2024-2025 学年第 2 学期

计算机科学与技术学院

(人工智能学院)

《机器学习项目实践》

报告

指导教师：周维达,方贤

班级：计算机科学与技术 21(4)班

学号：2021329600006

姓名：陈昊天

机器学习项目实践

任　务　书

一、设计目的：

1. 掌握机器学习项目方案设计的一般方法，主要包括系统分析、系统设计的组织和实施。

2. 掌握机器学习和图像数据处理方法和技术。

3. 培养把所学知识运用到具体实践对象，并能提出详细解决方案的能力。

二、任务要求：

1. 内容：选做以下题目之一：

   选做一：手写数字识别

   选做二：人脸识别

   选做三：车牌识别

   选做四：基于图像的手势识别

   选做五：纹理图像分类

2. 要求：上交一份课程设计报告；简要叙述算法的基本原理和步骤；给出带有详细注释的算法的实现代码；对代码运行结果进行分析说明；将实验过程中遇到的问题和解决方法写在心得体会中。

三、结果形式：

1、系统及界面、模块实现。

2、程序流程设计：关于所设计系统的模块结构和设计要点。

3、程序代码。

4、程序功能使用说明：包括程序功能模块的功能介绍和操作方法。

5、上机正常运行。

四、工作进度：　(共 2 周)

序号	实 践 内 容	时间	授 课 地 点	指导教师

1	Python 语言基础，机器学习概论	0.5 周	机 3	周维达/方贤
2	聚类的原理，神经网络模型，深度学习框架	0.5 周	机 3	周维达/方贤
3	任务布置，需求分析， 系统设计，编写代码	0.5 周	机 3	周维达/方贤
4	程序测试，考核答辩	0.5 周	机 3	周维达/方贤
合计		2 周		

五、成绩评定标准与考核：

本课程设计的评价由三部分组成，常规实验平时分（40%），课程设计报告（30%），综合实验演示回答教师提问（30%）。

1、 常规实验平时分：

(1) 优　　功能完善，全部测试正确，并且能够对局部进行完善

(2) 良　　功能完善，但测试欠缺

(3) 中　　功能基本完善，但程序尚有部分错误

(4) 及格　　实现了主要功能基本完善，有部分功能尚未实现。

(5) 不及格　功能不完善，且程序错误较多，无法运行

2、 课程设计报告：

(1) 优　　包括设计内容，设计思想，已经完成的任务及达到的目标，

设计思路清晰、书写条理清楚，源程序结构合理、清晰，注

释说明完整，有对本次课程设计的心得体会。

(2) 良　　包括设计内容，设计思想，已经完成的任务及达到的目标，

设计思路基本清晰、书写条理基本清楚，源程序结构合理、

清晰，注释说明基本完整，有对本次课程设计的心得体会。

(3) 中　　课程设计报告内容基本完整，思路较清晰，书写基本清楚，

源程序结构尚可，有注释说明但不完整

(4) 及格　　课程设计报告内容基本完整，思路较差，书写尚清楚。

(5) 不及格　课程设计报告内容不完整，书写没有条理。

3、 综合实验演示回答教师提问：

(1) 优　　系统提交测试功能全面，能回答教师提出的所有问题，并完全正确，思路清晰

(2) 良　　系统提交完成基本功能，基本能回答教师提出的所有问题，有些小错误

(3) 中　　系统提交完成基本功能，基本能回答教师提出的问题，少数问题回答错误或不清楚

(4) 及格　　系统提交功能不全，能回答教师提出的问题，但较多问题回答错误或不能回答

(5) 不及格　系统提交功能不全，基本不能回答教师提出的问题

要求同学们独立完成，发现抄袭的，经查实后以 0 分计。

基于图像的手势识别

一、 设计目的：

1. 掌握机器学习项目方案设计的一般方法，主要包括系统分析、系统设计的组织和实施。

2. 掌握机器学习和图像数据处理方法和技术。

3. 培养把所学知识运用到具体实践对象，并能提出详细解决方案的能力。

二、 实验原理与步骤

2.1 实验流程

为实现基于图像的手势识别系统，拟采用深度学习方法对手势图像进行分类。实验分为数据准备、数据增强和预处理、模型配置、训练、推理和评估阶段。在数据准备阶段整理并标注手势图片，分为训练集、验证集和测试集。数据增强和预处理阶段应用一系列增强和标准化操作提升模型泛化能力。模型配置阶段以 MobileNetV3_large 轻量级卷积神经网络为基础，适合多类别手势图片识别。训练阶段设计训练、验证、早停和模型保存等机制。推理和评估阶段给出最终模型，对用户上传的图片进行类别预测并输出结果。

