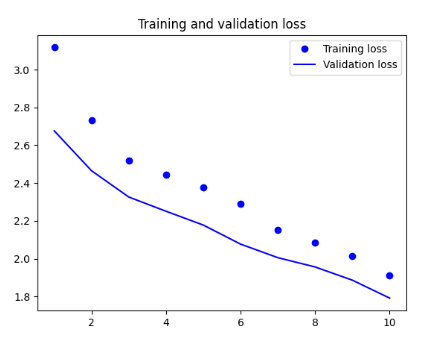
分析：实际上但epochs==30时模型精度已经收敛，趋于稳定，没必要再抬高训练次数，反而造成资源浪费。

* + - 1. 出现模型过拟合的现象



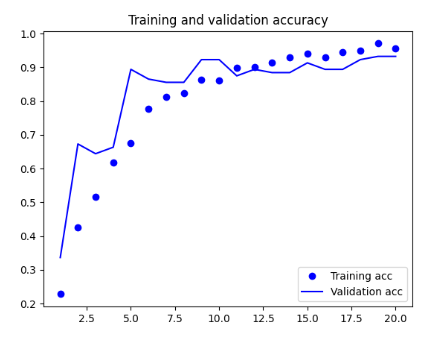
分析原因：学习率过低导致的模型泛化水平差，尝试提升模型学习率0.00001->0.001。

