《深度学习及其应用》实验报告

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 实验名称 | VGG深度学习网络实验 | | | 实验序号 | 3 | 实验日期 | 20231025 |
| 姓 名 | **黄瑞** | **院系** | 计算机学院 | 班 级 | 21104502 | 学 号 | 2110450219 |
| 专 业 | **人工智能** | | | 指导教师 | 林涛 | 成 绩 | 合格 |
| **一、实验目的和要求**  目的：学习VGG深度学习网络结构，并利用深度学习框架搭建VGG网络模型，完成图像分类实验  要求：  1、熟悉VGG16网络的基本结构。  2、利用PYTorch深度学习框架，或者其他深度学习框架完成VGG网络的搭建；可以对VGG16的网络结构进行合理的精简，以提高训练速度。  3、在训练好的VGG网络上，使用实验二的猫狗数据集，完成新图片的猫狗二分类；  4、完成实验报告内容，提交报告。 | | | | | | | |
| **二、实验步骤**    **1、安装实验环境，anaconda；**  安装过程参考实验一和实验二的报告  **2、安装PYTorch**  安装过程参考实验一或实验二的报告内容。  3、下载猫狗数据集，可以自行在互联网上下载，也可以从老师共享的网盘链接中下载  链接：https://pan.baidu.com/s/1hsOlOladL1-j7yV-PBo8uA  提取码：1234  VGG网络结构介绍：  VGG由5层卷积层、3层全连接层、softmax输出层构成，层与层之间使用max-pooling（最大化池）分开，所有隐层的激活单元都采用ReLU函数。  VGG使用多个较小卷积核（3x3）的卷积层代替一个卷积核较大的卷积层，一方面可以减少参数，另一方面相当于进行了更多的非线性映射，可以增加网络的拟合/表达能力。  卷积层全部都是3\*3的卷积核，用conv3-xxx表示，xxx表示通道数。其步长为1，用padding=same填充。  池化层的池化核为2\*2      参考代码：  import torch  import torch.nn as nn  import torch.nn.functional as F  class VGG16(nn.Module):  def \_\_init\_\_(self):  super(VGG16, self).\_\_init\_\_()  # 3 \* 224 \* 224  self.conv1\_1 = nn.Conv2d(3, 64, 3) # 64 \* 222 \* 222  self.conv1\_2 = nn.Conv2d(64, 64, 3, padding=(1, 1)) # 64 \* 222\* 222  self.maxpool1 = nn.MaxPool2d((2, 2), padding=(1, 1)) # pooling 64 \* 112 \* 112  self.conv2\_1 = nn.Conv2d(64, 128, 3) # 128 \* 110 \* 110  self.conv2\_2 = nn.Conv2d(128, 128, 3, padding=(1, 1)) # 128 \* 110 \* 110  self.maxpool2 = nn.MaxPool2d((2, 2), padding=(1, 1)) # pooling 128 \* 56 \* 56  self.conv3\_1 = nn.Conv2d(128, 256, 3) # 256 \* 54 \* 54  self.conv3\_2 = nn.Conv2d(256, 256, 3, padding=(1, 1)) # 256 \* 54 \* 54  self.conv3\_3 = nn.Conv2d(256, 256, 3, padding=(1, 1)) # 256 \* 54 \* 54  self.maxpool3 = nn.MaxPool2d((2, 2), padding=(1, 1)) # pooling 256 \* 28 \* 28  self.conv4\_1 = nn.Conv2d(256, 512, 3) # 512 \* 26 \* 26  self.conv4\_2 = nn.Conv2d(512, 512, 3, padding=(1, 1)) # 512 \* 26 \* 26  self.conv4\_3 = nn.Conv2d(512, 512, 3, padding=(1, 1)) # 512 \* 26 \* 26  self.maxpool4 = nn.MaxPool2d((2, 2), padding=(1, 1)) # pooling 512 \* 14 \* 14  self.conv5\_1 = nn.Conv2d(512, 512, 3) # 512 \* 12 \* 12  self.conv5\_2 = nn.Conv2d(512, 512, 3, padding=(1, 1)) # 512 \* 12 \* 12  self.conv5\_3 = nn.Conv2d(512, 512, 3, padding=(1, 1)) # 512 \* 12 \* 12  self.maxpool5 = nn.MaxPool2d((2, 2), padding=(1, 1)) # pooling 512 \* 7 \* 7  # view  self.fc1 = nn.Linear(512 \* 7 \* 7, 4096)  self.fc2 = nn.Linear(4096, 4096)  self.fc3 = nn.Linear(4096, 1000)  # softmax 1 \* 1 \* 1000  def forward(self, x):  # x.size(0)即为batch\_size  in\_size = x.size(0)  out = self.conv1\_1(x) # 222  out = F.relu(out)  out = self.conv1\_2(out) # 222  out = F.relu(out)  out = self.maxpool1(out) # 112  out = self.conv2\_1(out) # 110  out = F.relu(out)  out = self.conv2\_2(out) # 110  out = F.relu(out)  out = self.maxpool2(out) # 56  out = self.conv3\_1(out) # 54  out = F.relu(out)  out = self.conv3\_2(out) # 54  out = F.relu(out)  out = self.conv3\_3(out) # 54  out = F.relu(out)  out = self.maxpool3(out) # 28  out = self.conv4\_1(out) # 26  out = F.relu(out)  out = self.conv4\_2(out) # 26  out = F.relu(out)  out = self.conv4\_3(out) # 26  out = F.relu(out)  out = self.maxpool4(out) # 14  out = self.conv5\_1(out) # 12  out = F.relu(out)  out = self.conv5\_2(out) # 12  out = F.relu(out)  out = self.conv5\_3(out) # 12  out = F.relu(out)  out = self.maxpool5(out) # 7  # 展平  out = out.view(in\_size, -1)  out = self.fc1(out)  out = F.relu(out)  out = self.fc2(out)  out = F.relu(out)  out = self.fc3(out)  out = F.log\_softmax(out, dim=1)  return out  vgg=VGG16()  print(vgg) | | | | | | | |
| **三、实验结果与分析**  **模型结构:**      1. 数据加载:    2. 模型初始化    3. 模型训练    4. 训练结果    5. 训练模型加载 | | | | | | | |