2.2 数据集、预处理

如图 1、图 2 所示，数据集基于 HaGRIDv2 项目，包括图片文件夹以及对应的标注 json 文件，分别存储于 hagrid_dataset 和 hagrid_annotations 目录内。实验最初针对 one、peace、three 三类手势，第一批次训练完成后，在第二批次加入 fist、four、palm、call 四类手势。

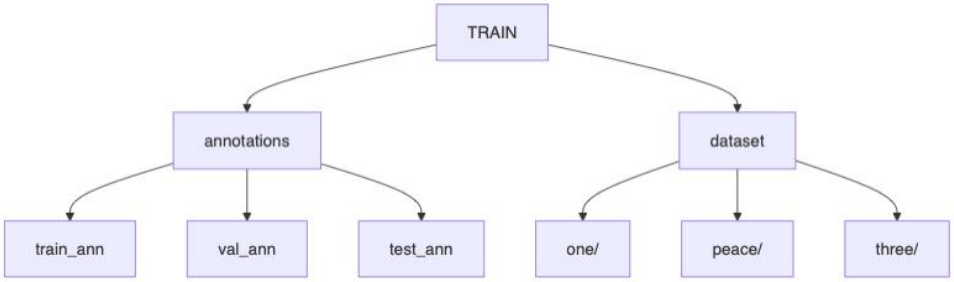

图 1 训练数据及其标注结构图

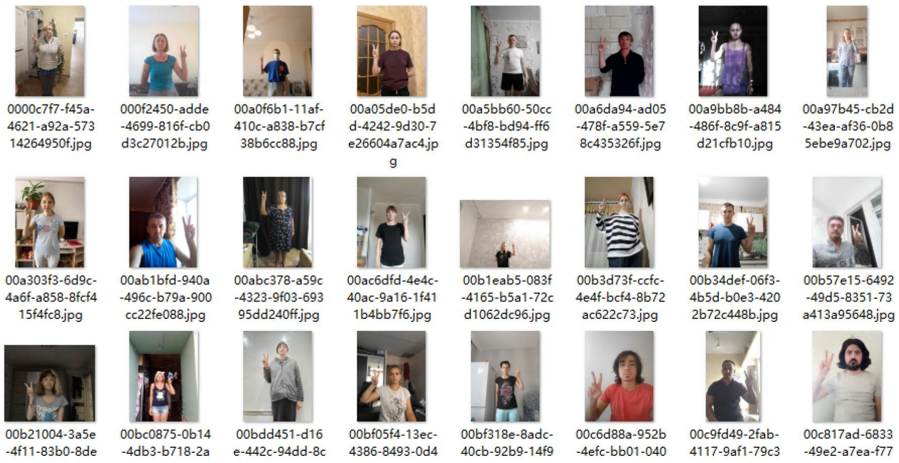

图 2　HaGRIDv2 训练数据

数据增强和预处理使用 albumentations 库，通过 train_transforms 、val_transforms 、 test_transforms 三组配置对图片进行操作，限定输入图片最大边，保证输入图片为正方形并填充指定像素，基于配置文件中的均值和标准差对图片归一化，将 ToTensorV2 转为 PyTorch 张量以进行模型训练。

2.3 训练原理

选用 MobileNetV3_large 作为特征提取网络，在分类头输出类别预测。采用交叉熵损失函数和随机梯度下降算法，不进行全梯度下降以跳出局部最优。带有 momentum 的随机梯度下降可缓解更新过程中的震荡，加速收敛。调度器使用 StepLR ，每隔 30 个 epoch 进行一次学习率衰减，让学习率按阶梯式逐步降低，利于提升模型泛化能力。

