

《OpenEuler应用实践》

综合设计报告

题 目： Tensorflow猫狗识别

学 院： 计算机与信息安全学院

专 业： 软件工程

姓 名： 车立钧

学 号： 2100301708

指导教师： 陈俊彦

2023年2月17日

**摘要**

应用背景: 本实验使用Python和TensorFlow框架，实现了一个猫狗图像分类器。首先，我们使用爬虫收集猫狗图像的数据集，并使用Keras提供的ImageDataGenerator类对数据集进行了预处理和增强。接着，我们构建了一个包含卷积层、池化层和全连接层的深度神经网络模型，并使用Adam优化器进行训练。最后，机器模型的评测与应用，其主要功能是基于数张猫和狗的图片学习模型，对图像中动物种类进行推断。

关键字：爬虫;OpenEuler;TensorFlow；Keras;CNN模型;

目录

[摘要 1](#_Toc29133)

[1、 项目背景、功能需求分析 2](#_Toc18762)

[2、 选题使用相关技术及环境 3](#_Toc30017)

[3、 概要设计 3](#_Toc19333)

[4、 详细设计与实现 4](#_Toc9714)

[5、 系统测试 7](#_Toc31725)

[6、 总结 9](#_Toc4969)

[参考文献 10](#_Toc14921)

## 项目背景、功能需求分析

**项目背景:**

TensorFlow是一个采用数据流图，用于数值计算的开源软件库。节点在图中表示数学操作，图中的线则表示在节点间像话联系的多维数据组，即张量。它灵活的架构让你可以在多种平台上展开计算，例如台式计算机中的一个或者多个CpU（或者GPU），服务器，移动设备等等。TensorFlow最初由Google大脑小组（隶属于Google机器智能研究机构）的研究员和工程师们开发出来，用于机器学习和深度神经网络方面的研究，但是这个系统的通用性使其也可广泛的用于其他计算领域。

**功能需求分析:**

猫狗图像识别是一项基于图像处理和人工智能技术的任务，旨在通过计算机自动识别出输入图像中的猫或狗。该任务的实现有着广泛的应用场景，如智能家居、自动驾驶、医学图像分析等。

数据集的获取和预处理：本项目需要获取足够的猫狗图像数据集，并对数据进行预处理和增强，以提高模型的性能和泛化能力。

模型的选择和训练：本项目需要选择合适的深度学习模型结构，并进行参数优化和训练，以提高模型的准确率和稳定性。

图像识别的准确率评估：本项目需要对训练好的模型进行准确率评估，并提供相应的评估报告和分析结果。

图像分类的实现：本项目需要实现一个基于训练好的深度学习模型的猫狗图像分类器，能够对输入的猫狗图像进行分类并输出分类结果。

## 选题使用相关技术及环境

**相关技术:**

re爬虫;

机器学习;

**环境:**

openEuler 20.03

python3.7.15

python3.7版本的anaconda

TensorFlow 2.10.0;

Keras 2.10.0;

## 概要设计

本次实验项目的目标在于通过对机器深度学习的训练来实现把猫和狗进行识别与分类。

首先，对猫和狗进行分类是一个二元问题，这一点上对图片样本中动物的识别应该不是猫就是狗，而不会出现其他动物，或者出现否定的情况，这一点上模型可能会对其它除猫和狗类别的动物造成误判。

