

****

智能工程训练课程报告

|  |  |
| --- | --- |
| 题 目 | **RGB-T目标跟踪** |
| 学生姓名 | 陈梓萌 |
| 学 号 | 8207211521 |
| 指导教师 | 余伶俐 |
| 学 院 | 自动化学院 |
| 专业班级 | 智能2101 |

2024年10月

RGB-T目标跟踪

摘要

RGB-T目标跟踪融合了可见光(RGB)和热成像(Thermal)两种模式的数据，旨在克服单一模式下光照、遮挡、复杂背景等因素对目标跟踪的影响。相比于传统的单模态跟踪方法，RGB-T跟踪利用了RGB图像的细节信息和热成像的温度信息，提升了在恶劣条件下的跟踪鲁棒性和准确性。本文综述了RGB-T目标跟踪的研究进展，复现了一种新型的融合机制和特征提取方法APFNet，显著提高了跟踪效果，尤其在低光照和复杂背景下表现优异。实验结果表明，该方法在多个公开RGB-T数据集上取得了领先的跟踪性能。

**关键词：**RGB-T目标跟踪 可见光 热成像 多模态信息融合 APFNet

**RGB-T Target Tracking**

**ABSTRACT**

RGB-T target tracking integrates data from both visible light (RGB) and thermal imaging (Thermal) modalities, aiming to overcome challenges such as lighting variations, occlusion, and complex backgrounds in single-modality tracking. Compared to traditional single-modality methods, RGB-T tracking leverages the detailed information from RGB images and the temperature data from thermal imaging, enhancing tracking robustness and accuracy in adverse conditions. This paper reviews the progress of RGB-T target tracking research and replicates a novel fusion mechanism and feature extraction method, APFNet, which significantly improves tracking performance, especially under low-light and complex background conditions. Experimental results demonstrate that this method achieves state-of-the-art performance on multiple public RGB-T datasets.

**Key words**: RGB-T target tracking, visible light, thermal imaging, multimodal information fusion, APFNet

目录

第1章 绪论 ………………………………………………………………………………….1

1.1 研究背景………………………………………………………………….… …………1

1.2 研究现状 ………………………………………………………………………………1

1.2.1 基于非深度学习的传统RGB-T目标跟踪技术…………………………………….1

1.2.2 基于深度学习的RGB-T目标跟踪技术…………………………………………….2

1.3 研究目标………………………………………………………………………………..2

第2章 方案论述与框架设计 …………………………………………………..…4

2.1 方案论述 ………………………………………………………………………………4

2.2 框架设计 ………………………………………………………………………………5

第3章 程序设计……………………………………………………………………6

3.1 数据预处理过程 ………………………………………………………………………6

3.2 第一阶段训练过程 ……………………………………………………………………6

3.3 第二阶段训练过程 ……………………………………………………………………8

3.4 第三阶段训练过程 ………………………………………………................................9

第4章 测试方案与测试结果 …………………………………………………..…12

4.1 测试方案……………………………………………………………………12

4.1.1 测试与训练数据集介绍……………………………………………..12

4.1.2 测试方案……………………………………………………………..13

4.2 测试结果……………………………………………………………………13

4.2.1 GTOT训练，RGBT234测试………………………………………..13

4.2.2 RGBT234训练，GTOT测试………………………………………...14

4.3 遇到的问题以及解决……………………………………………………….15

第5章 总结与体会………………………………………………………………….19

参考文献 …………………………………………………………………………….20

1. 绪论

1.1 研究背景

目标跟踪任务以视频为输入，要求模型识别每帧中的目标物体，并将前后帧中的相同物体进行关联，赋予唯一的TrackID，且该ID在整个序列中保持不变。所有出现的目标必须能够被及时追踪，并且目标的位置应尽可能与真实位置高度吻合。

近年来，基于RGB图像的目标跟踪技术取得了诸多研究突破。然而，单模态的RGB图像目标跟踪在实际应用中面临较大挑战。在光照变化、阴影、遮挡等复杂场景下，尤其是夜晚、雨雪等恶劣天气条件下，RGB图像的信息可能受到严重干扰，导致跟踪性能显著下降。

RGB-T目标跟踪（RGB-Thermal，结合可见光与热红外）旨在通过融合RGB图像和热红外图像的信息，实现对目标物体的持续、精准跟踪。RGB图像提供丰富的颜色和纹理等视觉信息，适合在光照良好的环境中进行目标识别与跟踪。而热红外图像则能够捕捉物体的热辐射信息，对光照条件不敏感，因此在夜间或光线不足的环境下表现尤为出色。

通过融合RGB和热红外两种模态的信息，RGB-T目标跟踪技术能够有效弥补单一模态的不足，提升在复杂场景下的跟踪稳定性和准确性。这项技术广泛应用于视频监控、自动驾驶、无人机跟踪等领域，对于提升这些系统的智能化和实用性具有重要意义。

1.2 研究现状

1.2.1 基于非深度学习的传统RGB-T目标跟踪技术

传统的基于非深度学习的RGB-T目标跟踪技术包括基于手工特征的方法、基于稀疏表示的方法、以及基于滤波器的方法等[1]。Han Xu等[2]提出的DRF模型，通过对信息源图像进行分解，引入了图像融合的解纠缠表示方法，缓解了单一信息提取不当的问题。该模型首先利用手工特征方法融合不同类型的表示，再将融合后的表示输入预训练的生成器以生成融合结果。尽管该模型在后续使用了基于深度学习的生成器，但在输入生成器之前，仍采用了手工特征来融合RGB和红外图像模态。然而，这种基于手工特征的方式存在特征表示能力不足的问题，跟踪性能也受到一定限制。

Lin Li等[3]研究了基于稀疏表示的可见光和红外图像融合的鲁棒目标跟踪方法。基于稀疏表示的跟踪器在面对噪声和误差时表现良好，然而，由于该方法使用像素强度作为特征表示，在复杂场景下的鲁棒性较弱。

Chengwei Luo等[4]提出了一种基于滤波器的鲁棒目标跟踪方法，该方法包括基于相关滤波器的跟踪（CFT）模块和基于直方图的跟踪（HIST）模块。相关滤波方法通过对滤波模板和搜索区域进行相关操作生成响应图，然后根据峰值确定目标位置。基于相关滤波器的跟踪器在精度和速度之间实现了良好的平衡，但在复杂环境下的特征表示能力仍然有限。

1.2.2 基于深度学习的RGB-T目标跟踪技术

基于深度学习的目标跟踪器能够通过训练获得更加鲁棒的特征表示，从而显著提升跟踪性能。然而，在执行RGB-T目标跟踪任务时，深度学习通常需要设计复杂的网络结构，并依赖大量数据进行训练。Lichao Zhang等[5]基于DiMP跟踪框架设计了多种融合策略，利用了一个包含9,335个视频、总计1,403,359帧的庞大训练数据集。该模型分析了多模态融合的有效性，探讨了像素级、特征级和响应级等不同融合机制。然而，这类大规模模型在训练和运行效率方面仍然存在局限，且其跟踪性能有待进一步提升。