2.4 训练和验证流程

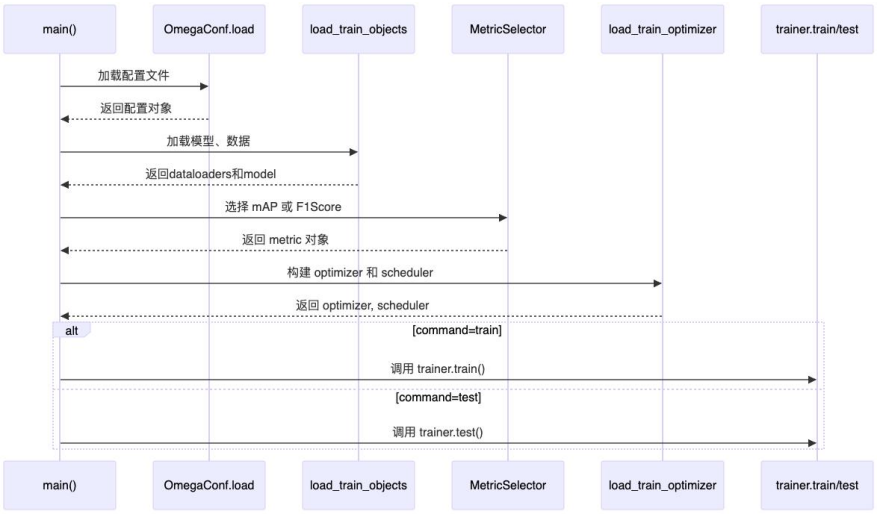

图 3　训练主控逻辑顺序图

如图 3 所示，训练主控逻辑在 run.py ，包括命令行参数解析、模型配置加载和训练流程调用，支持训练和测试两种模式。train_utils.py 实现了 Trainer 类，封装了训练、验证和测试过程。每次 epoch 训练遍历训练集所有 batch，计算损失并反向传播更新参数，定期在验证集上评估 F1Score 指标并执行早停策略，若连续 10 个 epoch 未提升则提前终止。达到指标提升时保存模型快照。

2.5 推理和部署

系统提供 predict_image_app.py ，结合 Streamlit 实现 Web 推理界面。用户可上传待识别图片，系统加载训练好的模型并即时输出推理结果，推理时统一采用和训练相同的数据增强流程。

三、 实验结果与分析

3.1 实验环境

实验环境和训练过程如表 1、图 4 所示。为下载 HaGRIDv2 数据集，本地磁盘需至少拥有 240GB 的空余磁盘空间。

配置项	说明
操作系统	Windows 11 Pro 24H2
中央处理器	Intel i9 9900KF
图形处理器	RTX 2080 Super
内存	96GB DDR4 3200MHz
Python 版本	3.11
CUDA 版本	2.18