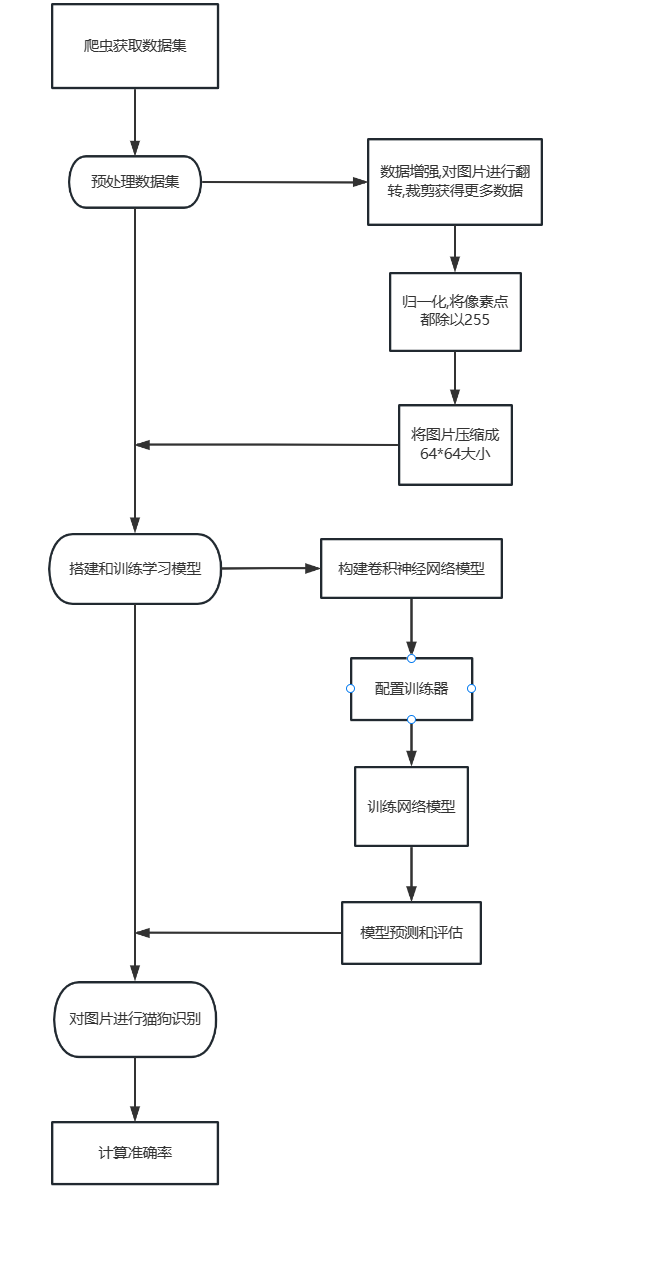
其次，我们需要通过网络爬虫搜寻一定数量的猫与狗的图片，要求猫和狗的图片数量相当，并将数据集分为训练集、验证集。用于机器深度学习。

随后，获取图片后，基于各类图片大小、质量参差不齐的原因，我们还需要对图片进行进一步的处理，最优先的是在于图片的大小需要统一，尽管在图片样本的数量尚未达到某一量级的时候，手工编辑压缩图片大小是可行，但我们仍然需要考虑到少量的图片样本用于训练时所会导致的模型预测准确率过低的情况发生，这就意味着图片样本需要达到某一量级，那么就需要进行数据增强,对图片进行翻转,平移,旋转一系列操作后得到更大的数据量，接着使用Python语言提供工具对图像转换为相同的大小、对图像进行归一化，随后还需要考虑到在图片样本中，需要识别的对象并非总是居于图像的中间的，而CNN模型的配置需要恰到好处，并根据需要进行模型调整。

## 详细设计与实现

**包括流程、功能描述和核心代码**

**流程:**



首先要获取用于模型训练的数据集，利用爬虫爬取https://www.vcg.com/的猫狗图片，爬虫使用re正则表达式获取图片链接,使用get请求下载图片并保存到本地文件夹,图片集简单地分为了猫和狗两类，且猫的样本数量和狗的样本数量基本相同，还需要将其分类成训练集的猫和狗、测试集的猫和狗以确保模型训练的顺利进行。设计为猫狗训练集分别有1433张图片，猫狗测试集分别有748张图片。

接着完成训练集与测试集的分类后需要进行数据预处理，即将图像大小统一，并分批打包成便于TensorFlow理解的数据形式。由于训练数据过少,为了防止过拟合的情况出现,将训练集的图片进行数据增强,对每张图片进行变形,旋转角度30度,水平和竖直方向随机移动的程度设置20%,剪切变换的程度设置20%,随机的放大20%,接着进行水平翻转,像素填充选择aabbccdd|abcd|aabbccdd模式。图片集均为彩色，即具备RGB颜色通道，像素点一般会被用3个0-255之间的数字描述，故使用Keras提供的ImageDataGenerator将像素编码除以255缩小为0-1之间的数字，方便神经网络中神经元的激活函数的计算，再使用Keras实例对象的flow\_from\_directory方法来对数据集中的图像来大小裁定为150\*150像素以及每100图片信息为一批，即完成了针对数据的预处理。

