合肥工業大學

2025 年《机器视觉》实验

实验内容：校园共享单车检测

姓　　名：折浩宇

学　　号：2023217896

完成时间：2025/12/27

一、实验目的

1. **理解目标检测的核心任务：**

   掌握目标检测在机器视觉中的作用，理解其同时完成“定位（给出目标位置/边界框）+识别（判断类别）”的双重目标。

2. **熟悉目标检测完整流程：**

   通过共享单车场景，系统理解并实践目标检测的基本链路：特征提取 → 目标定位 → 分类判断。

3. **完成校园场景共享单车检测任务：**

   面向校园道路、停车区等常见图像，设计并实现共享单车的检测方案，使模型能够输出共享单车的位置（bounding box/坐标）等结果。

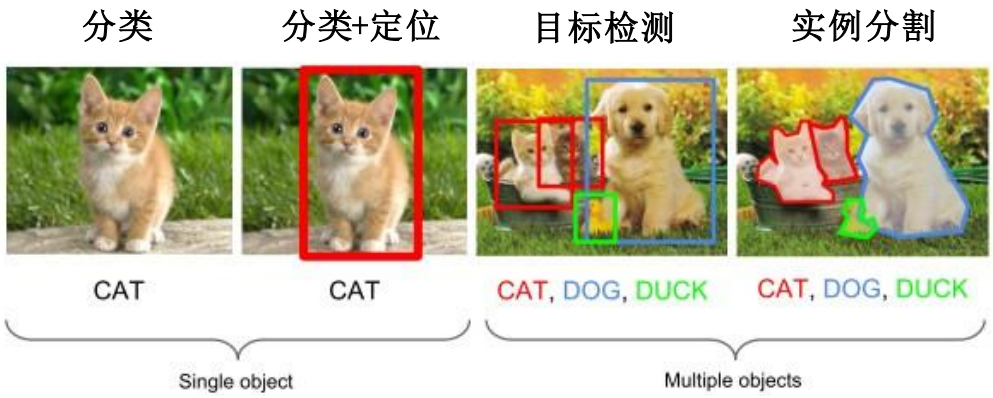
4. **训练与评估检测模型的工程能力：**

   基于给定数据集（如 COCO），完成数据准备、模型训练、结果可视化与性能评估，提升对目标检测算法与深度学习训练流程的实践能力。

二、实验原理

2.1 目标检测任务本质

分类：解决是什么？的问题，即给定一张图片或一段视频判断里面包含什么类别的目标。
定位：解决在哪里？的问题，即定位出这个目标的的位置。
检测：解决是什么？在哪里？的问题，即定位出这个目标的的位置并且知道目标物是什么
分割：分为实例的分割和场景分割，解决“每一个像素属于哪个目标物或场景”的问题。

