****

**商务智能课程设计报告**

|  |  |
| --- | --- |
| **题 目** | **二层神经网络处理图像分类** |
| **学 院** | **管理学院** |
| **专 业** | **信息管理与信息系统** |
| **学 号** | **21032313、21032312** |
| **姓 名** | **董文光、陈宗豪** |
| **分组组号** | **18** |
| **分组学号** | **21032313、21032312** |
| **分组姓名** | **董文光、陈宗豪** |
| **日 期** | **2024年7月11日** |

# 目录

[目录 2](#_Toc171590268)

[一、 问题描述 3](#_Toc171590269)

[二、 数据集获取 4](#_Toc171590270)

[三、 数据集描述 5](#_Toc171590271)

[四、 数据集预处理 7](#_Toc171590272)

[五、 数据建模 7](#_Toc171590273)

[六、实验结果分析 12](#_Toc171590274)

[七、总结与下一步改进措施 15](#_Toc171590275)

[八、 小组分工 16](#_Toc171590276)

[九、 个人报告 董文光 21032312 16](#_Toc171590277)

# 问题描述

在互联网时代，图片分类技术已经成为解决许多实际问题的关键工具，在多个领域具有重要的应用价值和现实意义：

1. **电子商务：**

• **产品推荐**：分析用户上传的商品图片，根据商品特征进行分类和推荐，提高购物体验。

• **假货检测**：帮助识别假冒商品，保护品牌和消费者的利益。

1. **智能设备：**

• **自动驾驶**：识别道路标志、交通信号、行人和其他车辆，辅助决策，提高驾驶安全性。

• **智能家居**：识别家庭成员的行为和状态，自动调节家电设备，提升用户体验。

1. **安全/监控**

• **人脸识别**：在安全监控系统中，图像分类技术可以用于人脸识别，验证身份，提升安全性。

• **异常行为检测**：在公共安全监控中，图像分类模型可以用于识别异常行为或事件，预防犯罪，保护公共安全。

1. **医疗诊断**

• **医学影像分析**：图像分类模型可以辅助医生分析医学影像，早期发现疾病（如癌症），提高诊断的准确性和效率。

• **皮肤病诊断**：通过对皮肤图片进行分类，图像分类模型可以帮助医生识别皮肤病变，提供诊断支持。

1. **农业/制造业**

• **制造业**：通过图像分类技术，可以实现生产线上产品质量的自动检测，识别有缺陷的产品，从而减少人工检测成本，提高生产效率。

• **农业**：在农业中，图像分类技术可以用于识别农作物的病虫害，帮助农民及时采取防治措施，保障农作物健康成长，提高农业产量。

在本项目中，我们的目标是使用 PyTorch 实现一个二层神经网络，构建和训练一个神经网络模型，使其能够准确地对这些图像进行分类。

# 二、 数据集获取

在本项目中，我们选择使用 CIFAR-10 数据集进行图像分类任务。

**CIFAR-10数据集介绍：**

CIFAR-10 是一个常用的计算机视觉数据集，全称为 Canadian Institute For Advanced Research 10-class dataset。该数据集是由加拿大高级研究院（Canadian Institute For Advanced Research, CIFAR）提供的一个广泛使用的计算机视觉数据集，由 60000 张 32x32 像素的彩色图像组成，共分为 10 个类别，每个类别有 6000 张图像。具体类别包括：飞机、汽车、鸟、猫、鹿、狗、青蛙、马、船和卡车。

**使用CIFAR-10 数据集的理由：**

1. **多样性和代表性：**

CIFAR-10 数据集包含 10 个不同类别的图像，每个类别均衡分布。这些类别涵盖了日常生活中的常见物体，使得该数据集具有较好的多样性和代表性，适合进行通用的图像分类任务。

1. **数据规模适中：**

数据集包含的图像数量适中，既足够大以训练出有效的深度学习模型，又不会因为数据量过大而导致训练时间过长。这使得 CIFAR-10 成为学术研究和课程设计中的常用数据集。

