# 《人工智能》期中作业报告

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 俞家宝 | 学号 | 2021117338 |
| 实验地点 | 院楼321实验室 | 实验日期 | 2023/11/26 |

基于卷积神经网络的图像识别系统

引言

本报告旨在介绍一个基于图像识别的应用系统。该系统旨在通过训练一个卷积神经网络模型，实现对上传图像的自动分类和识别。该系统的应用领域涵盖了图像识别、物体分类和图像智能处理等方面。通过该系统，用户可以上传图像并获取图像所属的类别信息，例如熊、猫、狗、鸭子、鹅、熊猫、猪、羊和虎等。

相关工作

在图像识别领域，已经存在许多类似的工作和研究。其中，深度学习技术的发展为图像识别带来了重大突破。与传统的机器学习方法相比，深度学习模型可以自动学习图像中的特征表示，从而实现更准确和鲁棒的图像识别。与其他图像识别系统相比，本系统的独特之处在于使用了卷积神经网络（CNN）作为模型架构，CNN能够自动从原始图像数据中学习特征，并通过多层卷积和池化操作实现对图像的高级抽象和表征，并结合了数据增强和归一化等预处理技术，具有更好的性能和更高的准确度。

系统介绍

本系统采用了PyTorch和torchvision库来构建和训练CNN模型。

系统的主要技术包括：

1. 数据预处理：通过使用transforms模块，对输入图像进行随机裁剪、水平翻转、归一化等操作，以增加数据的多样性和泛化能力。
2. 卷积神经网络模型：系统采用了一个三层卷积层和两个全连接层的CNN模型。模型通过卷积、批归一化、激活函数和最大池化等操作，对图像进行特征提取和抽象。
3. 损失函数和优化器：系统使用交叉熵损失函数来衡量预测结果与真实标签之间的差异，并采用随机梯度下降优化器来更新模型参数以最小化损失函数。

实验

数据集：

我们使用了一个包含多类动物图像的数据集，有熊、猫、狗、鸭子、鹅、熊猫、猪、羊和虎等9个类别的图像样本共8892份。

**电脑萤幕画面

中度可信度描述已自动生成电脑萤幕画面

中度可信度描述已自动生成电脑萤幕画面

中度可信度描述已自动生成电脑萤幕画面

低可信度描述已自动生成电脑萤幕画面

中度可信度描述已自动生成桌子上的电脑萤幕画面

低可信度描述已自动生成电脑萤幕画面

中度可信度描述已自动生成电脑萤幕画面

中度可信度描述已自动生成图形用户界面

描述已自动生成**

实验环境设置：

实验在一台配备GPU的计算机上进行，使用PyTorch、tkinter和torchvision库进行开发和训练。系统需要在支持py312以下和相关库的计算机上运行。



文本

描述已自动生成

文本

描述已自动生成

实验结果评定指标：

模型训练过程中，使用交叉熵损失函数作为损失指标，并使用准确率作为评估指标。准确率表示模型对图像分类的准确程度。

实验结果分析：

经过训练过程，模型在训练数据集上逐渐提高准确率，并在达到一定的训练轮次后收敛。在测试数据集上进行评估时，模型表现出较高的准确率，可以对上传的图像进行准确的分类和识别。

程序代码：

加载数据集：

|  |
| --- |
| import torch import torchvision from torchvision import transforms  # 规格化图像 transform = transforms.Compose([  transforms.RandomResizedCrop(256),  transforms.RandomHorizontalFlip(),  transforms.ToTensor(),  transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)) ])  def firstDataSet():  train\_dataset = torchvision.datasets.ImageFolder(  'pathToTrainAnimals', transform=transform)  train\_loader = torch.utils.data.DataLoader(train\_dataset, batch\_size=32, shuffle=True)  # 保存训练数据集  torch.save(train\_dataset, 'train\_dataset.pth')  return train\_loader |

创建CNN模型：

|  |
| --- |
| import torch from torch import nn  class Net(nn.Module):  def \_\_init\_\_(self):  super(Net, self).\_\_init\_\_()  self.conv1 = nn.Conv2d(3, 32, kernel\_size=3, stride=1, padding=1)  self.bn1 = nn.BatchNorm2d(32)  self.conv2 = nn.Conv2d(32, 64, kernel\_size=3, stride=1, padding=1)  self.bn2 = nn.BatchNorm2d(64)  self.conv3 = nn.Conv2d(64, 128, kernel\_size=3, stride=1, padding=1)  self.bn3 = nn.BatchNorm2d(128)  self.relu = nn.ReLU()  self.maxpool = nn.MaxPool2d(kernel\_size=2, stride=2)  self.fc1 = nn.Linear(128 \* 32 \* 32, 256)  self.fc2 = nn.Linear(256, 10)   def forward(self, x):  x = self.conv1(x)  x = self.bn1(x)  x = self.relu(x)  x = self.maxpool(x)  x = self.conv2(x)  x = self.bn2(x)  x = self.relu(x)  x = self.maxpool(x)  x = self.conv3(x)  x = self.bn3(x)  x = self.relu(x)  x = self.maxpool(x)  x = x.view(x.size(0), -1)  x = self.fc1(x)  x = self.relu(x)  x = self.fc2(x)  return x |