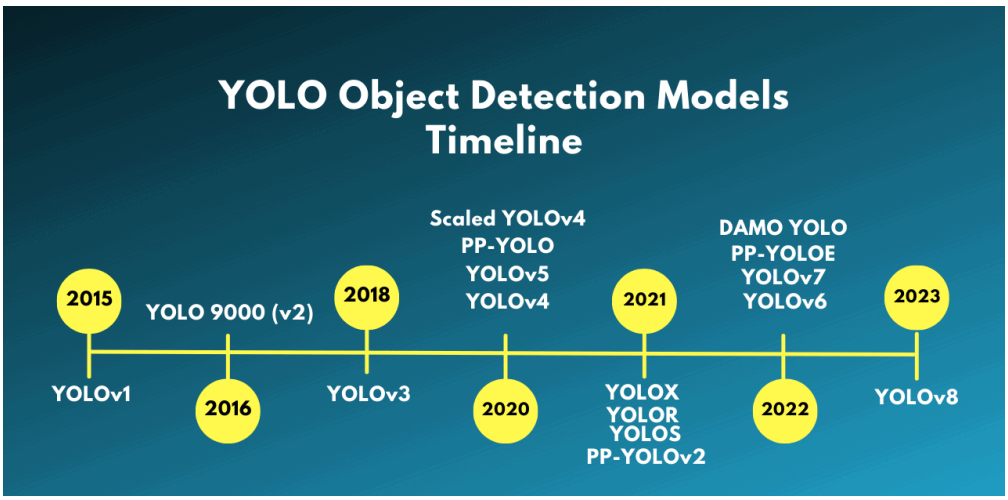


2.2 YOLO 简介

在早期阶段，**目标检测主要依赖于手工设计的特征提取算法和分类器**。这些方法通常基于边缘、纹理、颜色等低级特征，并结合模板匹配或统计模型进行目

标检测。然而，**这些方法受限于特征的表达能力和鲁棒性，对于复杂场景的检测效果较差。**

近年来，深度学习的兴起极大地推动了目标检测技术的发展。最具代表性的方法之一是基于区域的卷积神经网络(R-CNN)，它将目标检测任务分解为候选区域提取和区域分类两个子任务。后续的方法，如 Fast R-CNN、Faster R-CNN 和 YOLO(You Only Look Once)等进一步改进了速度和准确性。这些方法不仅能够自动学习特征表示，还能够在端到端的框架下进行目标检测。



2.3 YOLO 系列算法的步骤：

（1） 划分图像：

YOLO 将输入图像划分为一个固定大小的网格。

（2） 预测边界框和类别：

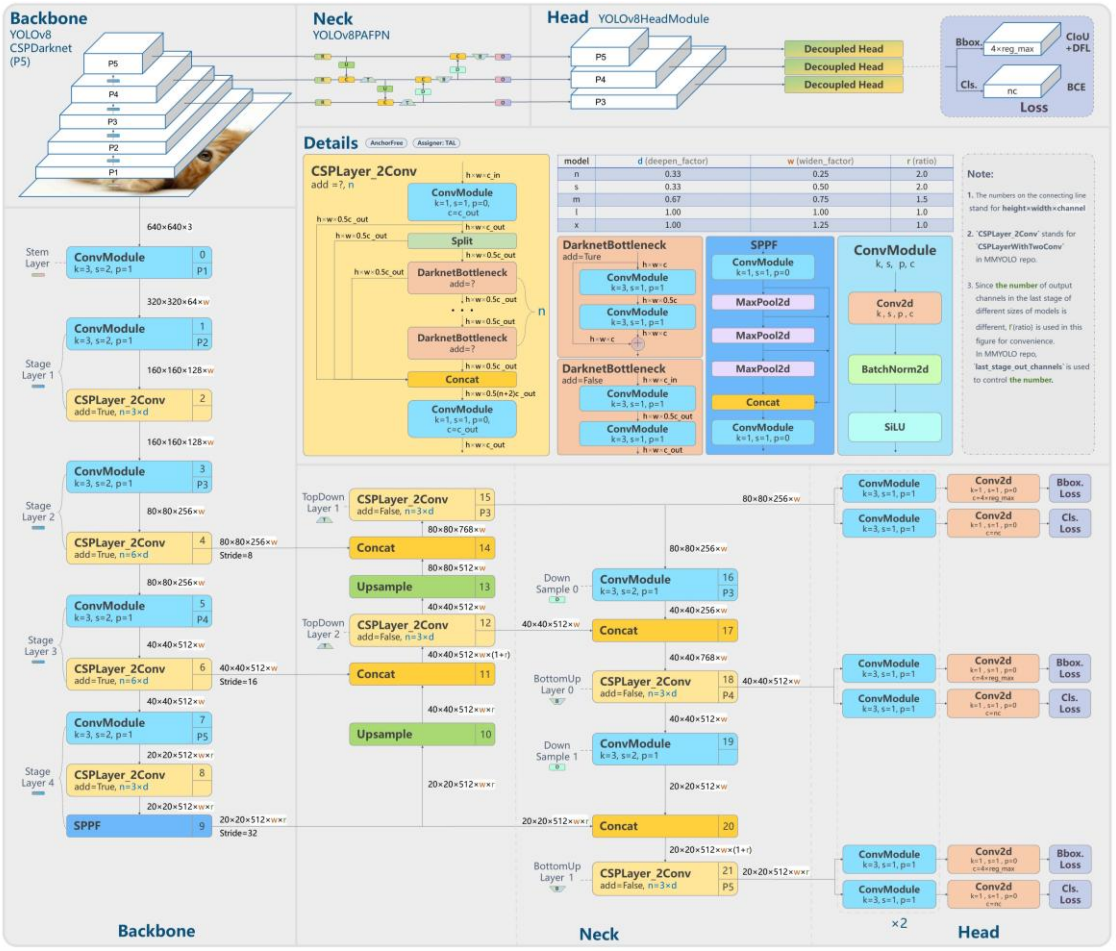
对于每个网格，YOLO 预测出固定数量（通常为 5 个或 3 个）的边界框。每个边界框由 5 个主要属性描述：边界框的位置（中心坐标和宽高）和边界框包含的目标的置信度。此外，每个边界框还预测目标的类别。

