zhejiang-yige

智能软件综合研究

实训报告



学期 2024-2025-1

班级 22计科（3）

姓名 张雅瑞

学号 2022334323029

教师 马明泽

|  |  |  |
| --- | --- | --- |
| 学号： 2022334323029 |  | 姓名： 张雅瑞 |

1. **实验目的和要求**
   1. 掌握数据预处理相关操作；
   2. 掌握深度学习模型构建、训练、评估、优化等过程；
   3. 掌握智能软件全周期开发、部署过程。
2. **实验内容**

Kaggle提供的眼部疾病识别数据集是一个结构化的眼科数据库，包含5000名患者的数据。每个样本包含1个ID号，6个与眼部疾病相关的特征（例如，年龄、性别、左右眼底照片、左右眼诊断关键字），以及8个分类标签（正常N、糖尿病D、青光眼G、白内障C、年龄相关性黄斑变性A、高血压H、病理性近视、其他疾病/异常O）。通过数据预处理、数据集划分、模型构建、模型训练、模型评估等步骤，完成眼部疾病识别任务。

1. **实验任务**

第10周任务要求按以下步骤完成眼部疾病识别问题。

1. **环境配置。**报告中需说明实验环境相关信息。
2. **数据加载及预处理**

* 数据加载及数据确认。要求展示头部数据、尾部数据、数据类型，确认数据完整性。
* 数据预处理。

（1）检查是否有重复数据并删除；

（2）检查是否存在缺失数据以及各特征缺失数据的比例，对缺失数据进行填充操作；

（3）将字符串和布尔字符转换成整数，完成编码操作；

（4）尝试分析特征之间的关系，将其可视化。

* 预处理数据保存。

1. **数据集分割。**将预处理后的数据集分割成训练集和测试集，其中训练集70%，测试集30%。
2. **模型构建、训练、评估**

利用预训练的VGG19模型对以上预处理后的数据集进行训练、可视化accuracy和loss，混淆矩阵，并对实验数据进行详细分析。

1. **实验过程和结果分析**

数据读取

|  |
| --- |
| import numpy as np # linear algebra  import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)  import cv2  import random  from tqdm import tqdm  import matplotlib.pyplot as plt  from tensorflow.keras.preprocessing.image import ImageDataGenerator  import os  os.environ['TF\_MLIR\_ENABLE'] = '1'  df = pd.read\_csv("archive/full\_df.csv")  df.head(3)  def has\_cataract(text):      if "cataract" in text:          return 1      else:          return 0  df["left\_cataract"] = df["Left-Diagnostic Keywords"].apply(lambda x: has\_cataract(x))  df["right\_cataract"] = df["Right-Diagnostic Keywords"].apply(lambda x: has\_cataract(x))  left\_cataract = df.loc[(df.C ==1) & (df.left\_cataract == 1)]["Left-Fundus"].values  left\_cataract[:15]  right\_cataract = df.loc[(df.C ==1) & (df.right\_cataract == 1)]["Right-Fundus"].values  right\_cataract[:15]  print("Number of images in left cataract: {}".format(len(left\_cataract)))  print("Number of images in right cataract: {}".format(len(right\_cataract)))  left\_normal = df.loc[(df.C ==0) & (df["Left-Diagnostic Keywords"] == "normal fundus")]["Left-Fundus"].sample(250,random\_state=42).values  right\_normal = df.loc[(df.C ==0) & (df["Right-Diagnostic Keywords"] == "normal fundus")]["Right-Fundus"].sample(250,random\_state=42).values  right\_normal[:15]  cataract = np.concatenate((left\_cataract,right\_cataract),axis=0)  normal = np.concatenate((left\_normal,right\_normal),axis=0)  print(len(cataract),len(normal)) |