Transformer[6]以注意力机制为核心，最初用于自然语言处理领域，近年来也在视觉跟踪领域取得了突破性进展。Xin Chen等[7]在孪生框架中设计了一个基于自我注意的增强模块和一个基于交叉注意的特征融合模块，旨在捕捉全局信息。一方面，基于孪生网络的框架将目标跟踪视为搜索区域与模板区域之间的相似性度量问题，从而实现了实时处理速度；另一方面，Transformer通过其自注意机制捕获序列中的长距离依赖关系，具有比传统CNN更大的感受野。

在RGB-T目标跟踪中，还需要应对热交叉、光照变化、尺度变化、遮挡和快速运动等挑战。Yun Xiao等[8]开发了APFNet，针对这五类挑战属性设计了五个分支，并提出了一个融合结构，将各分支信息有效结合。实验表明，该模型充分利用了多模态特征之间的关联性，在RGB-T目标跟踪任务中表现出色。

1.3 研究目标

1. 研究如何有效融合RGB和热成像数据，充分利用两者的互补特性，提升目标跟踪的准确性和鲁棒性，特别是在光照变化、遮挡和复杂背景下的跟踪性能。
2. 在保持跟踪精度的同时，优化网络结构，减少模型复杂度，提高跟踪器的实时性和运行效率，满足实际应用中的实时性需求。
3. 针对RGB-T目标跟踪中的热交叉、光照变化、遮挡、尺度变化和快速运动等常见问题，通过针对性解决方案，提高模型的鲁棒性。

第2章 方案论述与框架设计

2.1 方案论述

Yun Xiao等人 [8]提出了一种针对RGB-T目标跟踪的属性渐进融合网络（APFNet），其主要创新点在于通过将融合过程与五种常见的跟踪挑战属性（热交叉、光照变化、尺度变化、遮挡、快速运动）相解耦，从而提升了多模态信息的融合能力，并减少了对大规模训练数据的依赖。

首先，文章提出了将RGB和热红外（T）数据的融合过程根据不同的挑战属性进行解耦。与现有方法不同，该方案不再试图通过一个复杂的模型同时解决所有挑战，而是为每个挑战属性设计独立的融合分支。通过这种解耦方式，APFNet可以在每个挑战属性下高效地学习融合参数，减少模型参数数量，从而在小规模数据集上也能取得良好效果。

其次，针对RGB-T跟踪中常见的五个挑战（热交叉、光照变化、尺度变化、遮挡、快速运动），APFNet为每个挑战设计了独立的融合分支，利用小规模数据集单独训练这些分支。这些分支只需关注在特定属性下进行特征融合，因此能以较少的参数实现高效的多模态信息融合。

由于测试阶段的输入帧中无法直接得知具体属性，文章设计了一个基于SKNet的属性聚合模块，能够自适应地从所有融合分支中选择最有效的特征，并抑制未出现的属性带来的噪声特征。这种聚合机制可以根据不同场景动态选择融合特征，提升了模型的泛化能力。

此外，文章提出了一个增强融合Transformer模块，通过三编码器和两解码器结构，对聚合后的融合特征和模态特定特征进行自增强和交互增强。这种设计增强了不同特征间的信息交换，使得最终的跟踪表现更加鲁棒。

最后，文章提出了一个三阶段的训练方案，包括单独训练每个属性特定的融合分支、训练属性聚合模块、最后训练增强融合Transformer和微调其他模块。通过这种逐步训练的方式，确保了每个阶段都能有效学习特定的融合特征并逐步增强模型的整体性能。

2.2 框架设计

APFNet基于VGG-M模型的前3层作为基础网络，并将其扩展为双流架构，用于分别提取RGB和热红外图像的模态特定特征。在每一层中，嵌入了属性渐进融合（APF）模块，以逐层融合两种模态的信息。对于每个属性特定的融合分支，首先从RGB和T模态中提取特征，然后通过卷积层和ReLU激活函数进行处理。接下来，使用SKNet自适应地选择通道级特征，确保只保留对当前属性有用的信息。每个分支的结构相同，以确保统一的特征处理方式。在每个属性特定的融合分支处理完后，将融合特征送入聚合模块。该模块基于SKNet的注意力机制，自适应地为不同属性的融合分支分配权重，从而生成更加鲁棒的聚合特征。

APFNet的增强融合Transformer包括三编码器和两解码器。编码器分别对聚合特征和模态特定特征进行自增强，而解码器则通过交互增强进一步提升融合特征的表达能力。这种设计能够捕捉长距离的特征依赖关系，并有效整合不同模态的信息。

训练过程分为三个阶段，分别训练属性特定分支、聚合模块和增强模块。在在线跟踪时，为每个新视频序列随机初始化全连接层，并根据视频的目标样本进行微调。APFNet在每10帧内动态更新，以确保在跟踪过程中适应外观变化。

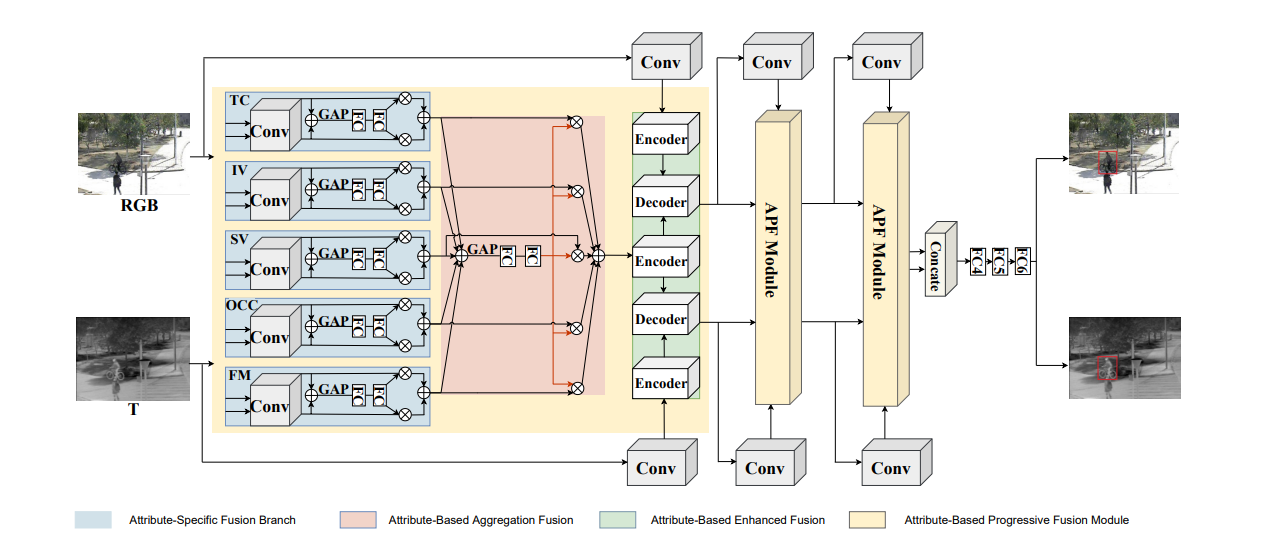


图2-1 APFNet的结构。TC、IV、SV、OCC 和 FM 分别代表五个挑战属性特定的分支，包括热交叉（thermal crossover）、光照变化（illumination variation）、尺度变化（scale variation）、遮挡（occlusion）和快速运动（fast movement）。其中，"+" 和 "×" 分别表示元素级的加法和乘法操作。GAP 指的是全局平均池化（global average pooling）。