（3） 单次前向传递：

YOLO 通过一个卷积神经网络（CNN）进行单次前向传递，同时预测所有边界框的位置和类别。(4) 损失函数：YOLO 使用多任务损失函数来训练网络。该损失函数包括位置损失、置信度损失和类别损失。位置损失衡量预测边界框和真实边界框之间的位置差异。置信度损失衡量边界框是否正确地预测了目标，并惩罚背景框的置信度。类别损失衡量目标类别的预测准确性。

（4） 非最大抑制：

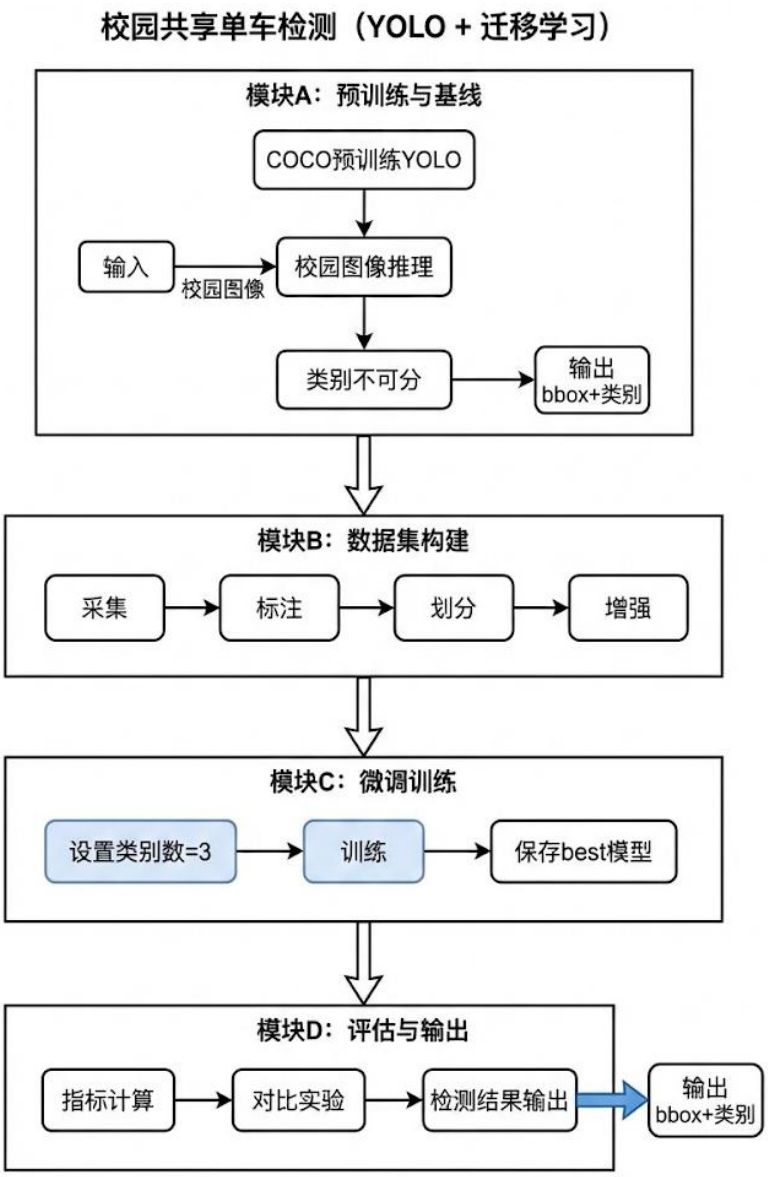
在预测的边界框中，可能存在多个相互重叠的框，代表同一个目标。为了消除冗余的边界框，YOLO 使用非最大抑制算法，根据置信度和重叠程度筛选出最佳的边界框。



