

# MARS: A Simple YOLOv8 Implementation

西北工业大学-计算机学院

机器学习与模式识别课程-期末考查作业

U10M11170.03

2025 年春夏学期

授课老师：朱艺宁

## 简介

目标检测任务是计算机视觉领域中的一个重要研究方向，它旨在从图像或视频中识别和定位出特定的物体，并确认它们在图像中的位置。通常目标检测任务需要完成以下两个子任务：

- 分类**：确定图像中每个物体的类别。
- 定位**：确定每个物体在图像中的位置，通常用边界框（bounding box）表示。

YOLOv8 是由 Ultralytics 于 2023 年 1 月 10 日开源的 YOLO 系列的改进版本，它是一个先进的实时物体检测框架。YOLOv8 的网络结构主要由 Backbone、Neck 和 Head 三个部分组成。设计上采用了特征分支堆叠和多尺度特征融合的思想，使模型可以检测不同尺度的物体。同时采取无锚检测（anchor-free）的方式，极大地提升了检测速度。在推出之后取得了广泛的认可，成为当时目标检测任务的 SOTA（state-of-the-art）模型。

## 分组和分数标准

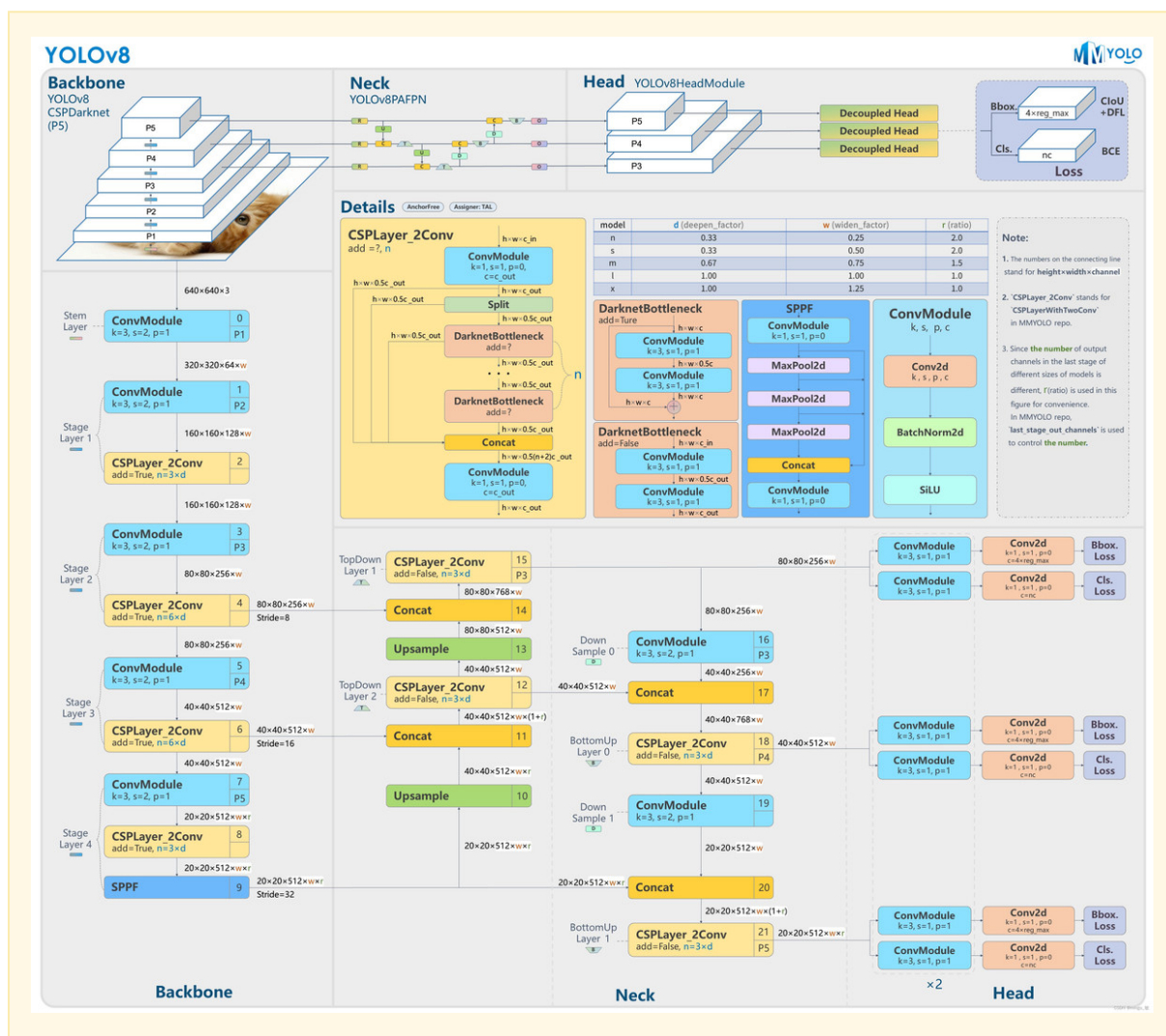
本项目提供了一个 YOLOv8 的简易实现版本。项目参与者需要基于该框架完成 YOLOv8 目标检测的基础功能，并在此基础上实现基于小样本增量学习。具体任务列表以及打分标准如下：