1. **数据处理方便：**

CIFAR-10 数据集的图像尺寸为 32x32 像素，数据格式简单且易于处理。对于初学者和研究人员来说，可以快速上手并进行模型训练和验证。

1. **广泛使用和标准化：**

CIFAR-10 数据集在计算机视觉领域被广泛使用，具有标准化的评估指标和大量的基准模型结果。这使得研究人员可以方便地对比新模型与现有模型的性能，进行结果评估和改进。

1. **支持多种框架：**

CIFAR-10 数据集可以通过多种深度学习框架（如 PyTorch、TensorFlow）方便地下载和加载。这为模型的开发和实验提供了极大的便利。

**导入CIFAR10数据集：**

**首先先导入必要的 python 依赖：**

|  |
| --- |
| **代码实现：** |
| import utils  import torch  import torchvision  import matplotlib.pyplot as plt  import statistics  import random  import time  import math |

**导入数据集：**

|  |
| --- |
| **代码实现：** |
| *# 设置随机种子确保实验结果可重复*  utils**.**reset\_seed(0)  *# 导入数据集*  data\_dict **=** utils**.**data**.**preprocess\_cifar10(dtype**=**torch**.**float64)  *# 输出数据集形状*  print('Train data shape: ', data\_dict['X\_train']**.**shape)  print('Train labels shape: ', data\_dict['y\_train']**.**shape)  print('Validation data shape: ', data\_dict['X\_val']**.**shape)  print('Validation labels shape: ', data\_dict['y\_val']**.**shape)  print('Test data shape: ', data\_dict['X\_test']**.**shape)  print('Test labels shape: ', data\_dict['y\_test']**.**shape) |

# 三、 数据集描述

**CIFAR-10数据集具体结构：**

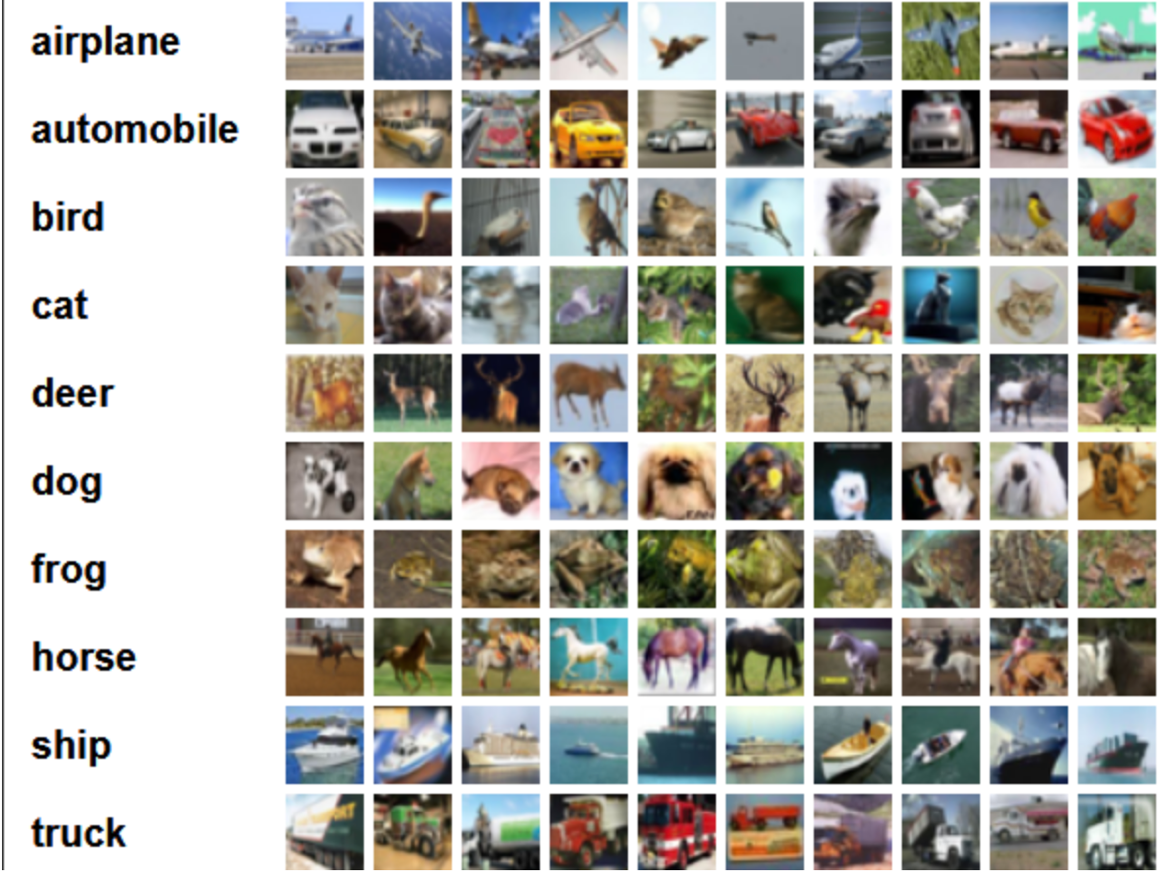
* 训练集：包含 50000 张图像，每个类别 5000 张。
* 测试集：包含 10000 张图像，每个类别 1000 张。