第3章 程序设计

* 1. 数据预处理过程

首先，我们需要针对RGB-T目标跟踪任务，从GTOT或RGBT234数据集中加载和预处理图像和标注数据，并根据指定的挑战属性（如快速运动、遮挡、尺度变化、光照变化和热交叉）筛选相关的图像和标签数据，最终生成预训练数据路径pkl文件。

我们通过set\_type变量确定使用的数据集是GTOT还是RGBT234，并根据数据集设置相关路径。challenge\_type变量用于指定要处理的数据挑战类型（如快速运动、遮挡等）。可以选择特定挑战类型或处理整个数据集（ALL）。此外，使用seqlist\_path路径加载对应数据集的序列列表。每个序列对应一个跟踪任务，其中包含RGB（可见光）和T（红外）图像。

接下来，我们可以针对不同的挑战属性构建数据集pkl文件，代码根据序列列表遍历每个序列，并读取RGB和红外图像路径，以及目标的标注文件（init.txt）。最后需要检查RGB图像、红外图像和标注的数量是否一致，确保数据完整性。

处理完所有序列后，代码将构建的数据集以pickle格式（.pkl文件）保存，供后续模型训练或预处理使用。

* 1. 第一阶段训练过程

接下来，我们需要按照模型的前向传播顺序，针对不同的挑战属性分支进行训练。我们首先创建MDNet模型的实例，使用指定的预训练模型路径init\_model\_path进行初始化。

model = MDNet(opts['init\_model\_path'], K)

随后，定义二元交叉熵损失函数（BCELoss），并调用set\_optimizer函数设置优化器，以便在训练过程中更新模型参数。

criterion = BCELoss()

optimizer = set\_optimizer(model, opts['lr'], opts['lr\_mult'])

接下来。进入训练循环，根据预设的周期数（n\_cycles），进行模型训练。每个周期中随机打乱序列顺序（k\_list），以确保训练的随机性。

for i in range(opts['n\_cycles']):

# Training

model.train()

prec = np.zeros(K)

k\_list = np.random.permutation(K)

我们将针对不同的挑战属性进行逐序列训练在每个序列的训练过程中，提取正样本和负样本区域，并传入模型进行前向传播，计算正负样本的分数。

pos\_regions\_v, neg\_regions\_v, pos\_regions\_i, neg\_regions\_i = dataset[k].next()

pos\_score = model(pos\_regions\_v, pos\_regions\_i, k)

neg\_score = model(neg\_regions\_v, neg\_regions\_i, k)

最后，计算损失，通过反向传播更新模型参数。使用optimizer.step()来应用梯度更新。

loss = criterion(pos\_score, neg\_score)

loss.backward()

optimizer.step()

在训练过程中，如果当前的精度超过了最佳精度，则保存当前模型的状态。这确保了最优模型可以在后续使用。也就是说，我们保存下来了最优模型。

if cur\_score > best\_score:

best\_score = cur\_score

torch.save(states, opts['model\_path'])

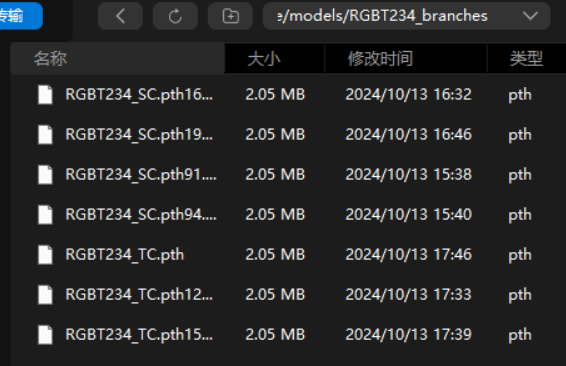
通过第一阶段的训练，我们将获得5个不同的挑战属性分支训练权重，以及一个ALL种类权重，这些文件将进一步用于第二阶段的训练。

