广州華商學院
GUANGZHOU HUASHANG COLLEGE

《专业综合实践》课程考核

题　　目：	基于深度学习的图像识别系统
学　　院：	**人工智能学院**
专　　业：	数据科学与大数据技术
年级班别：	21 本大数据 1 班
姓　　名：	李桄俊
学　　号：	421470123
指导教师：	刘盛

提交日期： 2024 年 11 月

基于深度学习的图像识别系统：牛鸡猫分类项目

摘要

本文详细介绍了一个基于深度学习的图像识别系统，用于牛、鸡、猫的分类任务。首先阐述了项目背景与目的，明确其在人工智能领域及农业养殖、动物保护等方面的重要性，并指出项目旨在培养学生相关技能。接着依次说明了数据预处理过程，包括数据集描述、图像尺寸调整、归一化处理、数据增强和数据集划分；模型构建部分涵盖模型选择、架构设计、激活函数、损失函数和优化器选择；模型评估方面介绍了准确率、召回率、F1 分数等评估指标及交叉验证、混淆矩阵等评估方法；结果分析对比了不同模型性能，讨论了模型在特定类别上的表现差异；最后针对模型优化提出了调整参数、尝试不同网络结构或正则化技术等措施，并提供了 github 项目链接及仓库目录结构截图。

关键词

深度学习；图像识别；数据预处理；模型构建；模型评估；模型优化

目录

1 项目背景与目的

1.1 背景

图像识别于人工智能领域意义非凡，其能让计算机解读图像内容，在安防监控、医疗影像诊断、自动驾驶等多领域均有广泛应用。鸡、牛、猫分类在农业养殖中可助力智能管理，如监测动物数量与健康、防范外来动物；于动物保护领域，有助于掌握动物分布与生存状况。

1.2 目的

培养学生图像处理与机器学习的基础知识，训练其运用深度学习模型解决鸡、牛、猫分类实际问题的能力，提升数据预处理、模型构建、评估与优化等综合技能，以增强学生在相关领域的实践与创新能力，使其能更好地应对图像分类挑战并为相关应用提供技术支持。

2 数据预处理

2.1 数据集描述

本项目中使用的图像数据集为牛、鸡和猫的分类任务而收集。数据集来源于公开的图像资源，每个类别包含数量均衡的图像样本，以确保模型训练时的公平性和有效性。图像样本涵盖了不同的环境背景和光照条件，增加了模型训练的挑战性，同时也提高了模型在实际应用中的泛化能力。

2.2 预处理步骤

（1）图像尺寸调整：

在代码中通过 transforms.Resize((256, 256))操作，将数据集中的所有图像统一调整为 256×256 的尺寸，这样做是为了让图像能够适应后续模型输入的要求，确保模型在处理图像数据时输入数据的维度一致性，便于模型进行有效的特征提取和分类计算。

（2）归一化处理：

使用了 transforms.Normalize(mean=(0.5, 0.5, 0.5), std=(0.5, 0.5, 0.5))语句来实现归一化处理。其目的是将图像的像素值按照设定的均值和标准差进行缩放，使得像素值大致分布在 0 到 1 的范围内（实际并非严格在 0 到 1，但经过这种归一化能使数据在模型训练时更利于收敛等操作），有助于提高模型训练的效率和稳定性，让模型能够更好地学习图像的特征。

（3）数据增强：

部分数据增强操作在代码中有体现，例如通过 transforms.RandomHorizontalFlip(p=0.3)和 transforms.RandomVerticalFlip(p=0.3)分别以一定的概率（这里是 0.3）对图像进行水平翻转和垂直翻转操作。这些数据增强手段可以在一定程度上增加数据的多样性，使得模型在训练过程中能够接触到更多不同形态的图像数据，进而提高模型的泛化能力，使其在面对实际应用中的各种图像情况时能够更准确地进行分类。

（4）划分数据集：

首先，利用 MyDataset 类来加载数据，在加载数据后，通过以下方式划分数据集：

先确定训练集和验证集的划分比例，如代码中通过 train_size = int(len(train_dataset)＊0.8)计算出训练集的大致大小，即占总数据集的 80%，而 validate_size = len(train_dataset) - train_size 则得到验证集的大小。

接着使用 torch.utils.data.random_split(train_dataset, [train_size, validate_size])函数将加载好的数据集按照上述计算的大小划分为训练集和验证集。