* + - 1. 模型未收敛



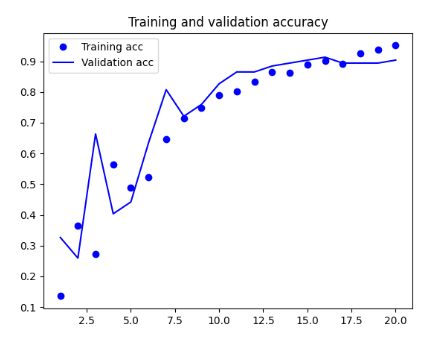
解决办法：提高学习率/增加训练次数

* + - 1. 模型收敛很不稳定



分析原因&解决办法：增加batchsize为原来的N倍时，按照线性缩放规则，学习率应该增加为原来的N。所以应该提升batchsize。

* + - 1. 模型仍旧过拟合

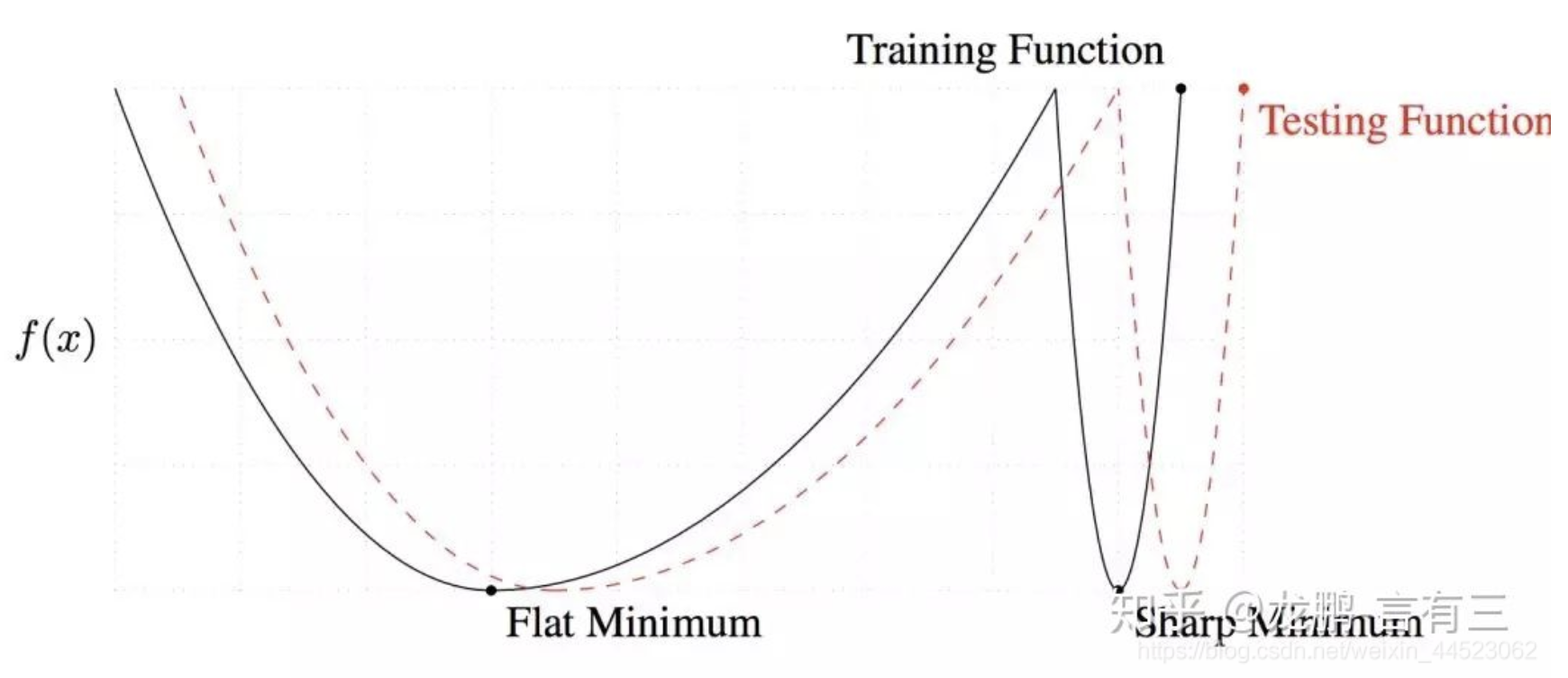


分析原因：

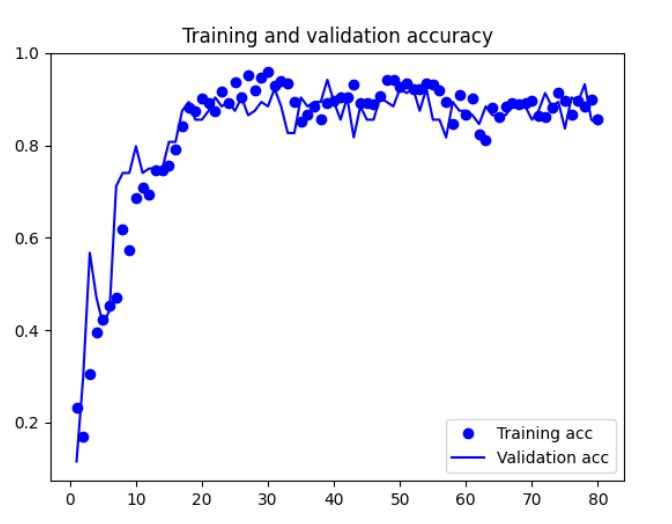
在一定范围内，一般来说 Batch\_Size 越大，其确定的下降方向越准，引起训练震荡越小。但是本次训练中batchsize调整到2000显然有些超出了“一定范围”，而batchsize过大会导致模型收敛到sharp minimum，比batchsize小的模型泛化能力更差。