模型训练：

|  |
| --- |
| import torch from matplotlib import pyplot as plt from torch import nn, optim from AnimalIdentify import LoadData, Module  train\_loader = LoadData.firstDataSet() model = Module.Net()  criterion = nn.CrossEntropyLoss() optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)  device = torch.device("cuda" if torch.cuda.is\_available() else "cpu") model.to(device)  num\_epochs = 1000 for epoch in range(num\_epochs):  running\_loss = 0.0   for images, labels in train\_loader:  images, labels = images.to(device), labels.to(device)  optimizer.zero\_grad()   outputs = model(images)  loss = criterion(outputs, labels)  loss.backward()  optimizer.step()   running\_loss += loss.item()   print("Epoch [{}/{}], Loss: {:.4f}".format(epoch + 1, num\_epochs, running\_loss / len(train\_loader)))  print("Finished Training")  #测试模型  model.eval()  with torch.no\_grad():  correct = 0  total = 0  for images, labels in train\_loader:  images, labels = images.to(device), labels.to(device)  outputs = model(images)  \_, predicted = torch.max(outputs.data, 1)  total += labels.size(0)  correct += (predicted == labels).sum().item()  print("Accuracy on the test set: {:.2f}%".format(100 \* correct / total))  # 保存模型  torch.save(model.state\_dict(), 'model.pth')  print("Model saved to 'model.pth'") |

用户图形界面GUI：

|  |
| --- |
| import tkinter as tk from tkinter import filedialog from PIL import Image, ImageTk from AnimalIdentify import identifyImage  animal = {0: "熊", 1: "猫", 2: "狗", 3: "鸭子", 4: "鹅", 5: "熊猫", 6: "猪", 7: "羊", 8: "虎"}  def upload\_image():  file\_path = filedialog.askopenfilename()  global preImage  if file\_path:  # 打开图像文件  image = Image.open(file\_path)  preImage = image  # 调整图像大小以适应预览区域  image = image.resize((512, 512))  # 将图像转换为Tkinter可用的格式  photo = ImageTk.PhotoImage(image)  # 在标签上显示图像  image\_label.configure(image=photo)  image\_label.image = photo  # 清除预测内容  updateText("")   def updateText(text):  # 在文本区域显示图像名称  name\_text.configure(state="normal")  name\_text.delete('1.0', tk.END)  name\_text.insert(tk.END, text)  name\_text.configure(state="disabled")  def recognize\_image():  # 在这里编写识别图像的代码  recognize = identifyImage.preIdentify(preImage)  updateText(animal[recognize])  pass  def exit\_app():  root.destroy()  # 创建主窗口 root = tk.Tk() root.title("Animals") root.geometry('1280x720') preImage = ""  # 创建上传按钮 upload\_button = tk.Button(root, text='上传图片', command=upload\_image, width=15, height=3) upload\_button.place(x=300, y=600)  # 创建识别按钮 recognize\_button = tk.Button(root, text='识别', command=recognize\_image, width=15, height=3) recognize\_button.place(x=950, y=400)  # 创建退出按钮 exit\_button = tk.Button(root, text='退出', command=exit\_app, width=15, height=3) exit\_button.place(x=950, y=500)  # 创建图像预览区域 image\_label = tk.Label(root) image\_label.place(x=100, y=50)  # 创建图像名称文本区域 name\_text = tk.Text(root, height=1, width=4, font=('Arial', 108)) name\_text.place(x=900, y=100) name\_text.configure(state="disabled")  # 运行主循环 root.resizable(False, False) root.mainloop() |

图像识别：

|  |
| --- |
| import torch import torchvision.transforms as transforms import Module  def preIdentify(image):  # 加载模型  model = Module.Net()  model.load\_state\_dict(torch.load('model.pth'))  model.eval()   # 图片预处理  transform = transforms.Compose([  transforms.Resize((256, 256)),  transforms.ToTensor(),  transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))  ])   # 图片预处理和模型推理  input\_tensor = transform(image).unsqueeze(0)  output = model(input\_tensor)   # 获取预测结果  \_, predicted\_idx = torch.max(output, 1)  predicted\_class = predicted\_idx.item()   # 输出预测结果  return predicted\_class |

系统界面截图：

系统界面包括一个上传按钮、一个识别按钮和一个退出按钮。用户可以通过上传按钮选择图像文件，并在界面上显示预览图像。点击识别按钮后，系统将调用预测函数对图像进行分类，并在界面上显示识别结果。

图形用户界面

低可信度描述已自动生成图形用户界面, 应用程序

描述已自动生成

猫的照片上写着字

描述已自动生成猫的照片上写着字

描述已自动生成

模型训练结果：

文本

描述已自动生成

系统运行情况：

系统运行稳定，能够准确地对上传的图像进行分类和识别。用户友好的图形用户界面使操作简便，能够满足用户的基本需求。

总结与期望

通过实验结果的分析，我们得出以下结论：该系统在识别不同类别的图像时具有较高的准确性和鲁棒性。然而，系统仍然存在一些潜在的改进空间。例如，可以进一步优化模型的架构和参数设置，以提高系统的性能。此外，可以考虑引入更大规模的训练数据集，以进一步提升模型的泛化能力。未来还可以探索其他图像识别任务和应用领域，如人脸识别、目标检测和图像生成等。总体而言，该图像识别系统具有广阔的应用前景，并有望在实际场景中发挥重要作用。