然后需要搭建模型的神经网络，在此处我采用第一层卷积层、第一层池化层、第二层卷积层、第二层池化层、第三层卷积层、第三层池化层、数据准备层、全连接层、Dropout层、全连接层的方式构建，通过tensorflow.keras.layers.Conv2D（）函数构建二维卷积层，通过tensorflow.keras.layers.MaxPooling2D（）函数构建二维池化层，通过tensorflow.keras.layers.Flatten（）与tensorflow.keras.layers.Dense（）来构建全连接层，其中第一层卷积层,由于图像样本大小为64\*64像素大小，且颜色为彩色图,即3个颜色通道，因此设置输入层为(64,64,3),同时设置卷积层过滤维度数，及卷积核数为64，卷积核大小为3\*3,第一层池化层，采取2\*2的大小取特征，用于强化前一层输入数据的最大特征，即将用于描述的参数中的最大值取出。随后第二层卷积层接收上一层输入数据（29,29,64）继续进行处理，第二层池化层、第三层卷积层、第三层池化层同理，直到全连接层，首先将维度转化为一维，即将数据展平，将其连接至后一具备512个神经元的全连接层,由于最后拥有确切的结果，且模型处理问题为二分类问题，所以最后的全连接层仅具备一个神经元用于作为输出层结果。其中，卷积层与第一层全连接层所使用的激活函数为”relu”，该激活函数能够更加有效率的梯度下降以及反向传播，避免了梯度爆炸和梯度消失问题，同时没有了其他复杂激活函数中诸如指数函数的影响，简化计算过程，使活跃度的分散更加分散让神经网络整体计算成本下降。dropout 能够避免过拟合，我们往往会在全连接层这类参数比较多的层中使用dropout；在训练包含dropout层的神经网络中，每个批次的训练数据都是随机选择，实质是训练了多个子神经网络，因为在不同的子网络中随机忽略的权重的位置不同，最后在测试的过程中，将这些小的子网络组合起来，类似一种投票的机制来作预测，有点类似于集成学习的感觉。最后的全连接层使用的激活函数为sigmoid用以将变量映射到0-1之间表示判断图像中动物类别的概率，与上述二元交叉熵共同作用。神经网络搭建完成就此完成。

接下来，在设置模型的训练方法时，采用RMSprop优化算法作为优化器，在参数空间更为平缓的方向，会取得更大的进步并且能够使得陡峭的方向变得平缓，从而加快训练速度，使模型更快的达到预期效果。学习率设置为0.0001,一般设置为0.1-0.001,设置0.0001是为了防止出现损失值爆炸,副作用是易过拟合,收敛速度慢。因此同时采用数据增强和Dropout层进行提升过拟合效果。采用binary\_crossentropy二元交叉熵作为损失函数来处理二元问题，通过对每个样本的损失进行求和或者求平均，可以得到整个数据集的损失值，作为模型的性能评估指标。二元交叉熵损失函数的优点是具有较好的数学性质和收敛速度。在训练过程中，模型在计算梯度时可以更快地收敛到最优解。因此，它在深度学习的二分类问题中被广泛使用。计算某一批样本中动物为猫或者为狗的概率的多少。采用accuracy作为衡量指标，这样来解决一个判别问题才有最终的结果。

随后，即开始对模型进行拟合，使用model的fit方法进行，记录下训练中的准确率与损失率用于模型的评估。Python的matplotlib库，绘制训练准确率和损失率的对比图，依此评估模型是否完成。