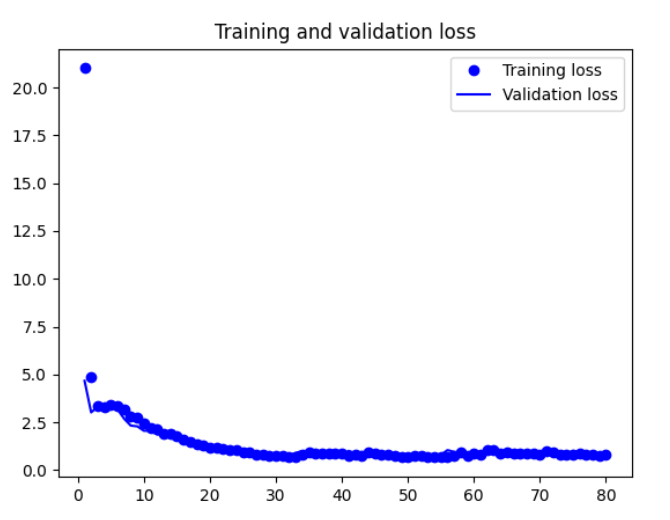
模型训练过度。

解决办法：尝试调整batchsize为200；添加l2正则化防止过拟合；进行dropout率为0.5的dropout防止过拟合。



* + - 1. 拟合效果良好

****

****

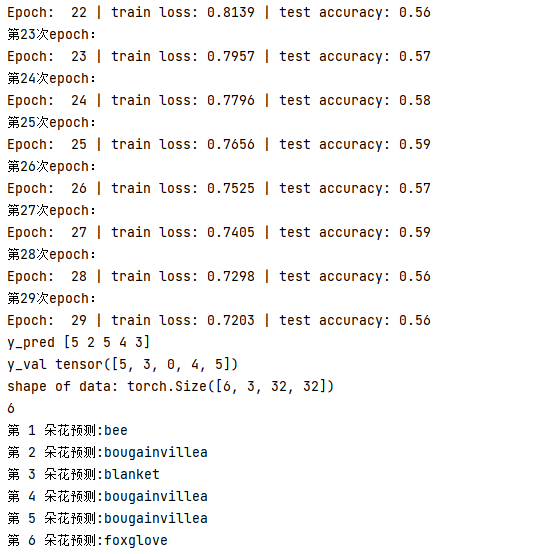
## 实验结果

*阐述模型测试与评估结果，包括各种模型在本地预测的效果截图与说明、ModelArts在线部署的预测效果截图与说明。*

**190110121-杨航：**

本地预测效果截图：

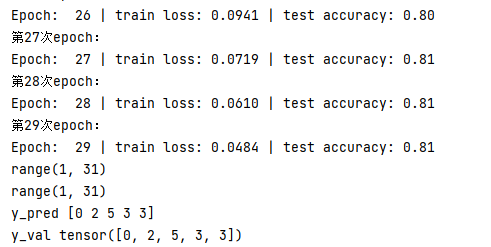
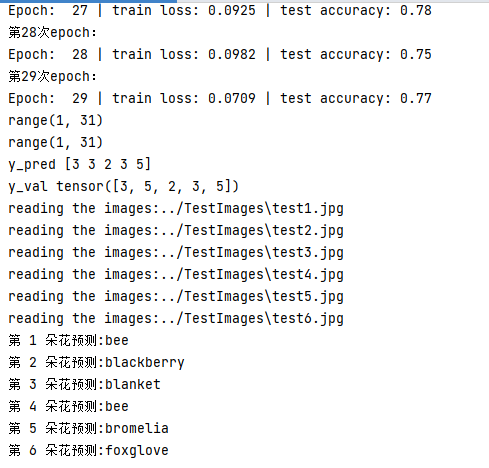
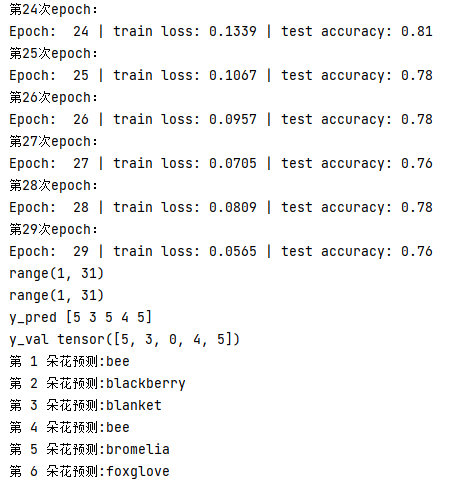
LeNet5：