图3-1 第一阶段训练所得的权重文件

* 1. 第二阶段训练过程

在第二阶段训练之前，我们需要在AAAICode/modules/model\_stage2.py文件中修改第一阶段分支权重文件路径，如图3所示：

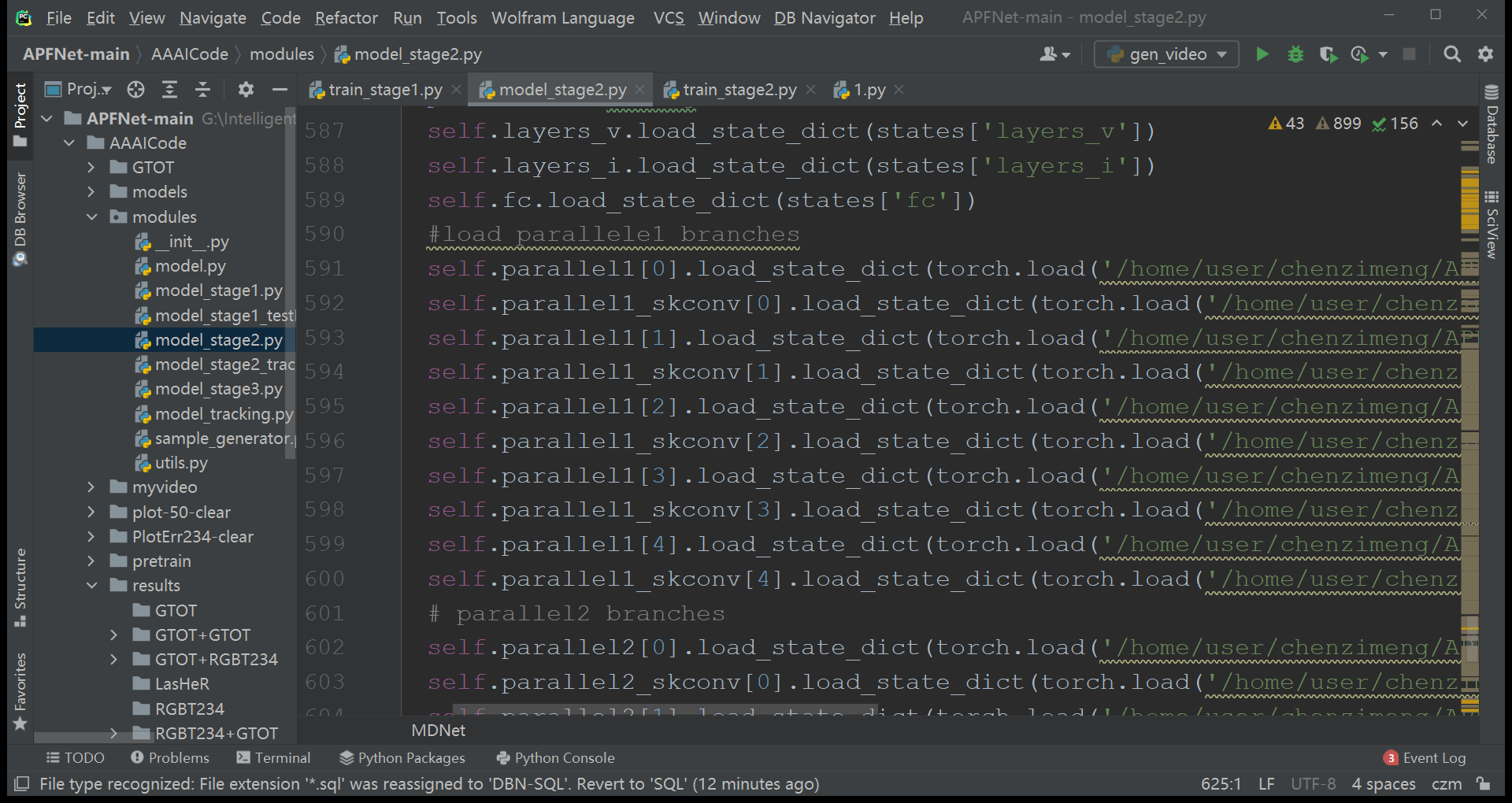


图3-2 model\_stage2.py文件路径修改示意

随后我们就可以在train\_stage2.py中进行训练了，其中的代码设计逻辑如下：

if opts['set\_type'] == 'RGBT234\_ALL':

img\_home = './RGBT234/'

data\_path = './pretrain/data/RGBT234\_ALL.pkl'

首先根据传入的选项set\_type，设置图像目录和数据路径。不同的类型会加载不同的数据集（如RGBT234\_FM、GTOT\_ALL等）。

with open(data\_path, 'rb') as fp:

data = pickle.load(fp)

K = len(data)

使用pickle从指定的.pkl文件中加载预处理的数据集，存储到data变量中，并获取数据集的大小K。

model = MDNet(opts['init\_model\_path'], K)

if opts['use\_gpu']:

model = model.cuda()

创建MDNet模型实例，加载指定的预训练模型路径。如果使用GPU，模型将被移动到GPU上。

criterion = BCELoss()

optimizer = set\_optimizer(model, opts['lr'], opts['lr\_mult'])

定义损失函数为二元交叉熵（BCELoss），并调用set\_optimizer函数为模型设置优化器，指定学习率。

for i in range(opts['n\_cycles']):

print('==== Start Cycle {:d}/{:d} ===='.format(i + 1, opts['n\_cycles']))

进入训练循环，根据预设的周期数n\_cycles进行模型训练。每个周期中，先打印当前周期的编号。

pos\_regions\_v, neg\_regions\_v, pos\_regions\_i, neg\_regions\_i = dataset[k].next()

if opts['use\_gpu']:

pos\_regions\_v = pos\_regions\_v.cuda()

在每个序列的训练过程中，提取正样本和负样本区域，并根据是否使用GPU将其移动到GPU上。

loss = criterion(pos\_score, neg\_score)

loss.backward()

optimizer.step()

计算损失，通过反向传播更新模型参数。如果设置了梯度裁剪，则在更新前对梯度进行裁剪，避免梯度爆炸。

if cur\_score > best\_score:

best\_score = cur\_score

torch.save(states, opts['model\_path']+'.pth')

在训练过程中，如果当前的精度超过了最佳精度，则保存当前模型的状态，确保最佳模型可以在后续使用。

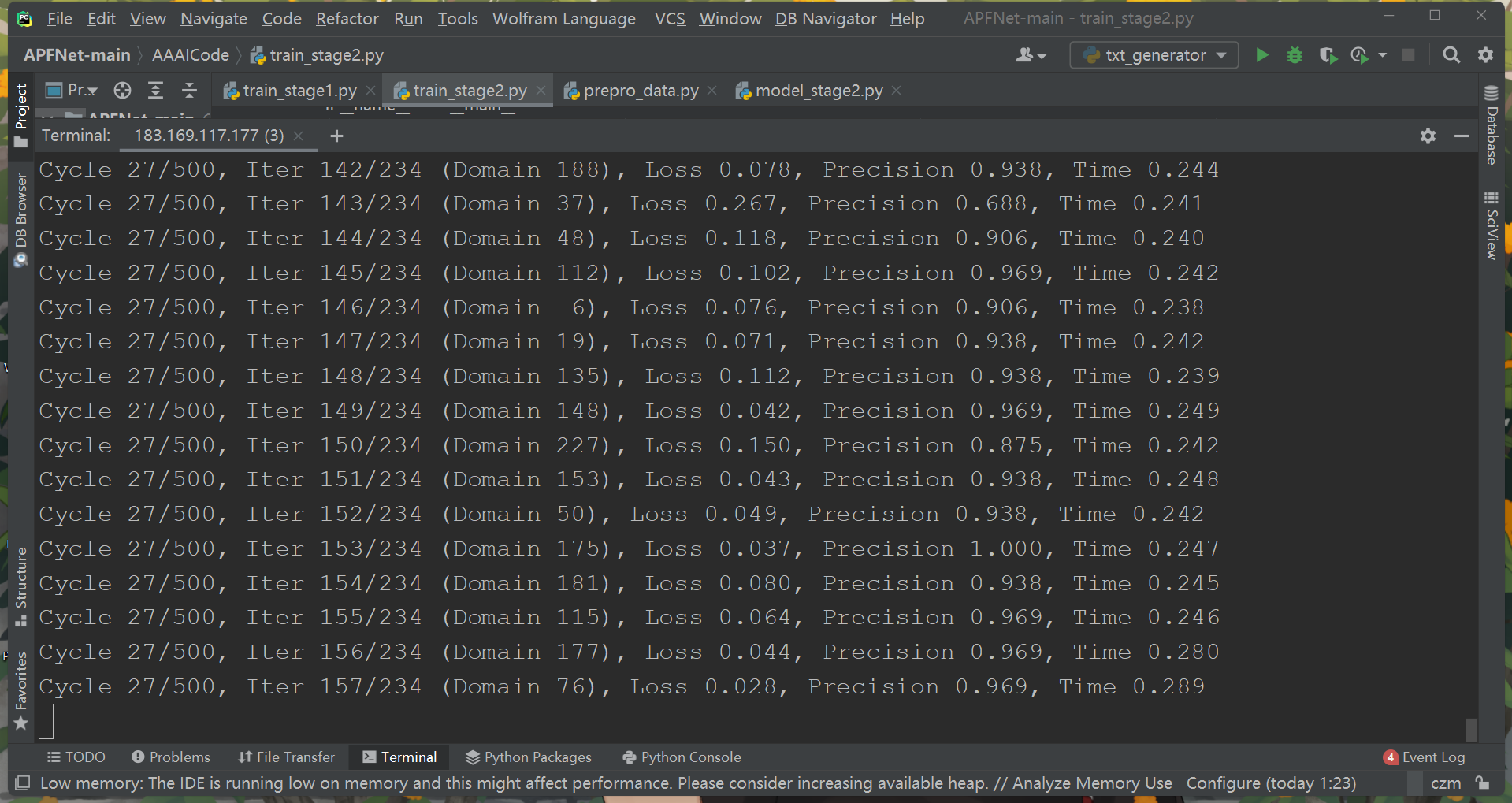


图3-3 第二阶段训练过程示意

* 1. 第三阶段训练过程

在完成第二阶段的训练之后，我们会得到融合权重文件，如图4所示。

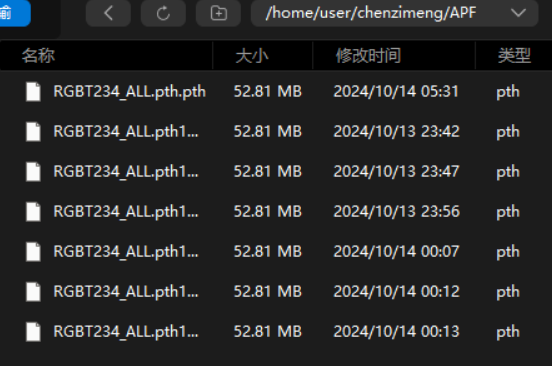


图3-4 完成第二阶段训练之后所得的融合权重文件

最后，我们将利用所得的融合权重文件，继续进行第三阶段的训练，有关train\_stage3.py的设计逻辑如下：

if opts['set\_type'] == 'RGBT234\_ALL':