| 任务                                   | 复杂度 | 基础分值 | 流程 tag                | 备注       |
|--------------------------------------|-----|------|-----------------------|----------|
| 阅读项目代码，理解目标检测的流程，并完成 YOLOv8 目标检测基础功能 | 5   | 30   | full                  | Required |
| 完成基于知识蒸馏的增量学习                        | 2   | 20   | teacher, distillation | Required |
| 完成项目报告                               | 3   | 15   |                       | Required |

|                                      |    |      |  |          |
|--------------------------------------|----|------|--|----------|
| 完成问答环节 9 道题（附在项目报告最后面）               | 3  | 20   |  | Required |
| 采用官方预训练模型权重（详见下文）                    | 2  | 20   |  | Bonus    |
| 替换 Backbone 为 Swin-Transformer 并评估效果 | 3  | 25   |  | Bonus    |
| 模型 ema 更新                            | 2  | 10   |  | Bonus    |
| 数据增强                                 | NA | 具体评估 |  | Bonus    |
| 训练流程优化                               | NA | 具体评估 |  | Bonus    |
| 网络结构改进                               | NA | 具体评估 |  | Bonus    |
| 其他改进                                 | NA | 具体评估 |  | Bonus    |

- **项目分组**：建议每组 3 人，最多不超过 4 人。
- **每项分数**：按照完成度与算法效果(mAP)评估，不超过该项最大分数。
- **最终组分数**：按照每组获得的总分进行排序，最终组分数和顺序强相关。
- **最终个人分数**：最终个人分数由最终组分数和个人组内参与度共同决定。
- **Required** 需全部完成，是成绩合格的必要条件。
- **Bonus** 是可选项，按照能力选择完成。需要在报告中说明做了什么，有哪些提升。

## YOLOv8 网络结构



## 必要背景知识

课堂上我们会介绍以下背景知识。

- 神经网络与目标检测任务简介。
- 特征提取模块中的多分枝特征堆叠思想简介。
- 多层特征融合思想简介。
- Anchor-based 和 anchor-free 方法的区别。
- 目标检测训练过程中的正样本匹配。
- 目标检测推理过程中的非极大值抑制。
- AP 和 mAP 评价指标。
- 知识蒸馏简介。
- Python 环境部署与项目运行；pdb 的使用；pytorch 框架简介。
- 如何从模型参数量估计模型推理和训练分别需要的显存大小。

视频资料：

<https://space.bilibili.com/94779326/lists/1789059?type=season>

## MAR20 数据集介绍

MAR20 是目前开源的规模最大的遥感图像军用飞机目标识别数据集，包含 3842 张图像、20 种军用飞机型号以及 22341 个实例。其中 20 个类别的名称分别为 A1, A2, A3, ... A20。