**CIFAR-10数据集具体数据类别：**

CIFAR-10是一个更接近普适物体的彩色图像数据集。CIFAR-10 是由Hinton 的学生Alex Krizhevsky 和Ilya Sutskever 整理的一个用于识别普适物体的小型数据集。一共包含10 个类别的RGB 彩色图片：

* 飞机（ airplane ）
* 汽车（ automobile ）
* 鸟类（ bird ）
* 猫（ cat ）
* 鹿（ deer ）
* 狗（ dog ）
* 蛙类（ frog ）
* 马（ horse ）
* 船（ ship ）
* 卡车（ truck ）

**CIFAR-10数据集具体数据示例：**



# 四、 数据集预处理

**1. 规范化：**

使用 zero-mean 对数据集进行规范化：

|  |
| --- |
| 代码实现： |
| mean\_image = X\_train.mean(*dim*=(0, 2, 3), *keepdim*=True)  X\_train -= mean\_image  X\_test -= mean\_image |

**2. flatten 降维：**

将多维图像数据转换为一维数组，直接应用于神经网络模型，简化了数据输入的准备过程：

|  |
| --- |
| 代码实现： |
| X\_train = X\_train.reshape(X\_train.shape[0], -1)  X\_test = X\_test.reshape(X\_test.shape[0], -1 |