另外，还单独定义了测试数据集 test_dataset，它与训练数据集采用相同的加载和预处理方式，但在后续使用中主要用于对训练好的模型进行最终的性能测试评估。

3 模型构建

3.1 模型选择

CNN 是一种专门用于处理具有网格结构数据的深度学习模型，其理论基础在于通过卷积层自动提取图像中的局部特征，利用池化层对特征进行下采样以减少数据维度并保留关键特征，最后通过全连接层对提取到的特征进行整合和分类决策。

3.2 模型架构

（1）卷积层：

模型中包含了多个卷积层结构，具体如下：

conv1：

输入通道数（in_channels）为 3，对应图像的 RGB 三个颜色通道。

输出通道数（out_channels）为 16，意味着经过该卷积层后会生成 16 个不同的特征图。

卷积核大小（kernel_size）为 3×3，通过在图像上滑动该卷积核进行卷积操作来提取局部特征。

步长（stride）为 2，控制卷积核在图像上滑动的步幅，步长为 2 使得每次滑动能覆盖更大的区域，从而在一定程度上降低数据维度。

该卷积层后还连接了批归一化层（nn.BatchNorm2d (16)）用于对卷积层输出的特征进行归一化处理，加快模型训练速度并提高模型稳定性，接着是激活函数 ReLU（nn.ReLU ()）用于引入非线性因素，使模型能够学习到更复杂的图像特征，最后是最大池化层（nn.MaxPool2d (kernel_size=2)），池化核大小为 2×2，通过取局部区域的最大值进行下采样操作，进一步降低数据维度。

conv2：

输入通道数为 16（即上一层 conv1 的输出通道数）。

输出通道数为 32。

卷积核大小、步长等参数设置与 conv1 类似，同样在卷积操作后依次连接批归一化层、ReLU 激活函数和最大池化层，以不断提取和优化图像特征。

conv3：

输入通道数为 32（来自 conv2 的输出）。

输出通道数为 64。

具有与前两个卷积层相似的卷积核大小、步长等参数设置，且后续同样连接批归一化层、ReLU 激活函数和最大池化层，持续对图像特征进行深度提取和处理。

（2）全连接层：

fc1：在经过一系列卷积和池化操作后，数据维度已经大幅降低，此时通过 fc1 = nn.Linear(3 * 3 * 64, 64)将特征图展平后的一维向量（长度为 3×3×64）映射到一个 64 维的向量空间，进一步对特征进行整合和变换。

fc2：接着通过 fc2 = nn.Linear(64, 10)将 64 维向量再映射到 10 维向量空间，继续对特征进行处理。

out：最后通过 out = nn.Linear(10, N)将 10 维向量映射到最终的分类类别数 N 对应的向量空间，这里 N 在原代码中是用于猫狗分类的类别数 2，若用于鸡、牛、猫分类则 N 应为 3。

3.3 激活函数、损失函数和优化器的选择

（1）激活函数：

在模型的多个层中均选用了 ReLU（Rectified Linear Unit）作为激活函数，如在每个卷积层之后以及全连接层 fc1 和 fc2 之后都使用了 ReLU 激活函数。ReLU 函数定义为，它的优点在于计算简单、收敛速度快，并且能够有效引入非线性因素，使得模型可以学习到复杂的图像特征关系，避免模型只是简单地进行线性变换而无法处理复杂的分类任务。

（2）损失函数：

选用了交叉熵损失函数（nn.CrossEntropyLoss ()）。交叉熵损失函数常用于多分类任务，它衡量了模型预测输出与真实标签之间的差异程度。在训练过程中，模型通过不断调整参数以最小化该损失函数的值，从而使得模型的预测结果更加接近真实情况。具体计算时，它会根据模型输出的概率分布和真实标签计算出一个损失值，模型依据这个损失值进行反向传播来更新参数。

（3）优化器：

采用了 Adam 优化器（optim.Adam (model.parameters (), lr=0.0008)）。Adam 优化器结合了自适应学习率的优点，它能够根据每个参数的梯度信息动态地调整学习率，使得模型在训练过程中能够更快地收敛到一个较好的参数值。其中设置的学习率（lr）为 0.0008，学习率控制着模型参数更新的步长，合适的学习率对于模型能否有效训练以及训练速度至关重要。