最后的loss还较大，可能是学习率较小或者训练回合不够的原因，导致结果表现不佳。

VGG-16简化版：

更改seed值对数据集进行了三次不同的划分，得到了三个结果，分别如下。

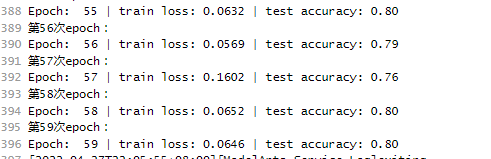


将几次测试结果取平均值得到该模型在训练集上的表现。

平均准确率：78%

ModelArts上训练结果：

（epoch为60，网络模型也比本地的更大一些）



虽然最后在训练集上的loss较小，但在测试集上的表现并不是很好。可能是由于网络层数不够，对特征的提取不够充分。

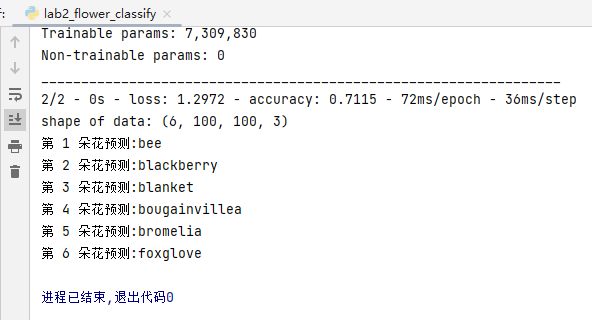
**190110130-向柳嫒：**

1. 本地预测效果截图（tensorflow）：

其中两次结果：

表格

低可信度描述已自动生成



文本

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成

平均准确率达到70%左右，loss值1.3左右

1. ModelArts上训练结果（epoch为40）：

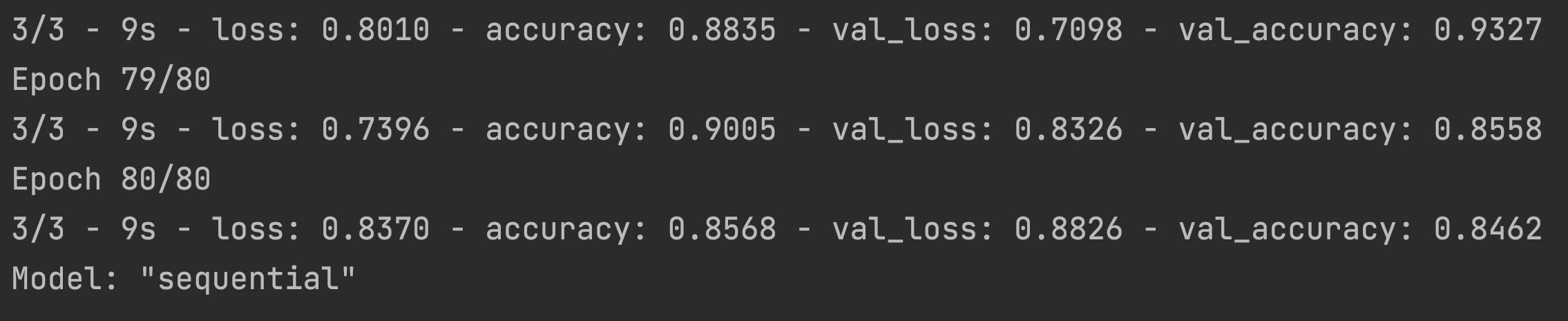
图形用户界面, 文本, 应用程序

描述已自动生成

最后的accuracy准确率有所提高，但是loss损失值表现不佳。

**190110115-谢俊安**

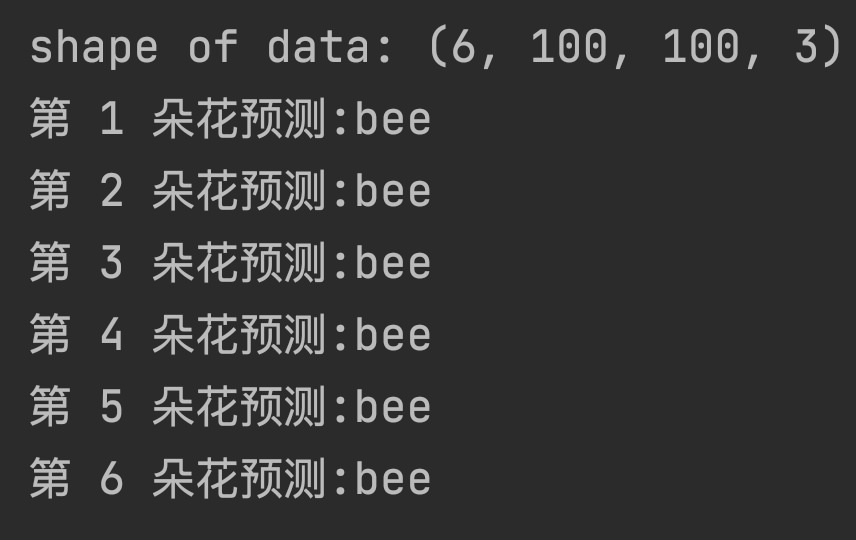
1. 本地预测效果：



****

预测结果显示错误的是bee和bromelia花种，初步估计可能是由于训练集太小导致的模型特征提取不够充分，预测效果不太明显。

1. ModelArts在线部署的预测结果：



可以看到效果非常差，原因是因为ModelArts平台不能访问外网的数据，因此无法在调用模型的时候加载imagenet权重，因此训练效果比本地差。

## 深度学习框架对比

1. 本地用到的几种深度学习框架的优缺点，在相同模型结构、超参数等情况下，不同框架在开发效率、训练速度及精度、推理速度及精度等方面的区别。

**190110115-谢俊安**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 框架 | 优点 | 缺点 | 开发效率 | 训练速度 |
| TensorFlow | 支持其他机器学习算法，支持强化学习，拥有更加完善的AI体系；  适合大规模部署，跨平台和嵌入式部署； | 版本分裂严重，不同版本几乎不可通用；  安装环境要考虑cuda/cudn版本；  不适合可变长度序列，缺少符号循环功能。 | 中等 | 较快 |
| PyTorch | 熟悉Python和常见的深度学习概念会非常容易上手；  安装不用考虑cuda/cudnn版本 | 做大规模分布式并行时需要定制性修改大量的代码； | 快 | 中等 |
| MindSpore | 支持昇腾芯片、GPU、CPU等异构平台；  昇腾+MindSpore组合下性能遥遥领先；  接口易用性能有保障；  在分布式集群和华为全栈下性能提升很大； | 社区比较小；  文档友好型差；  API接口混乱； | 慢 | 中等 |