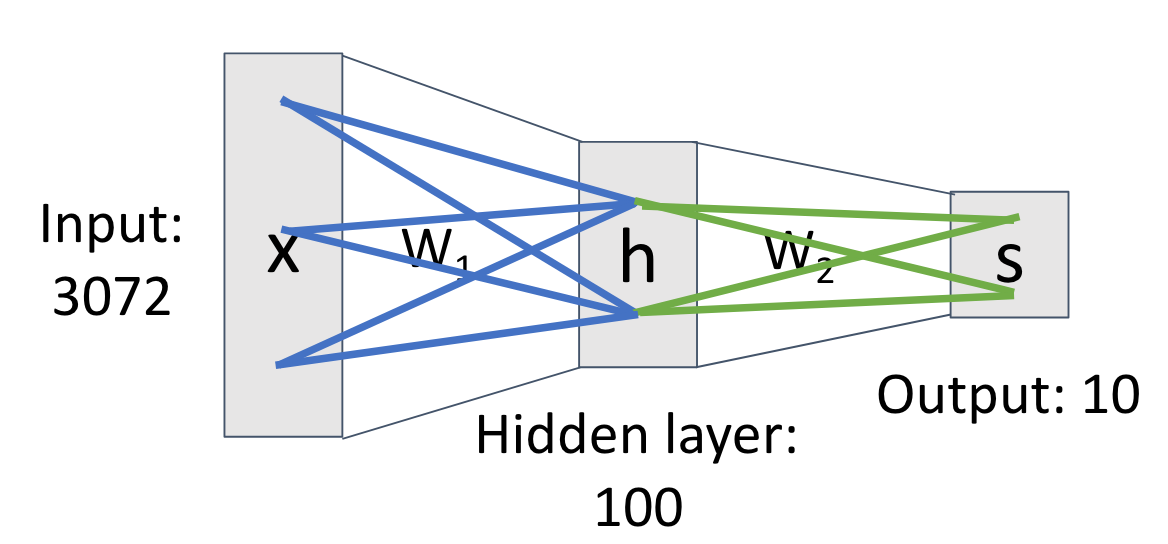
# 五、 数据建模

在数据预处理之后，我们构建并训练了一个二层神经网络模型。以下是模型的构建和训练过程的详细描述。

1. **模型构建**：

• **使用 PyTorch 构建了一个二层神经网络模型**。模型包含一个输入层、一个隐藏层和一个输出层。输入层接受图像数据，隐藏层进行特征提取，输出层进行分类。

二层神经网络包括：

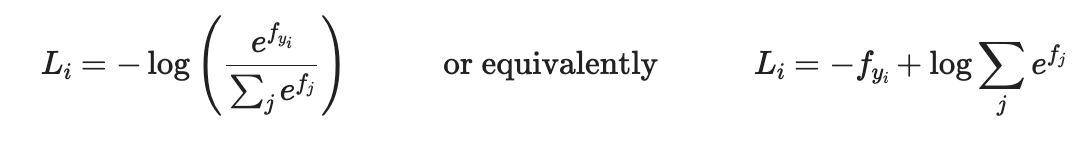
* 输入层
* 隐藏层， 使用ReLU（Rectified Linear Unit）激活函数
* 输出层

• **前向传播**：计算 scores，给定一张图片，我们可以根据以下公式计算它分别属于10个类别的分数。

z=(XW1+b1)W2+b2𝑧=(𝑋𝑊1+𝑏1)𝑊2+𝑏2

input layer -> ReLU (hidden) -> output layer (scores)

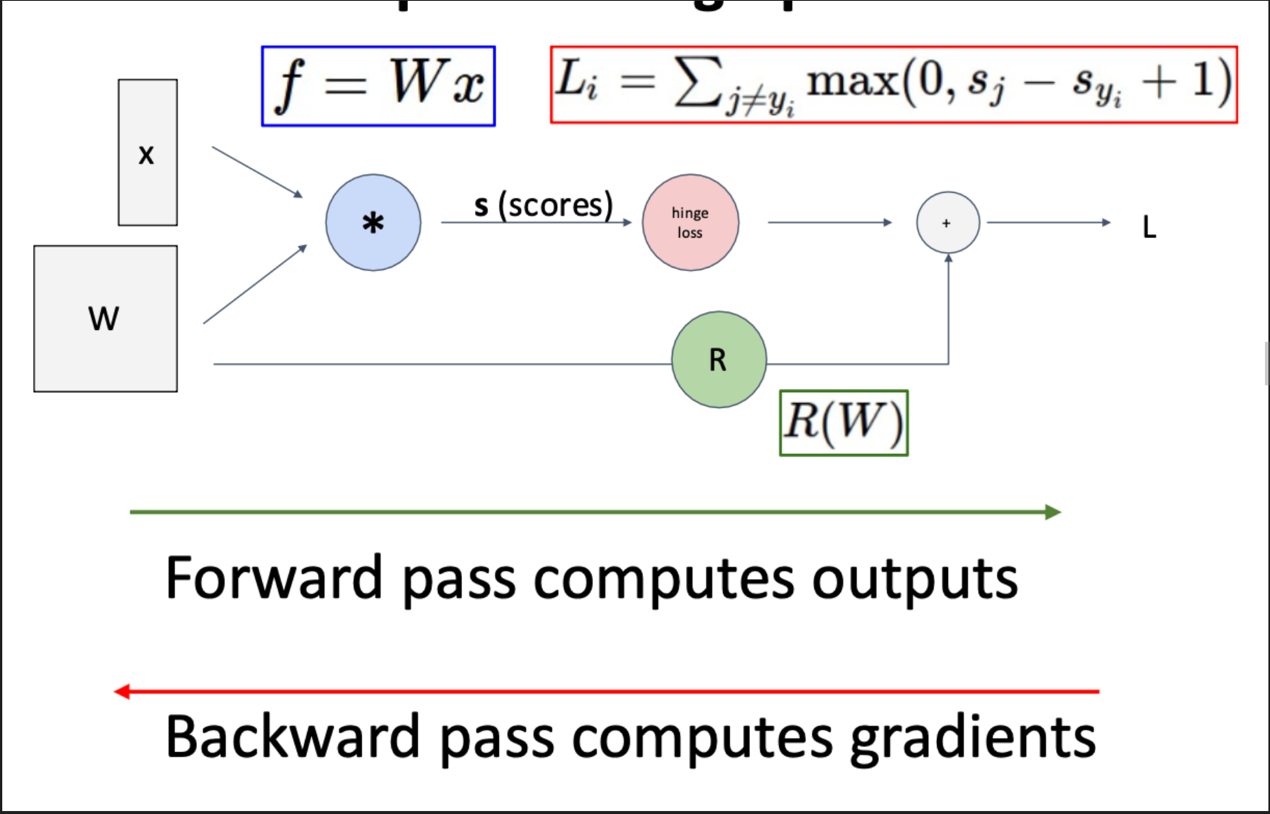
|  |
| --- |
| 代码实现： |
| W1, b1 **=** params["W1"], params["b1"]  W2, b2 **=** params["W2"], params["b2"]  N, D **=** X**.**shape  *# first layer*  hidden **=** X**.**mm(W1) **+** b1 *# (N, D) \* (D, H) + (H,)*  hidden[hidden **<** 0] **=** 0 *# relu*  *# second layer*  scores **=** hidden**.**mm(W2) **+** b2 *# (N, H) \* (H, C) + (C,)* |