创建数据集

|  |
| --- |
| from tensorflow.keras.preprocessing.image import load\_img,img\_to\_array  dataset\_dir = "archive/preprocessed\_images/"  image\_size=224  labels = []  dataset = []  def create\_dataset(image\_category,label):      for img in tqdm(image\_category):          image\_path = os.path.join(dataset\_dir,img)          try:              image = cv2.imread(image\_path,cv2.IMREAD\_COLOR)              image = cv2.resize(image,(image\_size,image\_size))          except:              continue            dataset.append([np.array(image),np.array(label)])      random.shuffle(dataset)      return dataset    dataset = create\_dataset(cataract,1)  len(dataset)  dataset = create\_dataset(normal,0)  len(dataset) |

查看数据

|  |
| --- |
| plt.figure(figsize=(12,7))  for i in range(10):      sample = random.choice(range(len(dataset)))      image = dataset[sample][0]      category = dataset[sample][1]      if category== 0:          label = "Normal"      else:          label = "Cataract"      plt.subplot(2,5,i+1)      plt.imshow(image)      plt.xlabel(label)  plt.tight\_layout() |

数据预处理

|  |
| --- |
| import numpy as np  from sklearn.model\_selection import train\_test\_split  from tensorflow.keras.utils import to\_categorical  # 假设 x 和 y 是你的数据和标签  # 确保 x 和 y 是 NumPy 数组  x = np.array([i[0] for i in dataset]).reshape(-1, image\_size, image\_size, 3)  y = np.array([i[1] for i in dataset])  # 数据集划分  x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.2, random\_state=42)  # 将标签转换为独热编码  y\_train = to\_categorical(y\_train, num\_classes=8)  y\_test = to\_categorical(y\_test, num\_classes=8)  # 打印转换后的标签形状以验证  print("y\_train shape:", y\_train.shape)  print("y\_test shape:", y\_test.shape)  np.isnan(x\_train).any(), np.isnan(y\_train).any()  np.isinf(x\_train).any(), np.isinf(y\_train).any() |

数据增强

|  |
| --- |
| datagen = ImageDataGenerator(      rotation\_range=20,      width\_shift\_range=0.2,      height\_shift\_range=0.2,      shear\_range=0.2,      zoom\_range=0.2,      horizontal\_flip=True,      fill\_mode='nearest'  )  datagen.fit(x\_train) |

多分类处理

|  |
| --- |
| # 构建模型  from tensorflow.keras.applications.vgg19 import VGG19  from tensorflow.keras.models import Model, load\_model  from tensorflow.keras.layers import Dense, Flatten, GlobalAveragePooling2D  base\_model = VGG19(weights='imagenet', include\_top=False, input\_shape=(224, 224, 3))  x = base\_model.output  x = GlobalAveragePooling2D()(x)  x = Dense(256, activation='relu')(x)  predictions = Dense(8, activation='softmax')(x)  model = Model(inputs=base\_model.input, outputs=predictions)  # 解冻 VGG19 的后几层以进行微调  for layer in base\_model.layers[:15]:      layer.trainable = False  for layer in base\_model.layers[15:]:      layer.trainable = True    from tensorflow.keras.utils import to\_categorical  from tensorflow.keras.optimizers import Adam  from tensorflow.keras.callbacks import ReduceLROnPlateau  # 编译模型  model.compile(optimizer=Adam(learning\_rate=1e-4), loss='categorical\_crossentropy', metrics=['accuracy'])  # 设置学习率调节器  reduce\_lr = ReduceLROnPlateau(monitor='val\_loss', factor=0.2, patience=5, min\_lr=1e-5)  # 模型训练  history = model.fit(datagen.flow(x\_train, y\_train, batch\_size=32), validation\_data=(x\_test,y\_test),                      epochs=50, callbacks=[reduce\_lr])  # history = model.fit(x\_train,y\_train,batch\_size=16,epochs=15,validation\_data=(x\_test,y\_test),  #                     verbose=1,callbacks=[checkpoint,earlystop])  loss,accuracy = model.evaluate(x\_test,y\_test)  print("loss:",loss)  print("Accuracy:",accuracy) |