<https://www.ygxb.ac.cn/zh/article/doi/10.11834/jrs.20222139/>

数据集结构如下所示：

```
[A]/auto/cvdata/mar20$ tree . -d
.
├── annotations
├── images
├── splits
│   └── v5
4 directories
```

其中 images 为所有的 jpg 图片。annotations 中为图片的标注信息。splits/v5 为这次项目使用到的数据集划分，具体信息如下：

- ✧ train.txt: 训练用数据集。
- ✧ validation.txt：训练过程中 validation 使用数据集。
- ✧ test.txt：测试数据集。
- ✧ small.txt：知识蒸馏中小样本数据集。

## 小样本增量学习

在应用与现实世界的目标检测任务中，很多情况下模型会遇到之前没有见过的新目标。这个时候我们需要“调整”模型，使之能够识别出更多的之前没见过的新类别，同时保留对旧类别目标的检测能力。为了实现这个目标我们面临如下挑战：

1. 新类别目标样本量少。

2. 重新使用全量数据（包含新旧类别所有样本）训练一个模型的成本过高。
3. 如果只用新类别样本数据去 finetune 已有的模型会造成对旧类别的灾难性遗忘。

为了解决上述问题，通常采取模型冻结、旧样本回放、知识蒸馏等方法。

<https://blog.csdn.net/u012347027/article/details/111415197>

<https://arxiv.org/abs/2011.13256>

<https://blog.csdn.net/keeppractice/article/details/145419077>

<https://blog.csdn.net/keeppractice/article/details/145419077>

- 本项目中我们采取 feature-based 知识蒸馏和 response-based 知识蒸馏相结合的方法。其中 feature-based 的蒸馏推荐采用 CWD 等方法。response-based 知识蒸馏可以仅针对分类部分，采用 KLDivergence 损失函数，也可以使用其他方法。
- 注意：我们不限使用其他蒸馏方法，项目报告中需要注明使用的蒸馏方法和效果。feature-based 和 response-based 方法至少各自实现一种。如果实现了效果更好的蒸馏方法可以得到 bonus 分数。

## 项目流程

我们在项目中提前配置好了不同的流程，只需要将其中的路径替换为本地的路径即可。大家需要自行理解项目配置的方法，详见 README。

```
MarsEngine(  
    mode=mode,  
    cfgname="c1.nano.full",  
    # cfgname="c1.nano.teacher",  
    # cfgname="c1.nano.distillation",  
    root="/auto/mars", # 注意项目运行root不要放在代码路径下  
    nobuf=nobuf,  
).run()
```

```

mcfg.phase = "nano" # DO NOT MODIFY
mcfg.trainSplitName = "train" # DO NOT MODIFY
mcfg.validationSplitName = "validation" # DO NOT MODIFY
mcfg.testSplitName = "test" # DO NOT MODIFY

# data setup
mcfg.imageDir = "/auto/cvdata/mar20/images"
mcfg.annotationDir = "/auto/cvdata/mar20/annotations"
mcfg.classList = ["A{}".format(x) for x in range(1, 21)] # DO NOT MODIFY
mcfg.subsetMap = { # DO NOT MODIFY
    ... "train": "/auto/cvdata/mar20/splits/v5/train.txt",
    ... "validation": "/auto/cvdata/mar20/splits/v5/validation.txt",
    ... "test": "/auto/cvdata/mar20/splits/v5/test.txt",
    ... "small": "/auto/cvdata/mar20/splits/v5/small.txt",
}

if "full" in tags:
    ... mcfg.modelName = "base"
    ... mcfg.maxEpoch = 200
    ... mcfg.backboneFreezeEpochs = [x for x in range(0, 100)]

if "teacher" in tags:
    ... mcfg.modelName = "base"
    ... mcfg.maxEpoch = 200
    ... mcfg.backboneFreezeEpochs = [x for x in range(0, 100)]
    ... mcfg.trainSelectedClasses = ["A{}".format(x) for x in range(1, 11)] # DO NOT MODIFY

if "distillation" in tags:
    ... mcfg.modelName = "distillation"
    ... mcfg.checkpointModelFile = "/auto/mars/ame/c1.nano.teacher/__cache__/best_weights.pth"
    ... mcfg.teacherModelFile = "/auto/mars/ame/c1.nano.teacher/__cache__/best_weights.pth"
    ... mcfg.distilLossWeights = (1.0, 0.05, 0.001)
    ... mcfg.maxEpoch = 100
    ... mcfg.backboneFreezeEpochs = [x for x in range(0, 25)]
    ... mcfg.epochValidation = False # DO NOT MODIFY
    ... mcfg.trainSplitName = "small" # DO NOT MODIFY
    ... mcfg.teacherClassIndexes = [x for x in range(0, 10)] # DO NOT MODIFY