• **损失函数选择：**在本次课程设计中，我们选取了softmax 作为损失函数， 其计算公式如下：

|  |
| --- |
| 代码实现： |
| scores **=** scores **-** scores**.**max(1, **True**)**.**values *# normalization*  softmax **=** scores **-** scores**.**logsumexp(1, **True**) *# one row loss*  loss **=** **-**softmax[range(N), y]**.**mean() *# mean*  *# regularization*  loss **+=** reg **\*** torch**.**sum(W1 **\*** W1) **+** reg **\*** torch**.**sum(W2 **\*** W2) |

• **模型优化：**在本次课程设计中，我们使用随机梯度下降SGD作为优化方法。

1. 计算梯度：

∇Θℓsoftmax(XΘ,y)=1mXT(Z−Iy)

|  |
| --- |
| 代码实现 |
| prob **=** softmax**.**exp() *# (N, C)*  prob[torch**.**arange(N), y] **-=** 1  prob **/=** N    grads['b2'] **=** prob**.**sum(0) *# (C, )*  grads['W2'] **=** h1**.**T**.**mm(prob) **+** reg **\*** 2 **\*** W2 *# (H, C)*  *# Relu gradient*  grads\_h1 **=** prob**.**mm(W2**.**t())  grads\_h1[h1 **<=** 0] **=** 0  grads['b1'] **=** grads\_h1**.**sum(0)  grads['W1'] **=** X**.**T**.**mm(grads\_h1) **+** 2 **\*** reg **\*** W1 |

1. 随机梯度下降：

|  |
| --- |
| 代码实现 |
| **for** it **in** range(num\_iters):  X\_batch, y\_batch **=** sample\_batch(X, y, num\_train, batch\_size)  *# Compute loss and gradients using the current minibatch*  loss, grads **=** loss\_func(params, X\_batch, y**=**y\_batch, reg**=**reg)  loss\_history**.**append(loss**.**item())  *# update parameter*  params['W1'] **-=** learning\_rate **\*** grads['W1']  params['b1'] **-=** learning\_rate **\*** grads['b1']  params['W2'] **-=** learning\_rate **\*** grads['W2']  params['b2'] **-=** learning\_rate **\*** grads['b2'] |