img\_home = './RGBT234/'

data\_path = './pretrain/data/RGBT234\_ALL.pkl'

首先，我们需要初始化数据集，根据传入的选项set\_type设置图像文件的目录和数据路径。通过判断不同的类型（如RGBT234\_ALL、GTOT\_ALL等），选择相应的数据集。

with open(data\_path, 'rb') as fp:

data = pickle.load(fp)

使用pickle模块加载指定的.pkl文件，读取预处理的数据集，并将其存储在变量data中。

model = MDNet(opts['init\_model\_path'], K)

if opts['use\_gpu']:

model = model.cuda()

创建MDNet模型的实例，使用指定的预训练模型路径进行初始化。如果使用GPU，模型会被移动到GPU上。

criterion = BCELoss()

optimizer = set\_optimizer(model, opts['lr'], opts['lr\_mult'])

定义损失函数为二元交叉熵（BCELoss），并调用set\_optimizer函数为模型设置优化器，指定学习率和学习率调整倍数。

for i in range(opts['n\_cycles']):

print('==== Start Cycle {:d}/{:d} ===='.format(i + 1, opts['n\_cycles']))

进入训练循环，根据设定的周期数n\_cycles进行模型训练。在每个周期开始时打印当前周期的编号。

pos\_regions\_v, neg\_regions\_v, pos\_regions\_i, neg\_regions\_i = dataset[k].next()

在每个序列的训练过程中，从数据集中提取正样本和负样本区域，随后将这些区域传入模型进行训练。

loss = criterion(pos\_score, neg\_score)

loss.backward()

optimizer.step()

计算损失，通过反向传播更新模型参数。如果设置了梯度裁剪，则在更新之前对梯度进行裁剪，以防止梯度爆炸。

if cur\_score > best\_score:

best\_score = cur\_score

torch.save(states, opts['model\_path']+'.pth')

在训练过程中，如果当前的精度超过了最佳精度，则保存当前模型的状态，以确保最佳模型可用于后续使用。

第4章 测试方案与测试结果

4.1 测试方案

4.1.1 测试与训练数据集介绍

GTOT数据集：出自2016 IEEE Transactions on Image Processing (T-IP)的Learning Collaborative Sparse Representation for Grayscale-thermal Tracking 文章，安徽大学李成龙课题组。数据集包括带统计偏差的50个视频对（以灰度图片-热红外图片对的序列）、视频对应的每帧GroundTruth注释、两个评估指标。视频对是对齐的灰度图片和热红外图。视频拍摄场景有16个，包括实验室、校园道路、操场、水池等等。

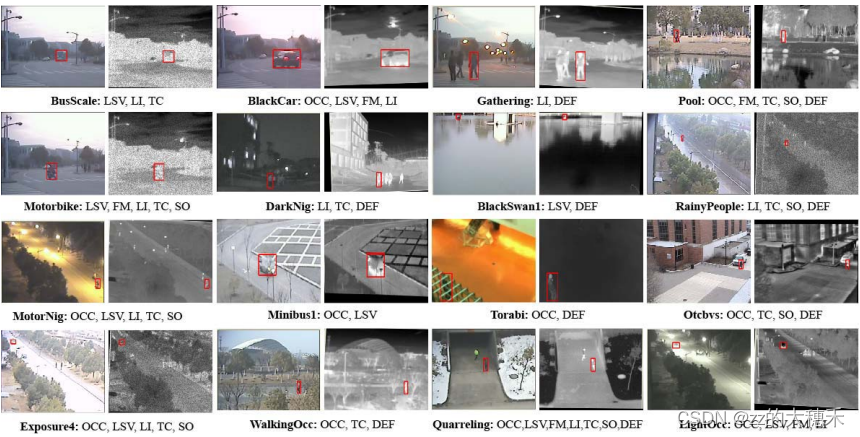


图4-1 GTOT数据集

GTOT数据集涉及的评价指标有：

Precision Score 预测框与真值框之间的中心位置距离。  
Success plot 预测框与真值框之间的交并比。AUC表示Success plot的曲线下距离。

RGBT234数据集：出自2019 Pattern Recognition (PR)的RGB-T Object Tracking: Benchmark and Baseline. 文章，安徽大学李成龙课题组。数据集包括234个RGB-T视频序列对及其对应真值GroundTruth。视频序列标注中有12个属性。总的帧数为234K，最长的视频序列有8K。标注序列第一帧的真值格式(x,y,box-width,box-height)。另外，一共有5个评估指标，其中包括了VOT中的评估指标, 具体如下：

* Maximum Precision Rate (MPR)
* Maximum Success Rate (MSR)
* Accuracy：衡量预测为真的这些帧的平均IOU。
* Robustness：衡量追踪器追到（预测框与真值框之间IOU大于阈值）的频率。
* Expected Average Overlap (EAO)：期望平均IOU结合了追踪器的准确度与鲁棒性两个评估指标。



图4-2 RGBT234数据集

4.1.2 测试方案

用于训练模型的硬件包括 Intel(R) Xeon(R) Platinum 8336C CPU @ 2.30GHz 和 NVIDIA GeForce RTX 4090 GPU。所使用的编程环境为 Ubuntu 22.04.4 LTS、Python 3.9 和 CUDA 11.8。在训练过程中，学习率为 0.0001，batch\_frames 为 8。模型在第一阶段、第二阶段和第三阶段的训练周期分别为 200 周期、500 周期和 1000 周期。我们采用的测试方案为：用RGBT234和 GTOT相互训练测试。

4.2 测试结果

4.2.1 GTOT训练，RGBT234测试

我们首先利用GTOT数据集做训练，RGBT234数据集做测试，最后生成的Success和Prercision图如图7和图8所示:

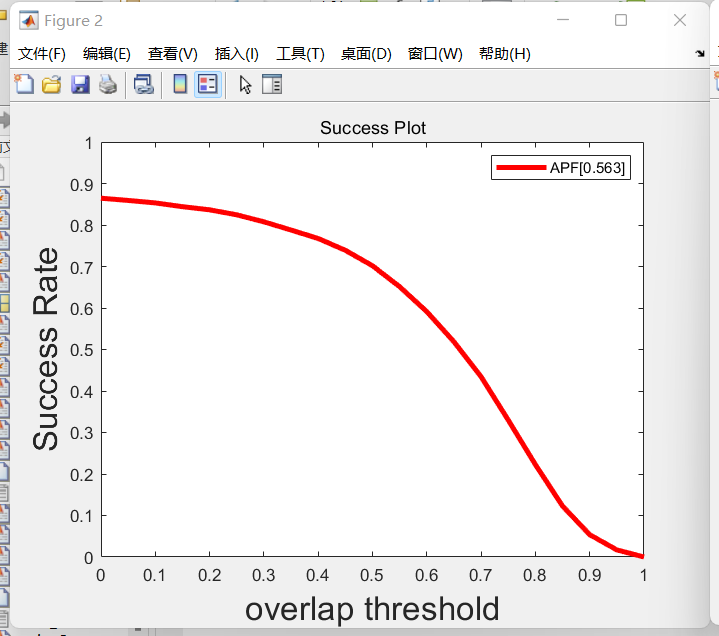


图4-3 Success图

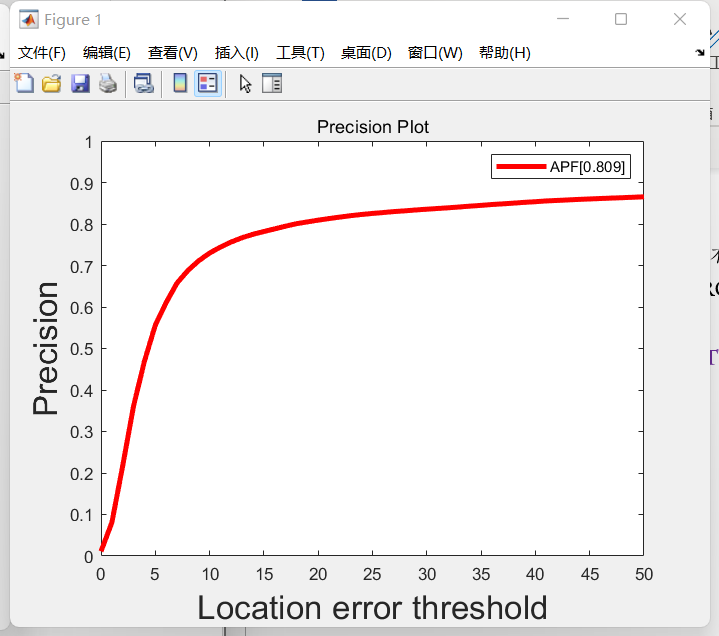


图4-4 Precision图

另外，在RGBT234数据集中的实际检测结果如图9所示：

图4-5 检测结果图（bikemove1）

4.2.2 RGBT234训练，GTOT测试

我们利用RGBT234数据集做训练，GTOT数据集做测试，最后生成的Success和Prercision图如图10和图11所示:

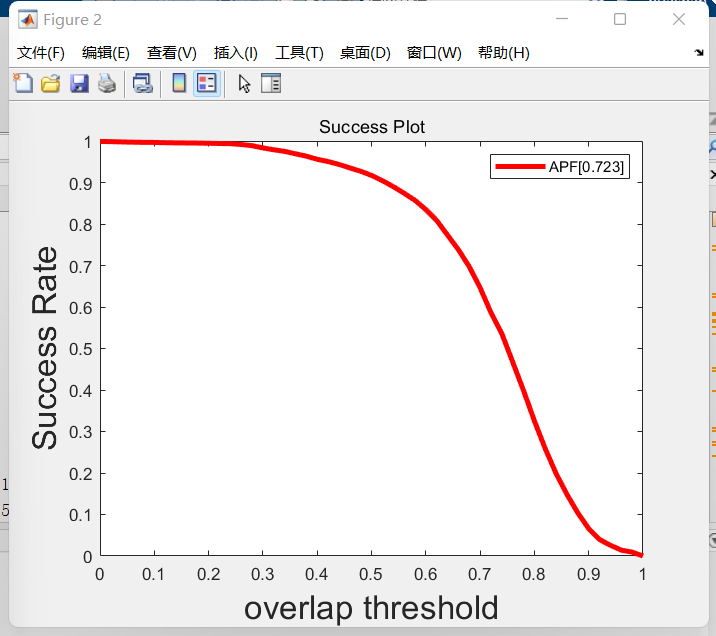


图4-6 Success图

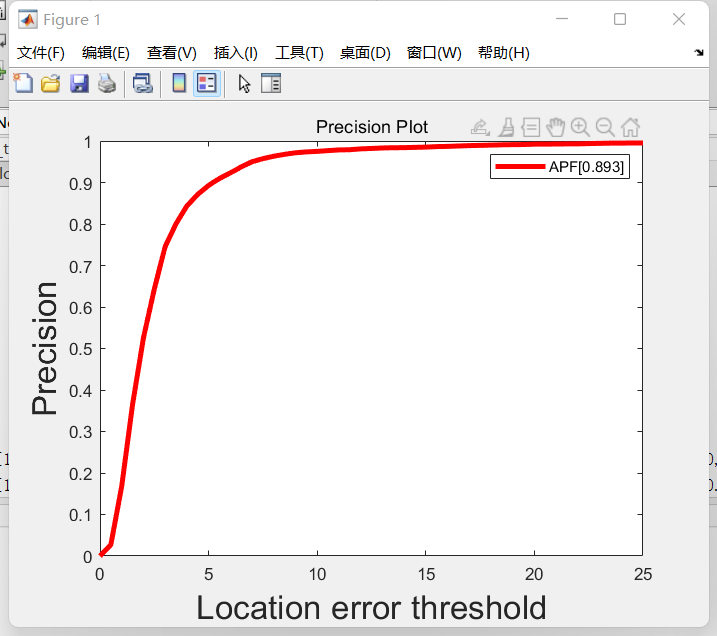


图4-7 Precision图

另外，在GTOT数据集中的实际检测结果如图12所示：



图4-8 检测结果图（fastCar2）

4.3 遇到的问题以及解决

首先，我们在利用GTOT做训练，测试RGBT234的实验中，遇到了一系列问题，列举如下：

1 Run过程中报错 UnboundLocalError: local variable ‘gt’ referenced before assignment

这个错误提示说明在 gen\_config 函数中，变量 gt 在被引用之前没有被赋值。这通常是因为在函数中某些条件下 gt 没有被正确初始化。通过检查gen\_config 函数，我发现必须将RGB\_T234数据集命名为RGBT234才可以被该函数识别，因此解决方式为，重命名RGB\_T234数据集文件夹为RGBT234；因此，如果遇到缺少Result文件夹报错，使用以下命令新建结果文件夹：

mkdir -p ./Results/RGBT234+GTOT/RGBT234/GTOT\_ALL\_Transformer/

2 prepro\_data.py运行报错：

Traceback (most recent call last):

File "./pretrain/data/prepro\_data.py", line 35, in <module>

assert len(img\_list\_v) == len(gt) == len(img\_list\_i), "Lengths do not match!!"

AssertionError: Lengths do not match!!