表 1 实验环境

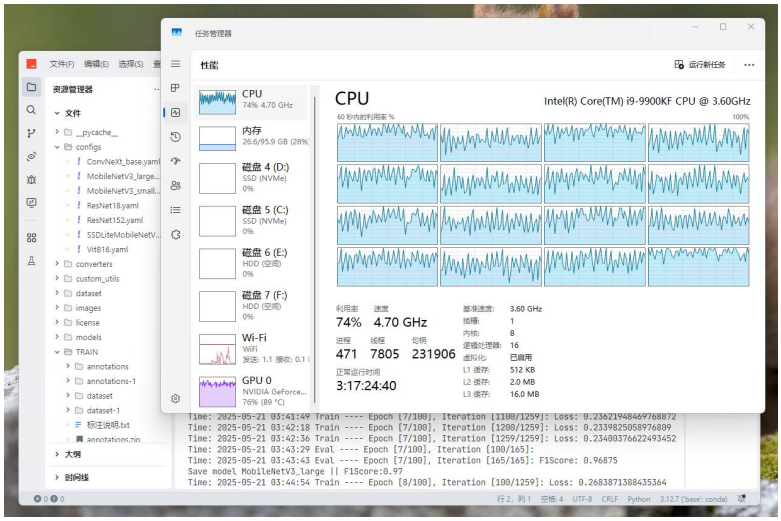

图 4 训练过程

3.2 训练结果与分析

实验过程中模型使用 MobileNetV3_large ， 设置训练轮数为 100， batch size 为 128, 并采用早停机制。训练过程中, 每隔 100 步记录一次损失值, 每个 epoch 均在验证集上评估 F1Score 指标。

```
Time: 2025-05-21 04:13:02 Train ---- Epoch [11/100], Iteration [900/1259]: Loss: 0.17206640872690412
Time: 2025-05-21 04:13:32 Train ---- Epoch [11/100], Iteration [1000/1259]: Loss: 0.17073336541652678
Time: 2025-05-21 04:14:03 Train ---- Epoch [11/100], Iteration [1100/1259]: Loss: 0.17067958414554596
Time: 2025-05-21 04:14:34 Train ---- Epoch [11/100], Iteration [1200/1259]: Loss: 0.17511191591620445
Time: 2025-05-21 04:14:52 Train ---- Epoch [11/100], Iteration [1259/1259]: Loss: 0.17911972220127398
Time: 2025-05-21 04:15:43 Eval ---- Epoch [11/100], Iteration [100/165]:
Time: 2025-05-21 04:15:57 Eval ---- Epoch [11/100], Iteration [165/165]: F1Score: 0.98046875
Save model MobileNetV3_large || F1Score:0.98
Time: 2025-05-21 04:17:08 Train ---- Epoch [12/100], Iteration [100/1259]: Loss: 0.2029094696044922
Time: 2025-05-21 04:17:38 Train ---- Epoch [12/100], Iteration [200/1259]: Loss: 0.21093734353780746
Time: 2025-05-21 04:18:07 Train ---- Epoch [12/100], Iteration [300/1259]: Loss: 0.18753769993782043
Time: 2025-05-21 04:18:38 Train ---- Epoch [12/100], Iteration [400/1259]: Loss: 0.19622059911489487
Time: 2025-05-21 04:19:08 Train ---- Epoch [12/100], Iteration [500/1259]: Loss: 0.19003020524978637
Time: 2025-05-21 04:19:38 Train ---- Epoch [12/100], Iteration [600/1259]: Loss: 0.21122594674428305
Time: 2025-05-21 04:20:08 Train ---- Epoch [12/100], Iteration [700/1259]: Loss: 0.21054815820285253
Time: 2025-05-21 04:20:39 Train ---- Epoch [12/100], Iteration [800/1259]: Loss: 0.21092259138822556
Time: 2025-05-21 04:21:09 Train ---- Epoch [12/100], Iteration [900/1259]: Loss: 0.20909776124689314
```