2. **模型训练**：

• 在训练过程中，我们记录了每个 epoch 的训练损失和训练准确率，以观察模型的收敛情况。训练损失逐渐减小，训练准确率逐渐提高，表明模型在不断学习和优化：

|  |
| --- |
| 代码实现： |
| **import** os  **import** utils  **from** utils.plot **import** plot\_stats  **from** two\_layer\_net **import** TwoLayerNet, find\_best\_net, nn\_get\_search\_params  *# 使用 float32 类型来减少训练时间*  utils**.**reset\_seed(0)  data\_dict **=** utils**.**data**.**preprocess\_cifar10(dtype**=**torch**.**float32)  *# ==此处为了演示速度我直接选取了目前最好的超参数组合==*  *# 寻找出最好的预设参数*  utils**.**reset\_seed(0)  best\_net, best\_stat, best\_val\_acc **=** find\_best\_net(data\_dict, nn\_get\_search\_params)  print(best\_val\_acc)  plot\_stats(best\_stat)  *# save the best model*  path **=** os**.**path**.**join('.', 'nn\_best\_model.pt')  best\_net**.**save(path) |

• 超参数（Hyper-parameter）调整：

|  |
| --- |
| 代码实现： |
| *# ...*  **for** learning\_rate **in** learning\_rates:  **for** hidden\_size **in** hidden\_sizes:  **for** reg **in** regularization\_strengths:  **for** learning\_rate\_decay **in** learning\_rate\_decays:  *# train, get accuracy*  *# compare and get best* |

|  |
| --- |
| 代码输出： |
| train with hyperparameters: lr = 1.0, hs = 1024, reg = 0.001, lr\_decay = 0.97: val\_acc = 0.5594000220298767  hyperparameters with best model: lr = 1.0, hs = 1024, reg = 0.001, lr\_decay = 0.97  0.5594000220298767 |

# 六、实验结果分析

模型分析与评估：

我们使用 CIFAR-10 数据集的测试集部分来评估我们构建的二层神经网络模型的性能。测试集包含 10000 张图像，每张图片与训练集同样，尺寸为 32x32 像素，包含 RGB 三个通道（彩色图像），并且图像均衡分布在 10 个类别中，每个类别包含 1000 张图像。