最后对验证集的猫狗各一百张图片进行验证并计算分类正确率

**功能描述:**

实验项目能自动爬取图片到数据集文件夹，以文件夹名称为动物种类，将样本按2:1分配训练集与测试集；

将原始图像数据进行预处理，包括裁剪、缩放、增强等操作；

使用TensorFlow搭建深度学习模型，采用卷积神经网络（CNN）等算法，对经过预处理的图像数据进行训练；

对训练得到的模型进行评估，计算模型的准确率、损失率指标；

读取已经训练完成的模型用于猫与狗的类别判定。

**核心代码:**

model = tf.keras.models.Sequential([

#如果训练慢，可以把数据设置的更小一些

tf. keras. layers. Conv2D(32,(3, 3),activation='relu',input\_shape=(64,64,3)),

tf. keras. layers. MaxPooling2D(2,2),

tf. keras. layers. Conv2D(64,(3, 3),activation='relu' ),

tf. keras. layers. MaxPooling2D(2, 2),

tf. keras. layers. Conv2D(128,(3, 3),activation='relu' ),

tf. keras. layers. MaxPooling2D(2, 2),

#为全连接层准备

tf. keras. layers. Flatten(),

tf. keras. layers. Dense(512,activation='relu' ),

tf. keras. layers. Dropout(0.5),

#二分类sigmoid就够了

tf. keras. layers. Dense(1,activation='sigmoid' ),

])