```

如上图所示，完整的项目分为三个子流程。分别为：

1. full：实现目标检测的基础功能，用全类数据（所有 20 类）训练。
2. teacher：知识蒸馏中的教师网络训练，只用前 10 类进行训练（通过设置“trainSelectedClasses”来实现 dataloader 加载特定的类别）。
3. distillation：知识蒸馏中的学生网络训练，只用 small 数据集训练。用来模拟小样本增量学习。

**注意：**

1. 标记为#DO NOT MODIFY 的配置请勿修改。
2. 为了测试结果的统一性，我们统一使用 nano 网络来训练和评估，代码中已经配置好。



### 3. 知识蒸馏相关的流程，我们统一规定 A1-A10 为旧类别，A11-A20 为新类别。

## 结果评估

- 我们使用 AP 和 mAP 来评估最终结果。
- 项目代码中包含了讲检测结果画出来并保存成图片的代码(painter.py)，可以在配置中添加相应配置并直接调用，以便大家直观欣赏模型运行的结果。但是就评价指标而言，我们只需要关注 AP 和 mAP 即可。

## 项目报告要求

1. 写明每个流程程序运行的结果（每一类的 AP 和 mAP）。可直接截图并标注流程 tag 名称，如下所示：

```
[2025-04-26T11:00:46.078748|engine.py:35] AP:
classIndex  className  AP
0 0 A1 0.847786
1 1 A2 0.984095
2 2 A3 0.870584
3 3 A4 0.737679
4 4 A5 0.743151
5 5 A6 1.000000
6 6 A7 0.957027
7 7 A8 0.949225
8 8 A9 0.911985
9 9 A10 0.972949
10 10 A11 0.883439
11 11 A12 0.843890
12 12 A13 0.899838
13 13 A14 0.936215
14 14 A15 0.672918
15 15 A16 0.958108
16 16 A17 0.976385
17 17 A18 0.961538
18 18 A19 0.761756
19 19 A20 0.794822

[2025-04-26T11:00:46.079381|engine.py:36] mAP=0.883

[2025-04-26T10:07:36.681860|engine.py:35] AP:
classIndex  className  AP
0 0 A1 0.799485
1 1 A2 0.938035
2 2 A3 0.881444
3 3 A4 0.888561
4 4 A5 0.724434
5 5 A6 0.862066
6 6 A7 0.954098
7 7 A8 0.966876
8 8 A9 0.964615
9 9 A10 0.950103
10 10 A11 0.000000
11 11 A12 0.000000
12 12 A13 0.000000
13 13 A14 0.000000
14 14 A15 0.000000
15 15 A16 0.000000
16 16 A17 0.000000
17 17 A18 0.000000
18 18 A19 0.000000
19 19 A20 0.000000

[2025-04-26T10:07:36.682221|engine.py:36] mAP=0.446

--Return--
> /sandbox/ame/workspace/cvrepos/mars/engine/engine.py(37)view()->None
-> import pdb; pdb.set_trace()
```