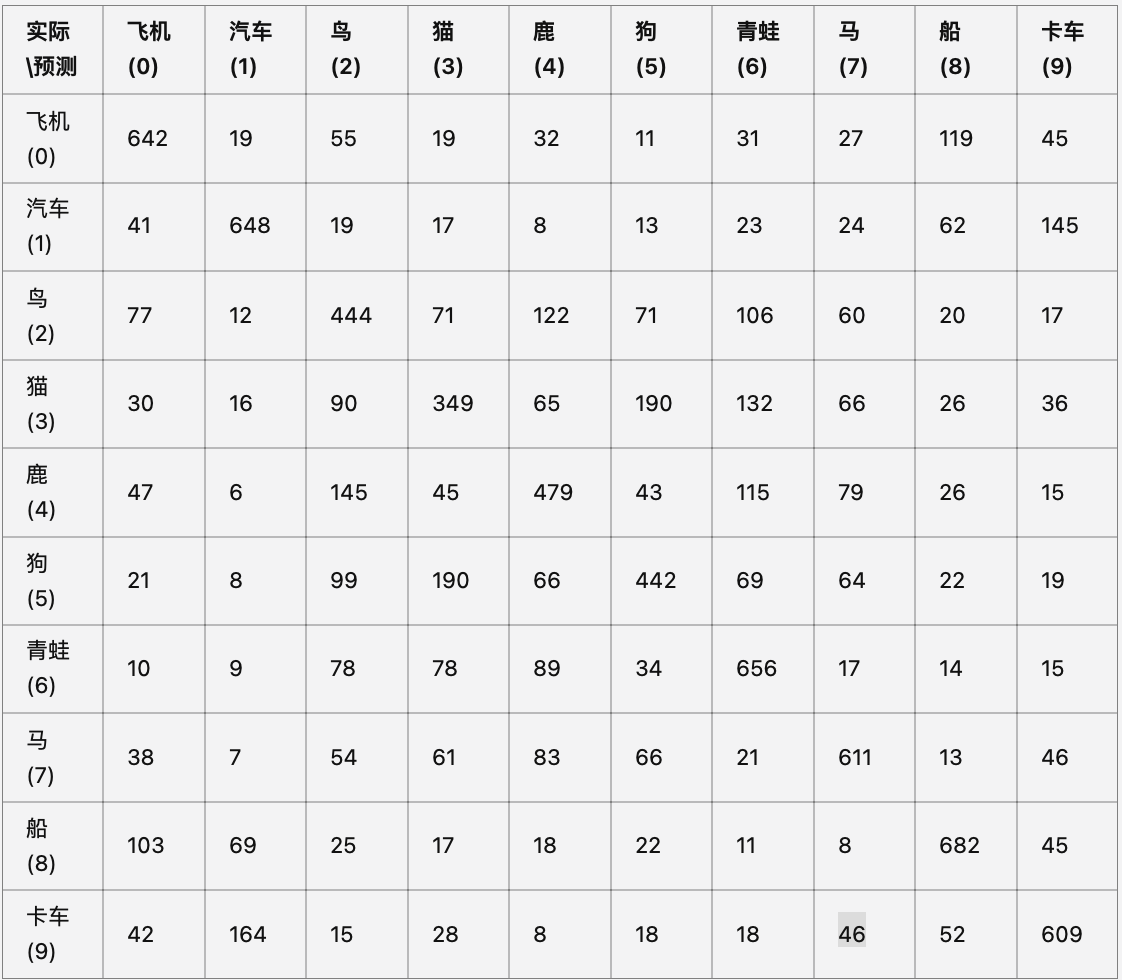
**在整个测试集上测试模型准确率：**

|  |
| --- |
| 代码实现： |

|  |
| --- |
| # 预测测试集准确率  y\_test\_preds = best\_net.predict(data\_dict['X\_test'])  test\_acc = 100 \* (y\_test\_preds == data\_dict['y\_test']).double().mean().item()  print('Test accuracy: %.2f%%' % test\_acc)  Test accuracy: 55.62% |

**在测试集中随机抽取五张图片测试：**

|  |
| --- |
| 代码实现： |
| from utils.utils import visualize\_dataset  # class 种类  class\_list = ['airplane', 'automobile', 'bird', 'cat', 'deer',  'dog', 'frog', 'horse', 'ship', 'truck']  rand\_num = 5  # 从测试集中随机选取 5 张图片  random.seed(0) # Set random seed for reproducibility  indices = random.sample(range(len(data\_dict['X\_test'])), rand\_num)  images = data\_dict['X\_test'][indices] # shape torch.Size([5, 3072])  labels = data\_dict['y\_test'][indices]  # 对比预测结果和真实标签  predicted\_labels = best\_net.predict(images)  for i in range(len(images)):  print(f"Image {i+1}: True label - {class\_list[labels[i]]}, Predicted label - {class\_list[predicted\_labels[i]]}")  # 输出这五张图片  images = images.view(5, 3, 32, 32)  img = torchvision.utils.make\_grid(images, nrow=5)  plt.imshow(utils.tensor\_to\_image(img))  plt.axis("off")  plt.show()   |  | | --- | | 代码结果： | | Image 1: True label - cat, Predicted label - cat  Image 2: True label - dog, Predicted label - dog  Image 3: True label - frog, Predicted label - frog  Image 4: True label - ship, Predicted label - airplane  Image 5: True label - horse, Predicted label - truck | |
|  |