# 系统测试

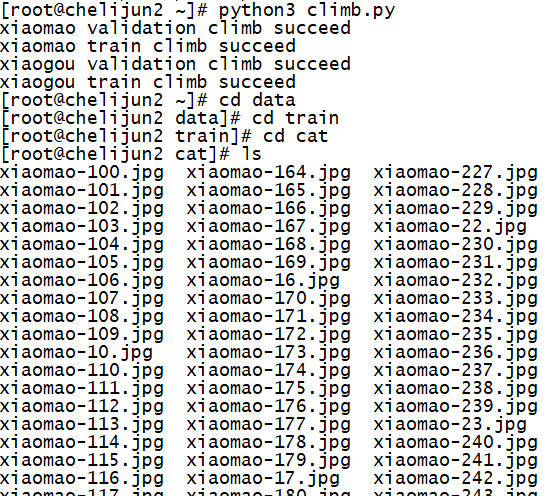


图5.1 爬取数据

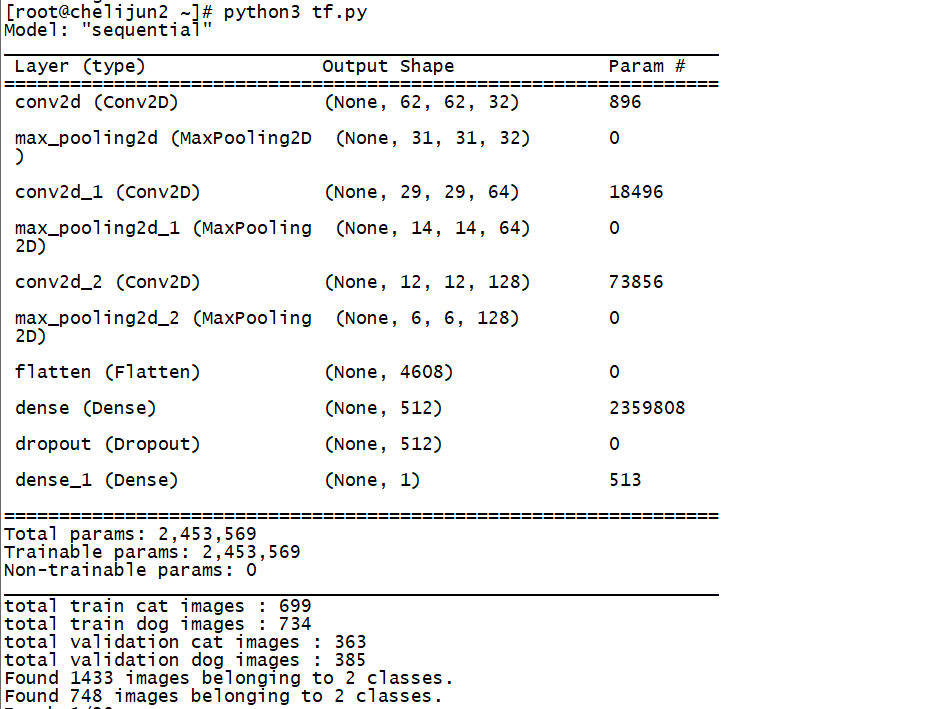


图5.2 卷积神经网络模型和读取数据

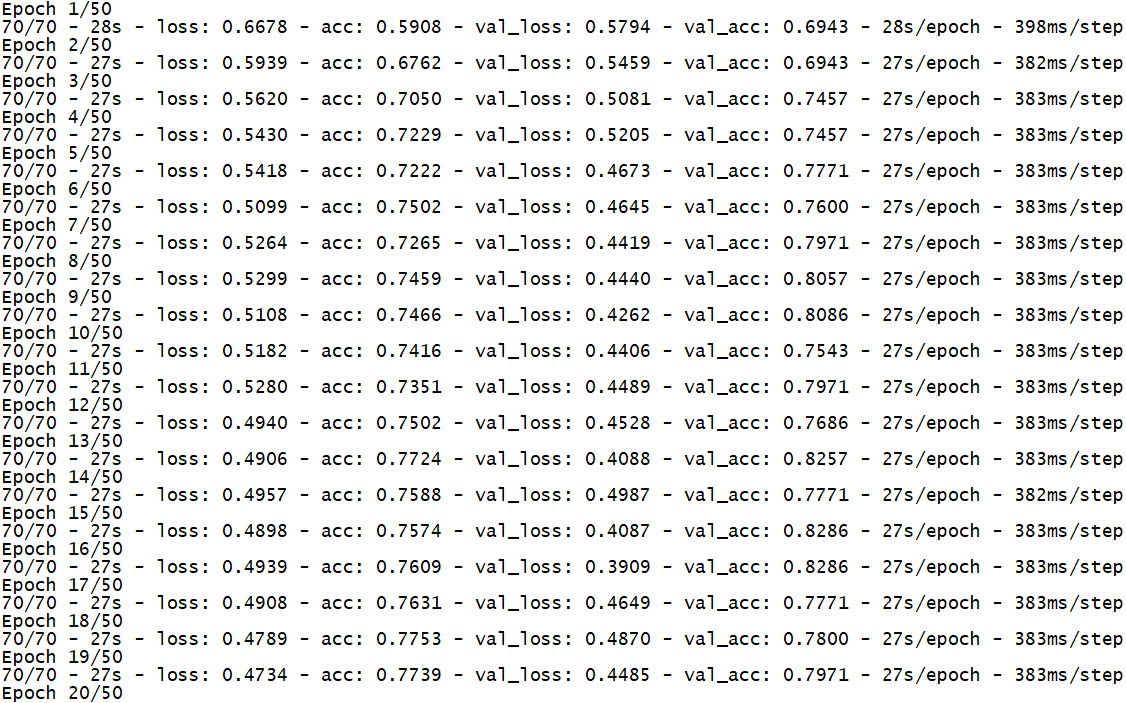


图5.3 计算准确率和缺失率

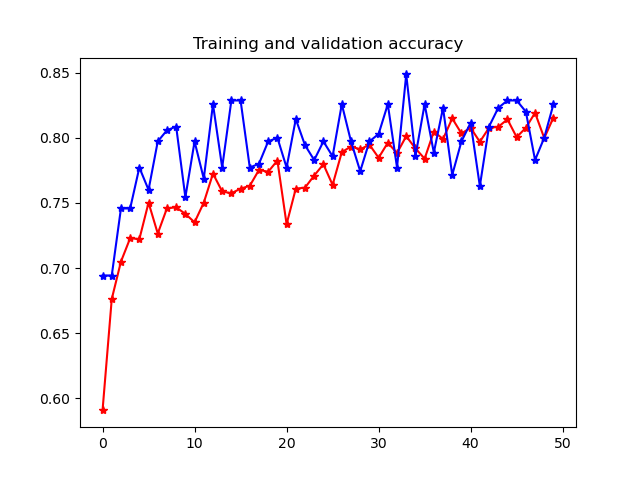


图5.4 模型正确率(红色表示训练正确率、蓝色表示预测正确率)