full teacher

```
[2025-04-26T12:18:19.488629|engine.py:35] AP:
classIndex  className  AP
0 0 A1 0.842938
1 1 A2 0.958126
2 2 A3 0.858622
3 3 A4 0.888541
4 4 A5 0.645863
5 5 A6 0.710654
6 6 A7 0.836789
7 7 A8 0.953730
8 8 A9 0.904560
9 9 A10 0.925873
10 10 A11 0.000000
11 11 A12 0.132061
12 12 A13 0.339822
13 13 A14 0.603690
14 14 A15 0.000000
15 15 A16 0.510717
16 16 A17 0.564428
17 17 A18 0.000000
18 18 A19 0.568050
19 19 A20 0.265523

[2025-04-26T12:18:19.489041|engine.py:36] mAP=0.575

--Return--
> /sandbox/ame/workspace/cvrepos/mars/engine/engine.py(37)view()->None
-> import pdb; pdb.set_trace()
```

distillation

```
[2025-04-26T12:18:56.253786|engine.py:35] AP:
classIndex  className  AP
0 0 A1 0.798008
1 1 A2 0.978595
2 2 A3 0.766307
3 3 A4 0.785735
4 4 A5 0.387800
5 5 A6 0.938427
6 6 A7 0.896143
7 7 A8 0.892033
8 8 A9 0.951663
9 9 A10 0.944480
10 10 A11 0.507503
11 11 A12 0.457562
12 12 A13 0.592881
13 13 A14 0.702998
14 14 A15 0.110056
15 15 A16 0.728644
16 16 A17 0.831956
17 17 A18 0.476117
18 18 A19 0.644219
19 19 A20 0.476864

[2025-04-26T12:18:56.254200|engine.py:36] mAP=0.693

--Return--
> /sandbox/ame/workspace/cvrepos/mars/engine/engine.py(37)view()->None
-> import pdb; pdb.set_trace()
```


DistillationPlus

2. 不要粘贴代码，会让报告冗长。
3. 写清楚参与成员的分工、各自贡献（该部分缺失或不详的小组会被要求补充）。
4. 所有的 bonus 项目，以及额外实现的功能，都请注明该功能添加之后的效果提升（只需要截图 AP 和 mAP 即可）。

5. 上图 distillationPlus 为在原本 distillation 的基础上增加了若干改进得到的结果。

## 代码规范

代码风格应和项目原始代码类似，或者参考 Google 编程风格指南

[https://zh-google-styleguide.readthedocs.io/en...](https://zh-google-styleguide.readthedocs.io/en/latest/google-python-styleguide/python3-guidelines/)  
<https://zh-google-styleguide.readthedocs.io/en/latest/google-py...>

本项目并不严格要求大家遵循某个指南，但是对于以下几个点会重点关注：

| 考察点                            | 正确  | 错误  |
|--------------------------------|---|---|
| 运算符两边加空格                       | <code>a = b + c</code>  | <code>a=b+c</code>  |
| 逗号后加空格                         | <code>x = f(a, b)</code><br><code>x = [1, 2, 3]</code>                          | <code>x=f(a,b)</code><br><code>x=[1,2,3]</code>   |
| 类名/方法名含义清晰，类名首字母大写，函数/方法名首字母小写 | <code>class MeowEngine(object):</code><br><code>def fit(self, xdf, ydf):</code> | <code>class meow_engine(object):</code><br><code>def Fit(self, xdf, ydf):</code><br><code>class X(object): (含义不清)</code><br><code>def abc(x): (含义不清)</code> |

**关于注释：**请勿过分注释；请勿无效注释。不同于其他开源项目开发中详细的注释风格，此项目中由于授课老师和助教是大家代码最终的阅读者，理想状态下阅读者应该可以直接从文件名，类名，方法名了解到该类是什么，该方法具体要做什么。我们鼓励零注释或仅在必要地方注释。鼓励代码风格简单紧凑和模块化。