**混淆矩阵分析：**

**正确比例：**

类别 0 的正确分类比例: 64.20%

类别 1 的正确分类比例: 64.80%

类别 2 的正确分类比例: 44.40%

类别 3 的正确分类比例: 34.90%

类别 4 的正确分类比例: 47.90%

类别 5 的正确分类比例: 44.20%

类别 6 的正确分类比例: 65.60%

类别 7 的正确分类比例: 61.10%

类别 8 的正确分类比例: 68.20%

类别 9 的正确分类比例: 60.90%

1. **总体表现**：

• 大多数类别的正确分类数（对角线上的数字）较高，这表明模型在大多数情况下能够正确分类。

2. **类别混淆**：

• **飞机 (0) 与船 (8)**：有 119 个实际为飞机的样本被错误分类为船，说明这两个类别在某些特征上可能有相似之处。

• **汽车 (1) 与卡车 (9)**：有 145 个实际为汽车的样本被错误分类为卡车，这表明模型在区分汽车和卡车时有一定的困难。

3. **类别间差异**：

• **鸟 (2)**：有大量的样本被错误分类为猫 (3)、鹿 (4)、狗 (5) 和青蛙 (6)，表明这些类别之间的特征可能有较大重叠。

• **猫 (3)** 和 **狗 (5)**：这两个类别互相混淆较多，实际为猫的样本中有 190 个被错误分类为狗，实际为狗的样本中有 190 个被错误分类为猫。这表明模型在区分猫和狗时存在较大困难，可能是因为这两个类别的外观特征比较相似。

4. **性能较好的类别**：

• **青蛙 (6)**：正确分类数为 656，错误分类数较少，表明模型在识别青蛙时表现较好。

• **马 (7)**：正确分类数为 611，错误分类数较少，表明模型在识别马时也表现较好。

# 七、总结与下一步改进措施

**项目总结：**

在本项目中，我们使用 PyTorch 构建并训练了一个二层神经网络模型，对 CIFAR-10 数据集进行了图像分类任务，并在最后分析、评估了模型。我们得出模型的优势与不足包括：

1. **优势：**

• **简单且有效**：二层神经网络结构简单，训练速度快，能够在 CIFAR-10 数据集上取得较好的分类性能。

• **良好的泛化能力**：测试准确率较高，表明模型在新数据上的表现良好。

1. **不足：**

• **复杂图像处理能力有限**：模型结构较为简单，对于图像特征的提取和处理能力有限，可能无法很好地处理复杂图像。

• **类别间差异处理不足**：模型在某些类别上的分类效果不佳，说明在处理类别间差异时存在不足。

**下一步改进措施：**

显而易见，我们的二层神经网络分类并没有达到对图片分类的完美效果，因此，该模型还有大量的改进空间，为了让图片识别与分类计数有更高的准确率，我们未来可以采用以下一些方向进行模型或者方法的优化，以达到更好的分类效果：

* 增加模型复杂度
* 优化训练过程
* 使用预训练模型
* 正则化技术
* 集成学习

# 八、 小组分工

董文光 21032313 ：

问题描述、数据收集、数据探索、模型分析。

陈宗豪 21032312 :

数据预处理、数据建模。

# 九、 个人报告 董文光 21032312

在本次商务智能课程的设计中，我们小组选择的项目是使用 PyTorch 实现一个二层神经网络，构建和训练一个神经网络模型，使其能够准确地对这些图像进行分类。神经网络是人工智能领域非常流行的理念，而图像识别则是一个较有历史的问题。我们首先通过检索和查阅文献来了解基本的神经网络的理念，然后在网络上找到了构建神经网络模型来进行图像分类的教程。由于时间和技术的限制，我们选择了二层神经网络的模型。并随后进行了一系列数据挖掘工作，包括：问题描述、数据收集、数据探索、数据预处理、数据建模、模型分析。

我在本次项目中负责问题描述、数据收集、数据探索、模型分析部分。在数据收集部分，选择CIFAR-10 数据集，并从网上的开源资源获取该数据集和基本的数据描述。在数据探索阶段，通过编程实现该数据集的某些图片输出，并结合开源资源中对数据集的基本介绍以了解该数据集的详情信息并验证内容是否正确。在建模完成后，通过输出准确率以评估模型的整体性能，得到55.62%的准确率，然后输出五张随机图片和相应的预测分类以直观展示其能力。另外，我们还通过构建混淆矩阵的形式更加详细和直观地分析模型能力，得到了该模型对每个不同图片类别的表现。