图 5 训练日志

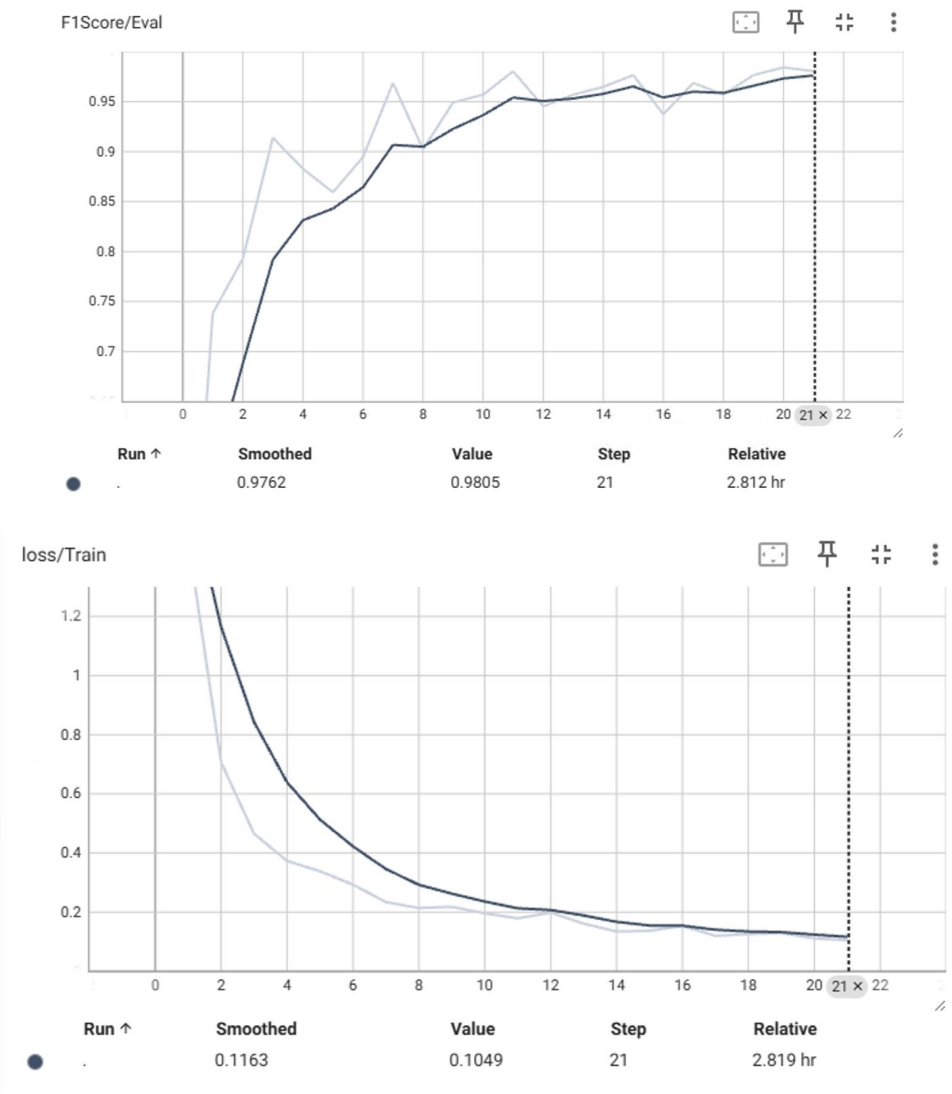

图 6　F1Score 和 loss 随训练轮数变化图

部分训练日志和过程如图 5、图 6 所示，模型的损失值持续下降，而 F1Score 随着 epoch 的增加快速上升并趋于稳定。例如，训练第 1~4 轮时情况如下：

第 1 轮末，Loss：约 1.943，验证集 F1Score：0.31
第 2 轮末，Loss：约 0.738，验证集 F1Score：0.74
第 3 轮末，Loss：约 0.706，验证集 F1Score：0.79
第 4 轮末，Loss：约 0.465，验证集 F1Score：0.91

随后几个 epoch，模型 F1Score 保持在 0.85~0.98，说明模型在识别手势图片上已经具备较高的准确率。损失值稳步下降，表明模型没有出现明显过拟合。在第 11 个 epoch 达到 F1Score 最高值 0.98 时程序自动保存了模型权重。该模型随后被用于最终推理和测试，能较为准确地对图片中的七类手势进行分类。