出现这个 AssertionError: Lengths do not match!! 错误意味着在 GTOT 或 RGBT234 数据集中，img\_list\_v（可见光图像列表）、img\_list\_i（红外图像列表）和 gt（ground truth，标注文件）的长度不一致。因此需要检查文件夹结构和文件数量是否匹配，即确保每个序列文件夹下的可见光（v 或 visible）、红外（i 或 infrared）图像数量与 init.txt 中的标注文件一致。通过排查GTOT数据集文件，发现在OccCar-1文件夹下的的35号、36号、37号、38号、42号文件出现重复，可能是由于百度网盘下载时出现的问题，而导致图片数量和标签数量不匹配。删除掉多余文件后，可以运行prepro\_data.py文件了。

3 train\_stage3.py文件报错：

Traceback (most recent call last):

File "./modules/model\_stage3.py", line 1353, in load\_model

self.parallel1.load\_state\_dict(states['parallel1'])

KeyError: 'parallel1'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):

File "train\_stage3.py", line 219, in <module>

train\_mdnet(opts)

File "train\_stage3.py", line 97, in train\_mdnet

model = MDNet(opts['init\_model\_path'], K)

File "./modules/model\_stage3.py", line 356, in \_\_init\_\_

self.load\_model(model\_path)

File "./modules/model\_stage3.py", line 1366, in load\_model

shared\_layers = states['shared\_layers']

KeyError: 'shared\_layers'

错误分析：

1. KeyError: 'parallel1'： 当程序试图加载模型状态时，期望从 states 中找到 parallel1 这个键（即 states['parallel1']），但在加载的模型文件中找不到对应的键。这可能意味着：

o 模型文件保存时没有包含 parallel1 的信息。

o 加载的模型与当前的模型架构不匹配。

2. KeyError: 'shared\_layers'： 程序在处理异常时，尝试从 states 中获取 shared\_layers，但是在模型状态文件中也找不到这个键。这同样可能是由于保存的模型文件不包含该部分，或者是模型架构发生了变化。

可能原因：

• 模型架构不匹配：训练模型时和加载模型时的架构不一致，导致模型文件中没有保存或不支持当前需要的层（如 parallel1 和 shared\_layers）。

• 模型文件不完整或错误：保存的模型文件可能不完整，或者是在某个阶段中只保存了部分网络权重，而非完整的模型结构。

因此，我分析在train\_stage3.py文件中命令行参数部分的设置，发现了以下问题：在-init\_model\_path那一栏，应该填入第二阶段生成的训练模型（only the model parameters generated by the second stage need to be loaded），而不是预训练权重文件，另外需要注意，我这里文件的名称是GTOT\_ALL.pth.pth。修改之后，可以进行第三阶段的训练了。