4 模型评估

4.1 评估指标

（1）准确率（Accuracy）：

准确率是指模型预测正确的样本数占总样本数的比例。它是最直观的评估模型性能的指标之一，计算公式为：准确率 = 预测正确的样本数 / 总样本数。例如，在鸡、牛、猫分类任务中，如果模型总共对 100 个样本进行了分类预测，其中有 80 个样本被正确分类，那么该模型的准确率就是 80%。准确率能够大致反映模型整体的分类效果，但在样本类别不均衡的情况下，可能会存在一定的局限性，比如某一类样本数量极少，即使模型对这类样本全部预测错误，准确率可能依然看起来较高。

（2）召回率（Recall）：

召回率也称为查全率，它主要衡量的是模型能够正确识别出某一类样本的能力。对于某一类样本，召回率的计算公式为：召回率 = 预测正确的该类样本数 / 实际该类样本数。比如在鸡、牛、猫分类中，对于鸡这一类样本，如果实际有 50 只鸡的样本，模型正确预测出了 40 只鸡，那么鸡这一类的召回率就是 40 / 50 = 80%。召回率关注的是模型是否能把某一类样本尽可能多地找出来，在一些对特

定类别样本检测要求较高的场景下，如在动物保护中确保能检测出所有濒危动物（假设鸡、牛、猫中有濒危动物类别），召回率就显得尤为重要。

（3）F1 分数（F1 Score）：

F1 分数是综合考虑了准确率和召回率的一个评估指标，它是准确率和召回率的调和平均数，计算公式为：F1 分数 = 2 * （准确率 * 召回率） / （准确率 + 召回率）。F1 分数能够平衡准确率和召回率之间的关系，避免单独使用准确率或召回率时可能出现的片面评估情况。当准确率和召回率都较高时，F1 分数也会较高，说明模型在分类任务上的综合性能较好。例如在鸡、牛、猫分类中，如果某一时刻模型对鸡类样本的准确率为 80%，召回率为 70%，那么鸡类样本的 F1 分数 = 2 * （0.8 * 0.7） / （0.8 + 0.7）≈ 74.7%。

4.2 评估方法

（1）交叉验证（Cross-Validation）：

交叉验证是一种确保模型泛化能力的有效评估方法。其基本思想是将数据集划分为多个不同的子集，通常采用 k 折交叉验证的方式，比如常见的 5 折交叉验证或 10 折交叉验证。以 k 折交叉验证为例，具体操作如下：

首先将整个数据集平均分成 k 个子集。

然后依次选取其中一个子集作为验证集，其余 k - 1 个子集作为训练集，进行 k 次这样的训练和验证过程。

在每一次训练过程中，使用训练集训练模型，然后用验证集对训练好的模型进行评估，得到 k 个不同的评估结果（如准确率、损失值等）。

最后综合这 k 个评估结果来判断模型的泛化能力，比如可以取这 k 个评估结果的平均值作为最终的评估指标。通过这种方式，模型能够在不同的数据子集组合上进行训练和验证，充分利用了数据集的信息，有效避免了因数据集划分方式单一而导致对模型泛化能力评估不准确的问题，从而更准确地衡量模型在未见过的数据上的表现能力。

（2）混淆矩阵（Confusion Matrix）：

混淆矩阵是一种直观展示分类结果的工具。在鸡、牛、猫分类任务中，它是一个 3×3 的矩阵（假设三类分类任务），行表示预测的类别，列表示实际的类别。具体如下：

预测类别	**实际鸡类**	**实际牛类**	**实际猫类**
预测鸡类	正确预测鸡类的数量	把牛误判为鸡的数量	把猫误判为鸡的数量
预测牛类	把鸡误判为牛的数量	正确预测牛类的数量	把猫误判为牛的数量
预测猫类	把鸡误判为猫的数量	把牛误判为猫的数量	正确预测猫类的数量

通过观察混淆矩阵，我们可以清晰地看到模型在不同类别之间的误判情况，比如哪些类别之间容易混淆，哪一类的预测准确率相对较高等等。这有助于我们深入分析模型的分类性能，针对出现的问题对模型进行进一步的优化和调整，例如，