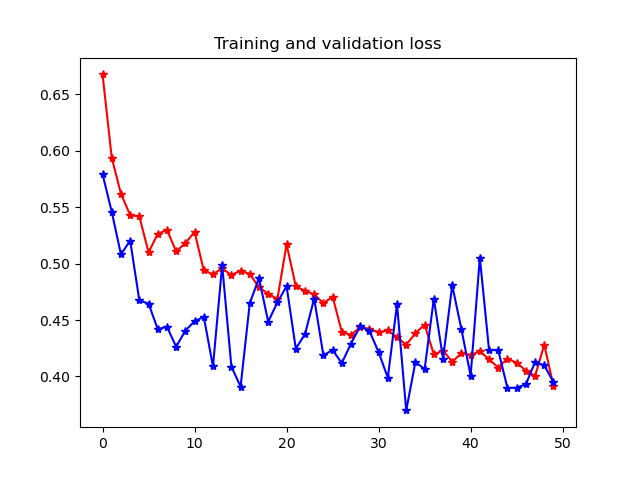


图5.5 损失率(红色表示训练损失率、蓝色表示预测损失率)

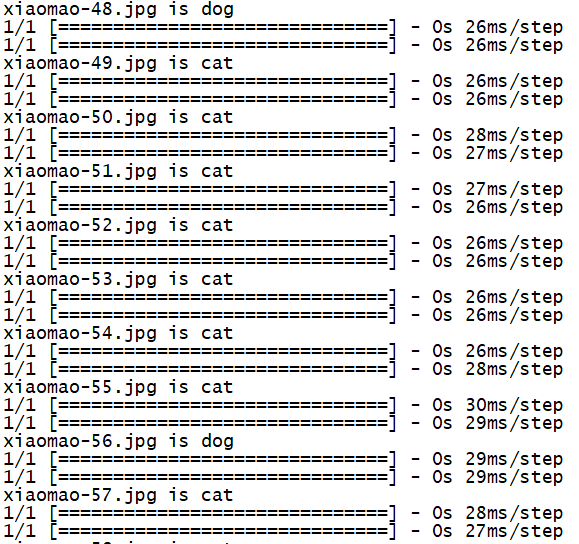
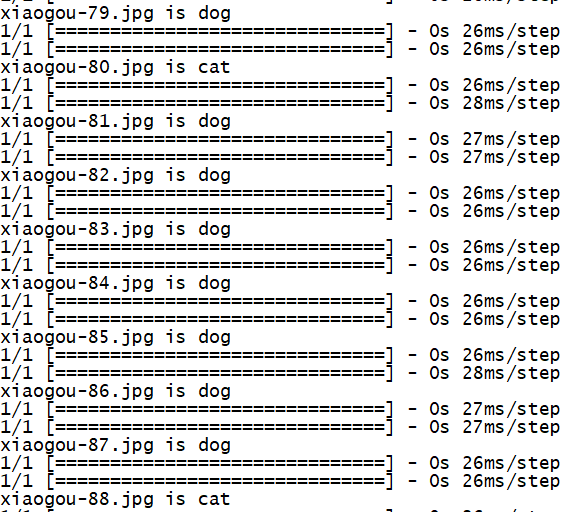


图5.5 部分分类结果

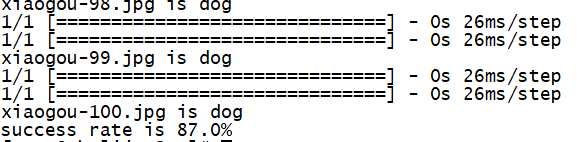


图5.6 分类正确率

# 总结

系统特点和个人开发过程中遇到的问题以及收获总结。

**系统特点:**

OpenEuler系统具有多层安全保障，包括应用层安全、内核安全、硬件安全等多方面的保障。经过了严格的测试和优化，保证了系统的稳定性和可靠性。兼容多种软件和应用，支持多种开发语言和框架。