2.云端不同框架的的区别。

**190110115-谢俊安**

|  |  |  |
| --- | --- | --- |
| 框架 | 在线部署 | 本地运行 |
| PyTorch | 训练时必须采用 state\_dict 的保存方式 | - |
| TensorFlow | 使用 model.save 接口保存模型 | - |
| MindSpore | 在 ModelArts 上部署更复杂一些；  只支持晟腾310芯片。 | - |

3.本地、云端同一框架之间的区别

**190110121-杨航**

|  |  |  |
| --- | --- | --- |
| 框架 | 优点 | 缺点 |
| Pytorch | 上手简单，代码简洁；定义网络方便；能够随时修改网络结构，方便调整参数，动态观察模型效果。 | 没有自带的可视化工具，动态编程导致模型效率较低，训练速度慢。 |
| ModelArts | 训练速度快，无需占用本地资源与算力，搭建好算法后能够对多个数据集进行多次训练；观察训练日志更加方便 | 对代码的修改不如本地方便，需要在本地修改后重新新建算法；可能会有版本号与本地不匹配的问题 |

**190110130-向柳嫒**

|  |  |  |
| --- | --- | --- |
| 框架 | 优点 | 缺点 |
| Tensorflow | 理解简单，网上学习资料很多，框架已经搭建好，绘制accuracy和loss图非常方便。 | 训练速度慢 |
| ModelArts | 无需搭建环境，只需上传数据集和代码即可运行，十分方便；训练日志清楚记录的历史训练，便于查找 | 每次修改完代码之后需要重新上传再重建、训练，稍显麻烦 |

**190110128-林先一 [3. MindSpore框架——本地&云端]**

|  |  |  |
| --- | --- | --- |
| 框架 | 优点 | 缺点 |
| MindSpore | 集成封装了很多功能库，一次安装成功后，可以使用其集成的许多工具完成具体功能，无需繁琐的去引入定义其他众多的包，整体代码简洁； “辩证地”提升了易用性和开发门槛； 定义网络结构时清晰明了； | 官方文档以及官方的问题解答教程较为敷衍，同时网路上的参考或者教程资源相对还是比较匮乏，出错以后只能到论坛中搜索或者提问； 本地安装无法一步到位，需要到官网按固定格式语法安装； 训练速度相较于其他两模型来说降低了很多； 框架整体设计的初衷是尽可能的封装为用户带来便利，但是过多或者过高层次的封装，降低了很多可用性，同时也带来了相对较多的学习成本（从其他框架迁移过来时） |
| MS on ModelArts | 在华为云使用MindSpore框架可以申请使用自家爸爸的昇腾Ai处理器资源池作为硬件支持； 平台训练速度快，存储资源容量大，摆脱本机的硬件的限制，算法配置成功后，可以对多个数据集进行多次训练；同时还有部署等功能 | 运行出错或者训练失败后需要修改代码再重新上传，代码修改繁琐； 操作流程繁琐； 不具备调试的功能； 数据读取及存储繁琐； 虽然不占用本地但是要钱的……. |

