ComSen2026寒假任务

概述

本次寒假任务分为两部分：以个人为单位完成的个人任务和以大创项目组为单位的团队任务。其中团队任务中的文献调研部分和基线选择部分原则上需要留校完成（预计3-4d），若有特殊情况可以上报。个人任务包括算法和硬件方向(智能体方向后续单独发布），完成对应方向的题目即可。**团队任务提交截止时间为1.31，个人任务提交截止时间为2.28。**

大创项目组任务

以大创项目组为单位，完成如下任务。

文献调研

阅读至少5篇文献，完成一篇文献综述（中英文均可），主要阐述该领域前人所做工作以及各种方法的优缺点。

基线选择及复现

根据文献调研的结果，选取一种方法作为基线并在公开数据集上对其进行复现。

个人任务

算法方向：零样本目标检测

任务目标

在**不使用目标类别标注框训练数据**（或仅使用源数据集的已知类标注框）的前提下，实现一个零样本目标检测系统，使其能够对未见过的新类别进行检测，并在标准评测集上完成定量评估与分析。由于零样本目标检测背后的原理较为复杂，可以将本任务视为纯粹的工程性任务，不需要过于详细了解算法原理。

基线复现（必做）

选择并复现一个公开的零样本目标检测方法作为基线（任选其一）：

- GroundingDINO（文本引导检测，工程可复现性较好）
- GLIP / GLIPv2（语言-视觉预训练 + 检测）
- OWL-ViT（开放词表检测，ViT+文本/类名嵌入）

- Detic（开放词表检测，结合分类词表扩展）
- YOLO-E（开放词汇目标检测，速度较快）

要求：

- 给出可运行代码/脚本（推理与评测能一键运行）。
- 在指定数据集上输出检测结果与 mAP（或 AP50 等）指标。

零样本设置与数据集（必做）

选择一个评测数据集并构造“已知类/未知类”划分（可用经典划分或自行制定，但要说明）：

推荐数据集（选 1 个即可）：

- MS COCO：常用做法是 65 seen / 15 unseen 或其他划分
- Pascal VOC：常用 VOC split（如 15/5 或 10/10）
- LVIS（更贴近开放词表，但工作量略大）

你需要：

- 明确定义 seen/unseen 类别列表
- 只在允许的监督范围内训练/调参（例如：只用 seen 类框监督，unseen 不可用框训练）

提示词（prompt）工程与对比实验（必做）

围绕“文本/类别描述如何影响检测效果”做系统实验。至少包含以下三种 prompt 形式，并比较对 unseen 类的影响：

1. 纯类名：`"cat"`
2. 模板句：`"a photo of a {class}"`
3. 细粒度描述：为每个类写 1~2 句属性描述（颜色/形状/用途/部件等）

要求输出：

- 不同 prompt 的定量对比表（至少 AP50 或 mAP）
- 2~3 个类别的可视化对比案例（同一图片不同 prompt 的检测差异）

小幅改进（必做：1 个点即可）

在基线基础上做一个一周内可实现的改进点，并通过实验验证有效性或分析其失败原因。可选方向（任选 1 个，不局限于下面的方向）：

方向 A：多提示集成（Prompt Ensembling）

- 为每个类别构造多种模板/描述，融合多次推理结果（如取最大置信度、加权融合、WBF/NMS 后融合）

方向 B：类别同义词扩展

- 为每个类别加入同义词/上位词/近义短语（例如 “sofa/couch”）
- 分析“词汇扩展”对召回率与误检的影响

方向 C：阈值/后处理自适应

- 为不同类别设置不同阈值（基于 seen 类验证集统计或基于语言相似度启发式）
- 或改进 NMS 策略（class-agnostic vs class-aware）对开放词表的影响

方向 D：语言引导的区域重排序

- 使用 CLIP 对候选框 crop 与文本做二次相似度打分重排（rerank），观察对误检的影响

要求：

- 明确改进动机与实现细节
- 给出与基线的对比实验（至少 1 组指标 + 若干可视化）

交付物（必交）

1. 代码仓库
    - 环境说明（conda/pip requirements）
    - 推理脚本、评测脚本、一键运行说明
2. 实验报告（PDF/Markdown）
    - 方法简介、零样本设定、实验设计
    - 定量结果表格、可视化案例
    - 失败案例分析（至少 2 张图）
3. 结果文件
    - 检测结果（JSON/COCO 格式等）
    - 关键日志（训练/推理参数、阈值等）

可选加分项（不要求）

- 在多个 unseen 类划分上验证泛化（不同 split）
- 关注长尾类别（如 LVIS rare）并分析语言偏置
- 速度/显存优化或部署（batch 推理、半精度等）

硬件方向：简易数字时钟

任务目标

设计并实现一个带有24小时制时钟显示和秒表计时功能的数字系统。

功能要求

- 基础功能 (时钟模式):
  - 显示：使用数码管显示 `时(HH) : 分(MM) : 秒(SS)`。
  - 进制：秒和分是60进制，时是24进制。
- 进阶功能 (秒表模式，选做):
  - 模式切换：通过一个开关或按键在“时钟模式”和“秒表模式”之间切换。
  - 精度：精确到0.01秒 (10ms)。
  - 控制：
    - Start/Stop: 启动或暂停计时。
    - Reset: 复位清零（仅在暂停时有效）。

硬件架构参考（如有余力可使用FPGA实现）

- 时钟信号源（使用555定时器或对称与非门构成的多谐振荡器）
- 控制电路（用于切换模式和控制秒表模式计时的开始与停止）
- 外部RTC时钟
- 计数器芯片
- 译码器芯片，用于将计数器芯片输出的四位数据转换成驱动晶体管的信号

交付物（必交）

- 原理图和PCB制板文件
- 演示视频
- HDL工程（若使用FPGA）

智能体方向：微调模型

学习并实践DataWhale开源在Gihub的Happy-LLM相关教程，重点关注5-6章。

https://github.com/datawhalechina/happy-llm
github.com

最终任务要求：实现一个LLM的微调（微调方式根据任务目标选择最合适的）并将微调后的LLM以API接口的形式部署在云服务器端，然后本地构建一个智能体来调用这个LLM。

建议：

- 可以尝试更高级的Agent框架：LangGraph，重点关注其中状态机和智能体工作图（Agent Graph）的实现

- 可以尝试自己调研现在市场上的需求，设计一个能真正解决需求的Agent产品（市场调研、需求挖掘、产品设计）
- 思考：微调后LLM的效果是更好了还是更差了？为什么？
- 尝试多种不同的微调策略，思考：微调是否有必要？怎么看待有人说提示词工程（Prompt Enginering）能实现任何微调能实现的功能（即微调是不必要的）？
- 可以尝试复现一些前沿论文的LLM训练策略，如：
  - Deepseek R1论文：https://arxiv.org/pdf/2501.12948
  - OpenAI 模型可解释性：《Weight-sparse transformers have interpretable circuits》