如果发现鸡类和猫类之间误判较多，可能就需要重新审视模型对这两类动物特征的提取和区分能力，进而采取相应的改进措施。

5 结果分析与优化

5.1 结果分析

（1）对比不同模型的性能：

若仅基于当前代码中的卷积神经网络（CNN）模型进行分析，与其他可能的模型（如简单的多层感知机模型）相比，CNN 具有明显优势。多层感知机模型在处理图像数据时，由于没有考虑图像的空间结构信息，难以有效地提取图像的局部特征，通常在图像分类任务中表现较差，准确率较低且泛化能力不足。而 CNN 模型通过卷积层和池化层能够自动学习图像的局部特征并逐步聚合高层语义信息，在处理图像分类任务时能够更好地捕捉图像中的关键信息，具有较高的准确率和较好的泛化能力。然而，与一些更复杂的先进模型（如基于 Transformer 架构的视觉模型）相比，CNN 可能在处理长距离依赖关系和复杂场景理解方面稍显逊色。例如在一些图像中鸡、牛、猫之间存在复杂的遮挡关系或背景干扰严重时，Transformer 架构的模型可能通过其自注意力机制更好地整合全局信息进行分类判断，而 CNN 可能需要更复杂的网络设计或数据预处理来应对。

（2）讨论模型在特定类别上的表现差异：

在鸡、牛、猫分类任务中，模型可能在不同类别上表现出一定的差异。例如，若数据集中鸡的图像在外观、颜色、姿态等方面变化较为丰富，且与牛、猫有较多相似特征（如某些品种的鸡与猫体型相近），模型可能在鸡类别的分类上准确率相对较低，误判为猫或牛的情况可能较多。而对于牛这种体型较大、特征相对明显且与鸡、猫在形态上差异较大的类别，模型可能表现出较高的准确率，召回率也相对较好。猫的类别可能由于其姿态多样、毛发纹理复杂等因素，处于中等的分类表现水平，可能会出现与鸡类在某些特征上混淆的情况，比如一些小型猫与鸡在图像中仅显示部分身体时，模型可能难以准确区分。

5.2 模型优化

（1）根据分析结果调整模型参数：

学习率调整：如果模型在训练过程中收敛速度过慢，可以适当增大学习率，如将原代码中的 lr=0.0008 调整为 lr=0.001，使模型参数更新步长加大，加快收敛速度。但如果学习率过大，可能会导致模型在训练过程中跳过最优解，出现损失值不收敛甚至增大的情况，此时则需要适当减小学习率。

批处理大小调整：可以尝试调整批处理大小（batch_size），原代码中为 50。增大批处理大小可以提高训练效率，但可能会导致内存占用增加且模型训练时的随机性降低；减小批处理大小则会增加训练的迭代次数，但可能使模型在每次迭代中更好地适应不同的数据样本，有助于提高模型的泛化能力。例如，可以尝试将批处理大小调整为 32 或 64，观察模型性能的变化。

（2）尝试不同的网络结构或正则化技术以提高性能：

1）：网络结构调整：

增加卷积层深度：在原有的 CNN 模型基础上，增加卷积层的数量，如再添加一个类似 conv3 的卷积层结构，进一步提取图像的深层特征，但要注意随着卷积层深度增加可能会出现梯度消失等问题，需要合理设计网络结构并可能添加一些如残差连接的机制来缓解。

调整卷积核大小和步长：尝试不同的卷积核大小和步长组合，例如使用较小的卷积核（如 1×1、3×3）并适当减小步长，以更精细地提取图像特征，同时避免因步长过大导致信息丢失过多。

2）：正则化技术应用：

L2 正则化：在模型的损失函数中添加 L2 正则化项，对模型的权重进行约束，防止模型过拟合。例如在定义损失函数时，可以修改为 `criterion = nn.CrossEntropyLoss() + lambda * torch.sum(torch.square(model.parameters()))`，其中 lambda 为正则化系数，通过调整 lambda 的值来控制正则化的强度。

Dropout 层添加：在全连接层之间添加 Dropout 层，如在 fc1 和 fc2 之间添加 `nn.Dropout(p=0.2)`，其中 p 为随机失活概率。Dropout 层在训练过程中随机将一部分神经元的输出设置为 0，迫使模型学习到更鲁棒的特征表示，减少神经元之间的共适应性，从而提高模型的泛化能力。

6 附录

github 项目链接（含代码、数据） 附带仓库目录结构截图

github 项目链接：https://github.com/LGJ-TO/1213.git

仓库目录结构截图：