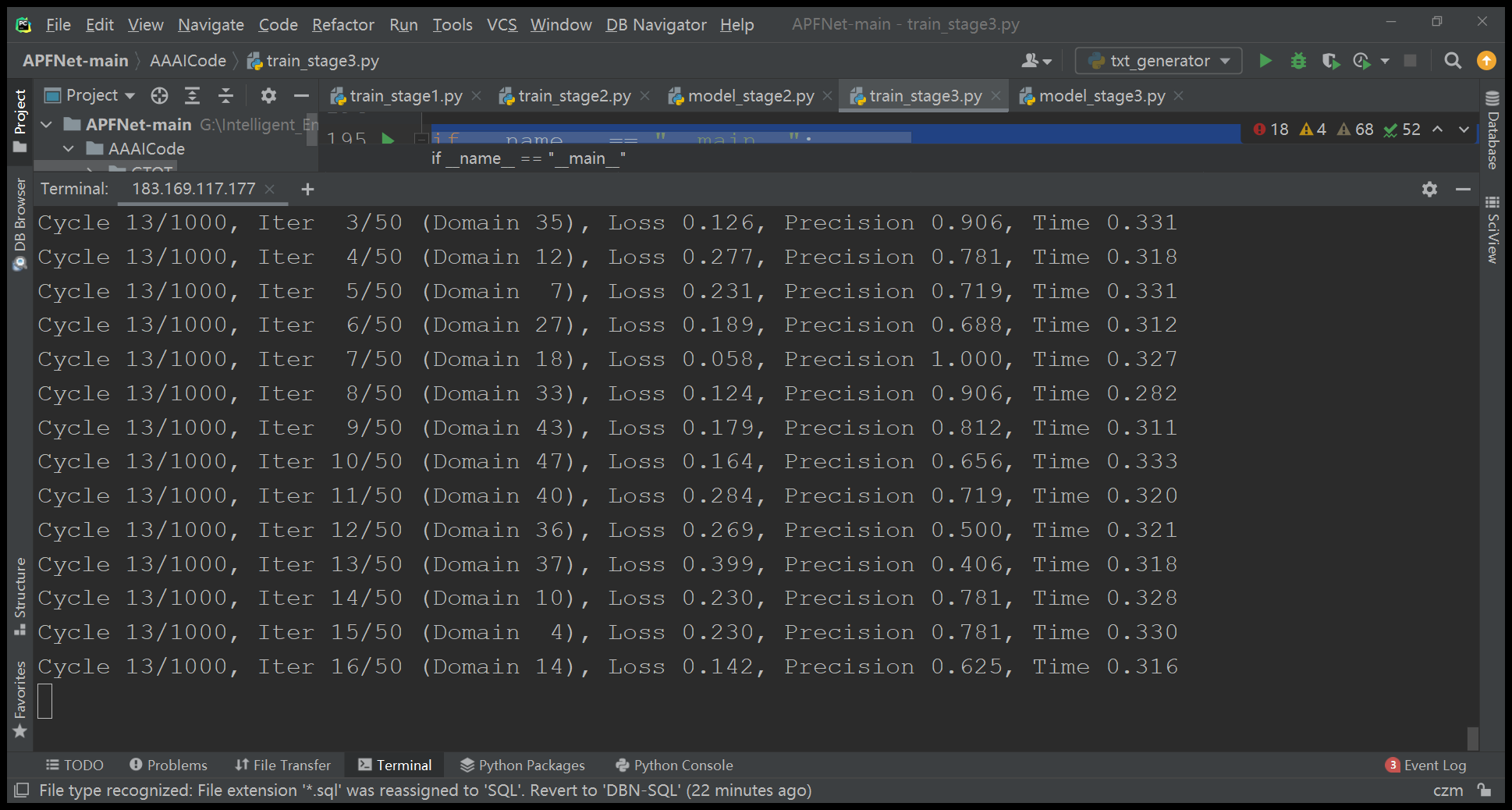


图4-9 阶段三训练过程示意图

然后，我们在利用RGBT234做训练，GTOT做测试的过程中也出现了以下错误：

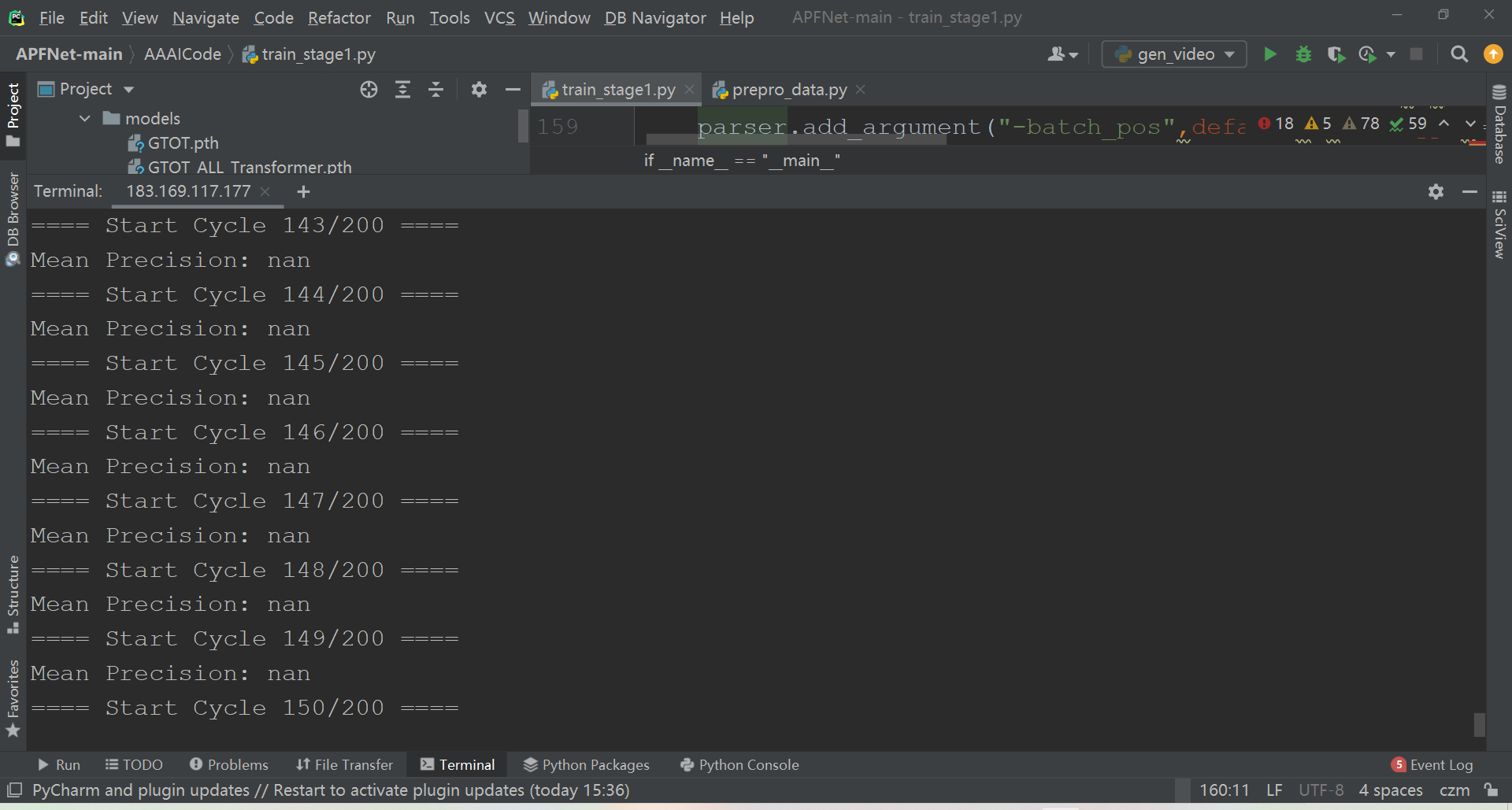


图4-10 RGBT234训练第一阶段出现Nan

分析：在训练网络时出现 Mean Precision: nan 的情况，通常是由于计算中出现了无效的数值（例如，零除、无穷大、缺失值等），导致精度无法被计算。最后通过分析对比GTOT数据集和RGBT234数据集，我们发现RGBT234数据集中没有标注挑战类型的tag文件，导致python程序无法识别出挑战类型，而无法进行第一阶段的训练。我们在网络上找到了最新的RGBT234数据集，含有tag挑战类型标注文件，再利用这个做训练，顺利解决了这个问题；

第5章 总结与体会

RGB-T目标跟踪任务是计算机视觉领域的一项重要研究，旨在利用可见光（RGB）和热成像（Thermal）图像的数据，实现对目标物体的精准跟踪。随着环境光照条件的变化、遮挡和复杂背景的挑战，单一模态的目标跟踪方法面临许多困难，而RGB-T融合技术通过结合两种不同模态的信息，显著提升了目标跟踪的鲁棒性和准确性。

在RGB-T目标跟踪的研究中，APFNet（Attribute-Based Progressive Fusion Network）为我们提供了一个新颖的思路。该网络通过将融合过程解耦为针对五个常见挑战属性（热交叉、光照变化、尺度变化、遮挡、快速运动）的特定分支，有效地解决了多模态信息融合的复杂性。

在参与RGB-T目标跟踪研究的过程中，我深刻体会到多模态信息融合的重要性以及其在实际应用中的挑战。APFNet的设计让我认识到，解决复杂问题的有效途径是将其拆解为多个相对独立的小问题进行处理，从而提高整体系统的性能和适应性。此外，适应性和实时性是我在研究过程中不断思考的两个关键要素，如何在保持模型准确性的同时，确保其在实时应用中的有效性，是我将来研究工作的重点方向。

参考文献

[1] H. Zhang et al., "A Comprehensive Review of RGBT Tracking," in IEEE Transactions on Instrumentation and Measurement, vol. 73, pp. 1-23, 2024, Art no. 5027223, doi: 10.1109/TIM.2024.3436098.

[2] H. Xu, X. Wang and J. Ma, "DRF: Disentangled Representation for Visible and Infrared Image Fusion," in IEEE Transactions on Instrumentation and Measurement, vol. 70, pp. 1-13, 2021, Art no. 5006713, doi: 10.1109/TIM.2021.3056645.

[3] Li, L., Li, C., Tu, Z., Tang, J. (2018). A Fusion Approach to Grayscale-Thermal Tracking with Cross-Modal Sparse Representation. In: Wang, Y., Jiang, Z., Peng, Y. (eds) Image and Graphics Technologies and Applications. IGTA 2018. Communications in Computer and Information Science, vol 875. Springer, Singapore.

[4] Chengwei Luo, Bin Sun, Ke Yang, Taoran Lu, Wei-Chang Yeh, Thermal infrared and visible sequences fusion tracking based on a hybrid tracking framework with adaptive weighting scheme, Infrared Physics & Technology, Volume 99, 2019, Pages 265-276, ISSN 1350-4495.

[5] Zhang, L.; Danelljan, M.; Gonzalez-Garcia, A.; van de Wei jer, J.; and Shahbaz Khan, F. 2019. Multi-modal fusion for end-to-end RGB-T tracking. In Proceedings of the IEEE International Conference on Computer Vision Workshops, 2252–2261.

[6] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems, 5998–6008.

[7] Chen, X.; Yan, B.; Zhu, J.; Wang, D.; Yang, X.; and Lu, H. 2021. Transformer tracking. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 8126–8135.

[8] Y. Xiao, M. Yang, C. Li, L. Liu, and J. Tang, “Attribute-based progressive fusion network for RGBT tracking,” in Proc. 36th AAAI Conf. Artif. Intell. (AAAI), 2022, vol. 36, no. 3, pp. 2831–2838.