爬虫的主要特点是自动化、高效性、可扩展性、数据结构化和可定制性，这使得它成为了很多数据获取、处理和分析工作中的不可或缺的工具。同时，由于爬虫可能会对网站造成不良影响，如给网站带来额外的负载，或者侵犯网站的知识产权等问题，因此在使用爬虫时需要遵守相关的法律法规和道德准则。

TensorFlow使用图来表示计算模型，图中的节点表示运算，边表示数据流，将各个运算以及它们之间的数据依赖关系组成一张图。在运行时，TensorFlow会将图优化并分解成多个计算单元，实现并行计算，提高了运算效率。TensorFlow可以轻松扩展和定制，用户可以通过实现自己的运算或者自定义模型层来扩展TensorFlow的功能，使得TensorFlow可以应对各种复杂的计算任务。TensorFlow是一个功能强大、灵活性高、可扩展性强、易于使用的深度学习框架，是众多深度学习从业者的首选工具之一。

**遇到的问题:**

在电脑上运行时,epochs设置为100,一个epoch , 表示：所有的数据送入网络中， 完成了一次前向计算+反向传播的过程，迭代一百次就会得到一百次的准确率和缺失率的比较，而云服务器性能不如电脑cpu导致模型训练运行过慢,只能通过减少迭代次数和将图片压缩得更小，迭代次数修改成50，图片压缩成64\*64大小，才能减少时间，当然也会降低模型的准确性。

结果过拟合现象，过拟合指的是模型在训练集上表现良好，但在新数据上的泛化能力却很差的现象。通过对原始数据进行随机的旋转、平移、缩放、翻转等变换，从而生成更多的训练数据，避免模型过度拟合。通过Dropout层在训练过程中随机的让一些神经元失活，从而降低模型的复杂度，避免过拟合。我使用了数据准确和dropout层。还可以通过添加L1或L2正则化项限制模型参数的大小和迁移学习。

**收获总结:**

学习爬虫不仅让我了解了互联网技术和原理，还帮助我获取了大量的网页数据，为我进行数据分析和挖掘提供了丰富的资源。学习爬虫也让我意识到了数据隐私和安全的重要性。

通过学习TensorFlow，我深入了解了神经网络的基本原理，包括前向传播、反向传播、损失函数、优化器等概念，掌握了TensorFlow的常用API和工具，包括张量、变量、占位符、会话、模型保存和加载等，学会了如何搭建和训练深度学习模型，包括搭建神经网络结构、选择合适的损失函数和优化器、使用数据预处理技术等，TensorFlow在不同领域的应用，包括图像处理、自然语言处理、推荐系统、游戏AI等，这些应用对我了解深度学习的实际应用有很大帮助。

机器学习需要大量的数据作为训练集，训练集中的数据应该尽可能地覆盖真实情况，这样模型才能够学习到更加精准的知识。同时，训练集的质量也非常重要，因为质量不好的训练集会使模型产生偏差，影响其预测效果。其次，机器学习算法的选择非常重要，不同的算法适用于不同的问题，因此需要对不同算法进行比较和评估。一些经典的机器学习算法包括线性回归、逻辑回归、决策树、支持向量机、神经网络等。另外，机器学习还需要特征提取和特征选择。特征提取是将原始数据转换为更好的特征表示，从而提高分类的准确率。特征选择是从大量特征中筛选出对分类有用的特征，减少训练的复杂度，避免过拟合。

# 参考文献

1. [https://www.bilibili.com/video/BV1FW4y1b7WM/?p=1](https://www.bilibili.com/video/BV1FW4y1b7WM/?p=1&vd_source)，2023,01

（2）https://blog.csdn.net/weixin\_43073921/article/details/118014982，2023.01

（3）https://www.bilibili.com/video/BV1rv4y1K7yV，2023.01

（4）https://blog.csdn.net/guo\_qingxia/article/details/113979135，2023.01

（5）https://blog.csdn.net/weixin\_43398590/article/details/105173936，2023.02

（6）https://blog.csdn.net/weixin\_43398590/article/details/105174367，2023.02

（7）https://mp.weixin.qq.com/s?\_\_biz=MzIwNDY0MjYzOA==&mid=2247501546&idx=2&

sn=0168d0512d188731041cbeb133e87cea，2023.02