**190110115-谢俊安 [1. TensorFlow框架——本地&云端]**

|  |  |  |
| --- | --- | --- |
| 框架 | 优点 | 缺点 |
| **TensorFlow** | 每次训练时从外网加载预训练的模型，利用imagenet权重能得到较好的效果；  上手简单，网上开源资料多。 | 训练速度相较慢。 |
| ModelArts | 不需按照麻烦的cuda就可以用GPU跑程序，速度快；  不占内地的空间； | 要钱；  云端上传文件有100的上限限制，当上传的数据集较大的时候很不方便；  不可访问外网的数据，因而需要自行下载数据； |

# 总结



## 问题及解决方法

**190110121-杨航**

问题：在进行网络搭建的时候，对应该搭建一个什么结构的网络比较迷茫。不知道卷积层、全连接层和池化层应该各自放在什么位置能够有比较好的表现。

解决方法：去查询近年得到广泛使用的模型的搭建方法及各层之间的连接，通过在现有模型上增删网络层、修改参数以达到好的效果。

**190110130-向柳嫒**

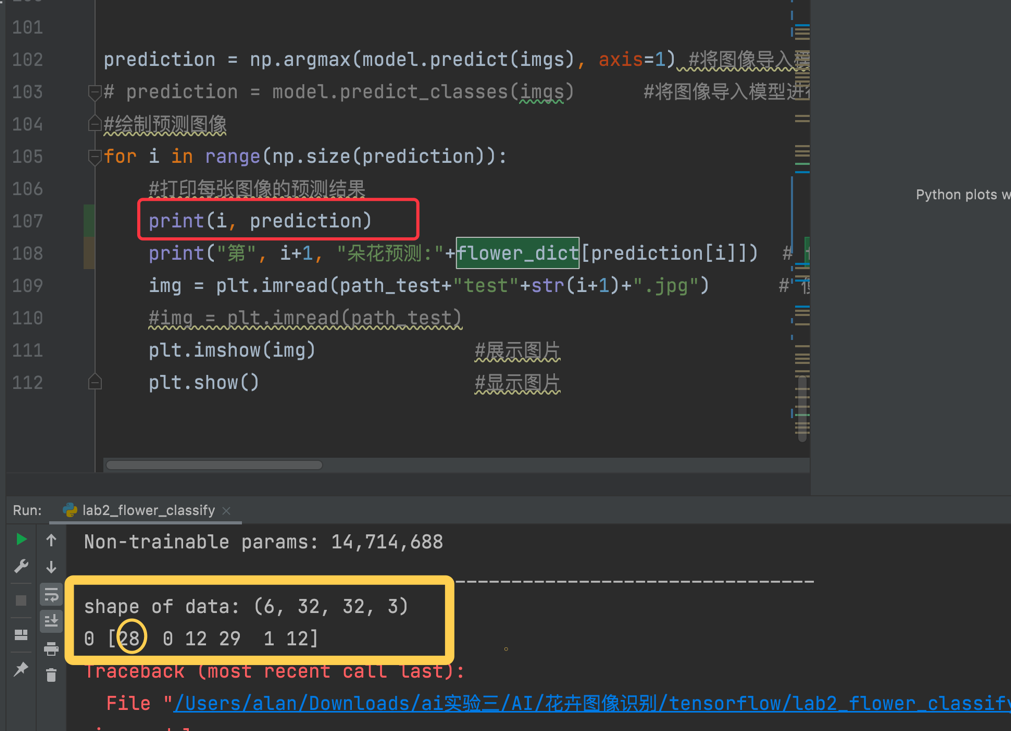
问题：首先安装环境就花了我不少时间。因为tensorflow安装错了版本，keras一直用不了。改了其他版本后，keras里面的包又用不了。上网查了很多资料，最后换了一种写法解决了。在实现模型结构时，对一些参数不知道怎么调整才能使结果更好，一长段时间怎么调都没效果。