打印报告

|  |
| --- |
| from sklearn.metrics import confusion\_matrix, classification\_report, accuracy\_score  import numpy as np  # 获取预测结果  y\_pred\_prob = model.predict(x\_test)  # 将预测结果转换为类别索引  y\_pred = np.argmax(y\_pred\_prob, axis=1)  # 将独热编码的标签转换回原始标签  y\_test\_orig = np.argmax(y\_test, axis=1)  # 计算准确率  accuracy = accuracy\_score(y\_test\_orig, y\_pred)  print("Accuracy:", accuracy)  # 打印分类报告  print(classification\_report(y\_test\_orig, y\_pred))  # 计算混淆矩阵  cm = confusion\_matrix(y\_test\_orig, y\_pred)  print("Confusion Matrix:\n", cm)  # 绘制混淆矩阵  from mlxtend.plotting import plot\_confusion\_matrix  plot\_confusion\_matrix(conf\_mat=cm, figsize=(8, 7), class\_names=["Normal", "Cataract"], show\_normed=True) |

打印学习曲线

|  |
| --- |
| plt.style.use("ggplot")  fig = plt.figure(figsize=(12,6))  # epochs = range(1,16)  epochs = range(1, len(history.history["accuracy"]) + 1)  plt.subplot(1,2,1)  plt.plot(epochs,history.history["accuracy"],"go-")  plt.plot(epochs,history.history["val\_accuracy"],"ro-")  plt.title("Model Accuracy")  plt.xlabel("Epochs")  plt.ylabel("Accuracy")  plt.legend(["Train","val"],loc = "upper left")  plt.subplot(1,2,2)  plt.plot(epochs,history.history["loss"],"go-")  plt.plot(epochs,history.history["val\_loss"],"ro-")  plt.title("Model Loss")  plt.xlabel("Epochs")  plt.ylabel("Loss")  plt.legend(["Train","val"],loc = "upper left")  plt.show() |

预测结果

|  |
| --- |
| # 预测结果展示  plt.figure(figsize=(12, 7))  for i in range(10):      sample = random.choice(range(len(x\_test)))      image = x\_test[sample]      category = y\_test\_orig[sample]      pred\_category = y\_pred[sample]        if category == 0:          label = "Normal"      else:          label = "Cataract"        if pred\_category == 0:          pred\_label = "Normal"      else:          pred\_label = "Cataract"        plt.subplot(2, 5, i + 1)      plt.imshow(image)      plt.xlabel("Actual:{}\nPrediction:{}".format(label, pred\_label))  plt.tight\_layout()  plt.show() |

1. **实验总结**
   1. **总结**

通过这次实验，我对分类任务的代码设计、问题调试和模型优化有了更深入的理解，总结以下几点：

1. **明确任务目标是模型设计的核心：**
   * 二分类和多分类任务在标签格式、输出层设计和损失函数选择上差异显著。如果目标不明确，容易导致代码混乱，模型性能受损。
2. **统一代码逻辑，避免混杂：**
   * 本次实验中，二分类和多分类的代码逻辑混杂，导致训练和评估过程多次报错。明确任务后，需全局调整代码逻辑，保持一致性。
3. **数据增强要结合领域特性：**
   * 医学图像分类对特征保真的要求较高，数据增强策略需根据具体任务调整，避免特征失真。
4. **充分利用预训练模型：**
   * 预训练模型如VGG19在特征提取上表现优秀，但应结合任务冻结部分层，只微调后几层以适应特定数据集。
5. **逐步调试，分析结果：**
   * 本次实验中，由于标签处理不当、模型设计冲突等问题，模型性能反复受影响。通过逐步调整代码逻辑，明确问题来源，可以有效改进实验结果。
   1. **心得体会**

我第一次实验时，采用的二分类的代码来实验，但是复现效果不佳，于是将多分类的模型设计来改善原来的部分代码，但由于二分类的代码和多分类的代码混合，模型效果依旧不佳

这次实验让我深刻认识到二分类与多分类任务在设计逻辑上的差异，以及代码一致性的重要性。未来，我将更注重实验的整体规划，确保代码结构清晰，目标明确，并针对问题进行有针对性的优化。