3.2 成果可视化

用户界面如图 7 所示。在实际 Web 演示中，用户可上传图片进行识别，模型

基本能准确输出类别，验证模型效果。

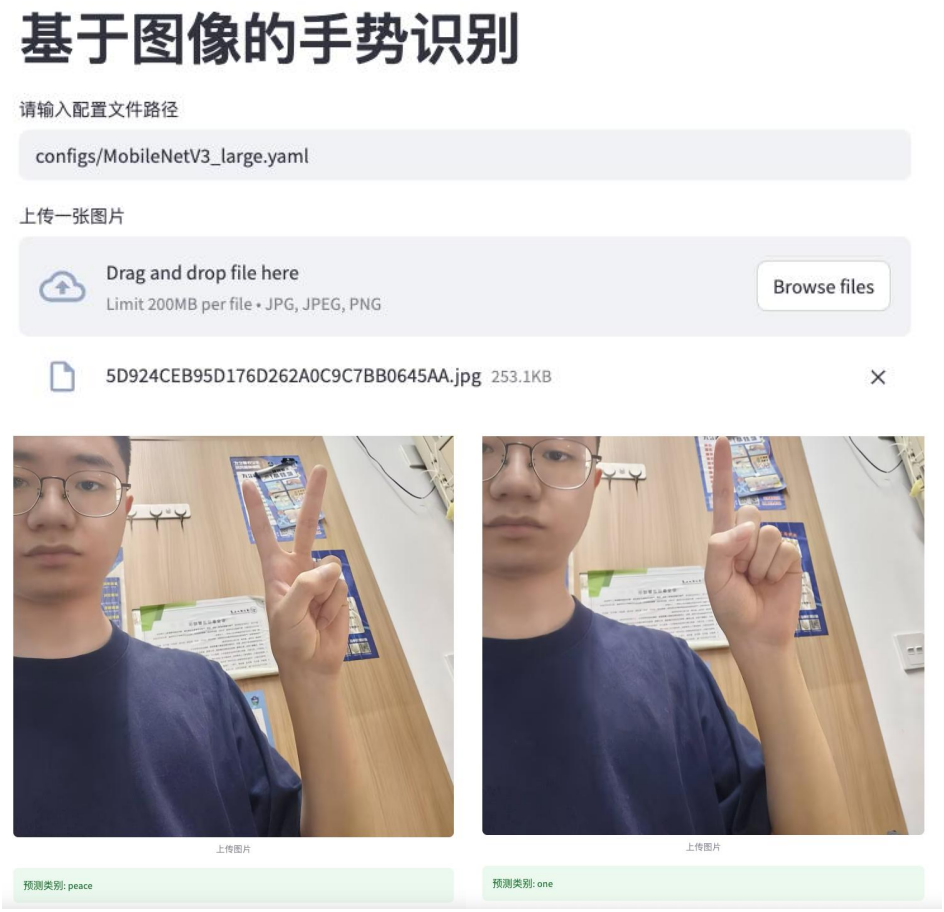

图 7 用户界面

四、 实验心得体会

在基于图像的手势识别实验中，我对机器学习项目的完整流程有了系统性的深刻体验。通过本项目，我切实掌握从数据采集与标注、数据增强到模型训练、结果评估到推理部署的开发流程。对数据增强、模型选择和训练过程诸如早停、学习率调度等策略的实际应用，有效加深我对理论知识和实际工程的理解。

寻找合适的实验数据集是一大难题。hagrid 项目提供海量的数据集内容，其精简版本也需 119.4 GB 磁盘空间。为此，我转移至远程服务器进行实验工作，花费数天时间下载数据集并探索效果较好的模型。受时间限制，我最初选用数据集中的 3 个典型类别进行训练，并在第一批训练完成后加入另外 4 个类别。测试结果表明 MobileNetV3_large 模型在准确率上优于 MobileNetV3_small，在训练难度上优于 ResNet18 和 ResNet152。

在实验中，面对海量图片和高并发负载，通过合理配置 batch size 和数据加

载大幅提升实验效率。实际操作 albumentations 图像增强库提升模型的泛化能力，让我体会到良好数据预处理对最终模型效果的重要性。使用 MobileNetV3_large 作为骨干网络，兼顾模型轻量性与准确率，为后续应用部署提供可能性。