## 采用官方预训练模型权重

读取 YOLOv8 官方的预训练模型文件(.pth 或者.pt)，修改其中的参数名，使之和本项目模型代码匹配，并使用预训练的参数初始化模型。

- 充分理解.pth 文件中保存的是什么，Pytorch 如何将.pth 文件中的权重加载到模型中。
- 我们自己构建的模型如果网络结构和官方的模型一致，这个时候只需要修改.pth 文件中的参数名，就可以将别人预训练好的参数加载到自己的模型中。
- 如何下载官方的模型（选择 yolov8n.pt）：  
<https://docs.ultralytics.com/models/yolov8/#key-features-of-yolov8>

参考资料 [https://blog.csdn.net/weixin\\_43863869/article/details/129664097](https://blog.csdn.net/weixin_43863869/article/details/129664097)



## 数据/项目初始代码下载

数据集：

<https://kaggle.com/datasets/8c8e80acb36310fda8be762de0a77559832179b3ef14c...>

代码：<https://github.com/NWPU-CPS/Mars>

## 代码提交

提交方式：请将项目文件打包成 zip 文件发送至课程邮箱 [ml\\_course2025@163.com](mailto:ml_course2025@163.com)

邮件命名格式：ML2025final\_学号\_姓名\_第 X 组

附件文件命名：ML2025final\_学号\_姓名\_第 X 组.zip

截止时间：5 月 30 日 23:59 分

- 请勿包含数据或者模型缓存文件（转换的官方预训练的 backbone 除外），仅需打包必要的代码和项目报告即可。
- 项目报告应包括：**内容介绍、结果分析与总结、成员分工说明、问答环节。**
- 参考项目初始代码运行方式，我们会更改你所提交的代码中的相关路径，然后直接使用

```
1 $ python mars.py
```

来运行代码。因此对项目的入口处的代码请保持和初始代码同样的风格。我们所使用的用来训练和评估预测结果的数据和项目提供的数据集完全一致。

## 环境要求

- Python  $\geq 3.10$ 。
- 推荐使用 Anaconda 提供的 Python 环境。
- GPU 显存  $\geq 4GB$

## 注意事项

1. 项目启动过程中的难点在于理解现有的框架，我们推荐大家使用 pdb 这个工具，在运行项目的过程中熟悉从 dataloader 的数据加载逻辑，到模型内的 dataflow 逻辑，最后经过 loss

模块计算 loss。这个流程中的每个过程都可以用 pdb 暂停到某一步，然后查看变量的值和结构(x.shape)。掌握项目代码的关键就是掌握项目内的数据流动和结构变化。

2. 所有项目代码中未实现的部分均标注了：NotImplementedError。这些部分需要大家完成。
3. 项目完成过程中，需要大家查阅大量资料，而不是拘泥于教科书和 ppt。因此信息搜集和整合很重要。
4. 组间禁止抄袭代码。我们会对提交的代码进行相似度检测，并对相似度高的 team 成员面试。注意：代码重命名，重构，添加冗余代码/注释等操作不能降低相似度。
5. 请在报告中明确每一位成员的分工，明确所提创新点的贡献者。
6. 无法自行解决计算设备的同学可以联系助教获取帮助。
7. 禁止直接使用官方 Ultralytics 或者其他 YOLOv8 的实现版本，但是欢迎大家参考这些项目中的代码并使用。

## 问答环节

请在项目报告中对如下问题做出你的回答（要素**齐全**、**精炼**）：

1. 如何理解这句话的含义：“All models are wrong, but some are useful.”
2. 基于深度学习的目标检测任务中，深层网络和浅层网络分别适合检测大目标还是小目标？
3. 阐述你对 YOLOv8 网络结构中 neck 部分的理解。
4. 阐述你对 CSP layer（也叫 C2F）模块的理解。
5. 简述训练过程中正样本匹配的流程。（可以用语言描述也可以用流程图）
6. 简述推理过程中非极大值抑制的流程。（可以用语言描述也可以用流程图）
7. 如何理解 YOLOv8 损失函数中的类别损失是什么？如何理解？类别预测输出激活函数为什么用 sigmoid 而不是 softmax？
8. 如何理解 YOLOv8 损失函数中的 bounding box 的 DFL 损失？
9. 阐述你对不同知识蒸馏方式的理解。