解决方法：上网查找资料弄清CNN的原理，还有框架里各种函数里面参数的含义。之前一直没考虑修改model.compile的参数提高效果，后来尝试修改后发现其实损失函数的选择也会对结果有比较大的影响。

**190110115-谢俊安**

1. **对深度学习神经网络的原理不熟悉。**

进一步导致预测结果出现越界的异常（预测出本来不应该存在的标签，如下图）



解决办法：不要急于求成，先去理解基本原理和代码的逻辑思路，再去写代码。以上BUG是因为全链接层的书写错误，未写最后的输出特征导致的。

1. **对深度学习的不熟悉，急功近利地想要完成实验。**

分析及解决办法：

这是第一次正式地接触深度学习，最开始对于卷积层、全连接层、池化层等概念一直处于一知半解的状态，对于batchsize、lr、激活函数等对模型能力的影响更是完全不知道。并且对个人任务预估的目标太高（完成三个平台成熟模型的搭建和对比）以至于项目初期没有沉下心去理解模型的原理和内涵，只是盲目地调整参数查看训练结果。

此外遇到的困难还包括：刚入手深度学习，对于如何选型、成熟的优秀的模型是哪些、如何找等问题很茫然，没有框架的意识，导致前期浪费了很多时间。

最后发现这样像无头苍蝇一样乱撞乱碰运气的测试方法不仅杯水车薪，而且对于自身学习并没有好处。于是终于沉下心选择学习tensorflow平台上深度学习开发。

## 实验的启发、总结及建议

**190110121-杨航**

在这次实验之前已经尝试过很多次系统地学习深度学习的知识，但最多也只是停留在调取已经成熟的库、模型上，虽然学习了网络搭建的皮毛知识，但是逐层定义地搭建网络还是第一次。在之前调取像VGG16、VGG19等这样的网络模型的时候，由于网络内部结构已经确定，最多需要调整的参数只有batch大小和学习速率，但如果要有较好的训练效果，模型本身才是更重要的。要对模型有优化需要对内部有更深的了解，所以个人认为这一次实验是很有意义的。

**190110130-向柳嫒**

这次实验借机让我学习了卷积神经网络的原理和搭建，感慨神经网络的精妙。在调整代码时，遇到瓶颈要积极上网查找资料，思考原因，而不是盲目地一个个随便尝试。

**190110115-谢俊安**

前期盲目地乱套模型+调试参数浪费了大量的时间，但一直处于一知半解的状态。最后去阅读了所选用模型的源码，阅读了相关的论文。对模型本身有了一定的认知，再查看与理解其他项目的调用别的模型方式，才对整个迁移学习的过程有了一定的了解。

做完之后才觉得项目的难度其实也没有很高，需要操作的代码量也没有很大，但是最开始毫不了解的时候确实觉得非常难以下手。哪怕是调用模型的任务，因为网上其他项目的深度学习任务代码框架和老师给的不一样，所以想要真正理解并且完成此次任务还是需要去用心理解透彻之后，再回到框架自己认真书写。实际阅读并理解了一些VGG16的源码之后，感觉其实自己的工作量也并不小。

这门课是深度学习入门的第一堂课，本次实验之后学到了很多很多！

# 参考文献

[1] 理解Dropout

*t*[*https://blog.csdn.net/stdcoutzyx/article/details/49022443*](https://blog.csdn.net/stdcoutzyx/article/details/49022443)

[2] 深度学习中Batch size对训练效果的影响

[*https://blog.csdn.net/u010417185/article/details/79533539*](https://blog.csdn.net/u010417185/article/details/79533539)

[3] nn.Conv2d和其中的padding策略

[*https://blog.csdn.net/g11d111/article/details/82665265*](https://blog.csdn.net/g11d111/article/details/82665265)

**190110115-谢俊安：**

[1] VGG in Tensorflow (<https://www.cs.toronto.edu/~frossard/post/vgg16/>]

[2] Usage of VGG16 (<https://github.com/machrisaa/tensorflow-vgg>)

[3] Very Deep Convolutional NetWorks for Large-Scale Image Recognition (<https://kns.cnki.net/kns8/Detail?sfield=fn&QueryID=4&CurRec=1&recid=&FileName=XTYY202109045&DbName=CJFDLAST2021&DbCode=CJFD&yx=&pr=&URLID=>)