五、 主要代码解释

5.1 训练和测试代码

训练主循环遍历每个 epoch，主进程判断是否需提前停止训练，若需要停止则跳出循环。每轮训练开始时，模型被设置为训练模式，记录当前 epoch。训练过程中，Logger 用于记录训练日志。每个 batch 迭代时，将梯度清零，图片和标签送 GPU，随后进行前向传播计算损失，反向传播更新参数。每次迭代的 loss 都会被记录。训练结束后，如有学习率调度器，则根据配置调整学习率。主进程会将本轮平均 loss 写入 TensorBoard。根据配置，训练过程中会周期性地进行验证和测试。

```
def train(self):
    if self.train_data is None:
        raise Exception("Cannot train without training data")
    for epoch in range(self.epochs_run, self.max_epoch):
        if self.gpu_id == 0: # 主进程判断 提前停止训练
            if self.early_stop():
                self.stop = True
                if self.n_gpu > 1:
                    torch.distributed.broadcast(self.stop, 0)
        if self.stop:
            break
        self.model.train() # 训练模式
        self.current_state["epoch"] = epoch # 记录当前 epoch
        if self.n_gpu > 1:

self.train_data.sampler.set_epoch(self.current_state["epoch"])
        with Logger("Train", self.max_epoch, len(self.train_data),
self.config.log_every, self.gpu_id) as logger:
            # 遍历每个 batch
            for iteration, (images, targets) in
enumerate(self.train_data):
                self.optimizer.zero_grad() # 梯度清零
                # 将图片和标签送 GPU
                images = list(image.to(self.gpu_id) for image in
```

```
images)
            targets = [{k: v.to(self.gpu_id) for k, v in t.items()}
for t in targets]
            # 前向传播计算损失
            loss = self.model(images, targets)
            loss.backward() # 反向传播
            self.optimizer.step() # 优化器更新参数
            if self.n_gpu > 1: # 多卡训练 同步 loss
                world_size = dist.get_world_size()
                dist.reduce(loss, dst=0, op=dist.ReduceOp.SUM)
                loss = loss / world_size
            logger.log_iteration(iteration + 1,
self.current_state["epoch"], loss.item())
        if self.scheduler is not None: # 学习率衰减
            if self.config.scheduler.name == "ReduceLROnPlateau":
                self.scheduler.step(self.current_state["loss"])
            else:
                self.scheduler.step()
        if self.gpu_id == 0: # 主进程记录 loss 到 tensorboard
            self.current_state["loss"] =
logger.loss_averager.value
            self.summary_writer.add_scalar(
                "loss/Train", self.current_state["loss"],
self.current_state["epoch"]
            )
      # 验证
      if self.config.eval_every > 0 and
self.current_state["epoch"] % self.config.eval_every == 0:
          self.val()
      # 测试
      if self.config.test_every > 0 and
self.current_state["epoch"] % self.config.test_every == 0:
          self.test()

def test(self):
   self.model.eval() # 评估模式
   if self.test_data is None:
       raise Exception("Cannot test without test data")
   with Logger("Test", self.max_epoch, len(self.test_data),
self.config.log_every, self.gpu_id) as logger:
      for iteration, (images, targets) in
enumerate(self.test_data):
          # 图片、标签送 GPU
          images = list(image.to(self.gpu_id) for image in images)
```

```
            targets = [{k: v.to(self.gpu_id) for k, v in t.items()} for
t in targets]
            with torch.no_grad(): # 不进行梯度计算
                output = self.model(images) # 前向推理
            # 计算当前 batch 指标
            metric = self.metric_calculator(output, targets)
            logger.log_iteration(iteration + 1,
self.current_state["epoch"], metrics=metric)
        if self.gpu_id == 0: # 主进程，写入 TensorBoard
            for key, value in metric.items():
                self.summary_writer.add_scalar(f"{key}/Test", value,
self.current_state["epoch"])
```

5.2 结果展示和用户界面代码

load_model(config_path) 方法用于加载指定路径的配置文件，根据配置构建模型和推理用的数据变换。如有模型权重 checkpoint，加载权重到 CPU ，恢复模型参数，将模型设置为评估模式并返回模型和变换。在 Streamlit 前端界面中，网页标题设置为“基于图像的手势识别”，用户可输入配置文件路径并上传图片。上传的图片被转为 RGB 并转为 numpy 数组，应用推理变换处理。推理时关闭梯度，仅做前向推理，模型输出类别概率，取概率最大值对应的类别索引，显示原图和预测类别。

```
def load_model(config_path):
    conf = OmegaConf.load(config_path) # 加载配置
    model = build_model(conf) # 构建模型
    # 构建推理变换
    transform = get_transform_for_inf(conf.test_transforms)
    if conf.model.checkpoint is not None:
        snapshot = torch.load(conf.model.checkpoint,
map_location=torch.device("cpu")) # 加载模型权重到 CPU
        # 加载模型参数
        model.load_state_dict(snapshot["MODEL_STATE"])
    model.eval() # 评估模式
    return model, transform

st.title("基于图像的手势识别")
config_path = st.text_input("请输入配置文件路径",
"configs/MobileNetV3_large.yaml")
if config_path:
    model, transform = load_model(config_path) # 加载模型和变换
    uploaded_file = st.file_uploader("上传一张图片", type=["jpg",
"jpeg", "png"])
```

```
if uploaded_file is not None:
    # 图片转 RGB
    image = Image.open(uploaded_file).convert("RGB")
    img_np = np.array(image) # 转 numpy
    # 应用变换处理
    processed = transform(image=img_np)["image"]
    with torch.no_grad(): # 禁用梯度，仅推理
        output = model([processed]) # 前向推理
    # output 标签最大概率的索引
    label = output["labels"].argmax(dim=1).item()
    st.image(image, caption="上传图片", use_column_width=True)
    st.success(f"预测类别: {targets[label]}")
```