三、实验流程与解析

3.1 整体流程

本实验采用迁移学习思路完成校园场景两轮车目标检测。首先**训练随机权重在 COCO 数据集上的 YOLO 模型**，在校园图像上进行基线测试，发现 COCO 类别体系无法区分“电动车”和“哈啰单车”。为实现细粒度识别，随后自行采集并标注“电动车/哈啰单车/汽车”三类数据集，按训练/验证/测试集划分后，对 YOLO 模型进行 **nc=3 的类别适配与微调训练**。最后使用 mAP、Precision、Recall 等指标评估模型，并通过可视化推理结果验证模型能够输出目标位置（bbox）并正确区分两类目标



实验流程图

3.2 基线方案：COCO 预训练 YOLO（随机初始化）与测试

基线目的：

为获得具备通用目标检测能力的模型，并验证 COCO 类别体系对本实验细粒度类别（电动车、哈啰单车）的支持情况，本实验首先在 COCO 数据集上对 YOLO 模型进行训练，得到 COCO 基线模型。

基线训练方法（随机初始化）

- **模型初始化方式：**不加载任何预训练权重，模型参数采用随机初始化。
- **训练数据：**COCO 目标检测数据集。

- **训练过程：**使用 COCO 的标注类别对 YOLO 进行端到端训练，使模型学习通用目标检测能力（目标定位与分类）。

测试与现象记录：

训练完成后，将得到的 COCO 基线模型用于校园场景图片推理测试，观察输出类别与检测结果。实验发现：

- 模型能够完成一定程度的两轮车目标检测（输出边界框），**但输出类别受限于 COCO 的标签体系；**
- COCO 的**类别定义无法覆盖“哈啰单车”这一细分类别，同时对校园电动车也缺乏与实验需求一致的专门类别；**
- 因此仅依赖 COCO 训练得到的模型**难以实现“电动车 vs 哈啰单车”的稳定区分。**

结论；

COCO 基线模型具备通用检测能力，但受限于数据集标签粒度，无法满足本实验对细粒度类别区分的要求。为实现“电动车”和“哈啰单车”的准确区分，需要构建自定义数据集并在此基础上进行微调训练。

3.3 自建数据集构建（采集、定义、标注）

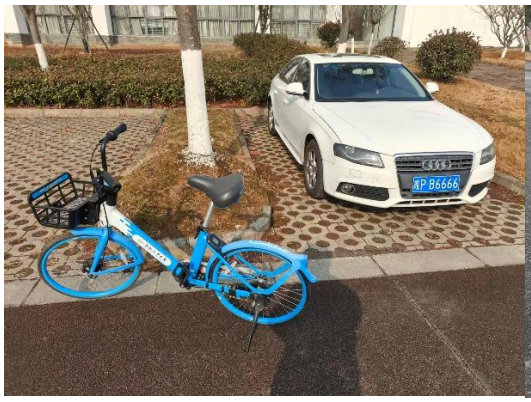
3.3.1 类别定义与标注规范：

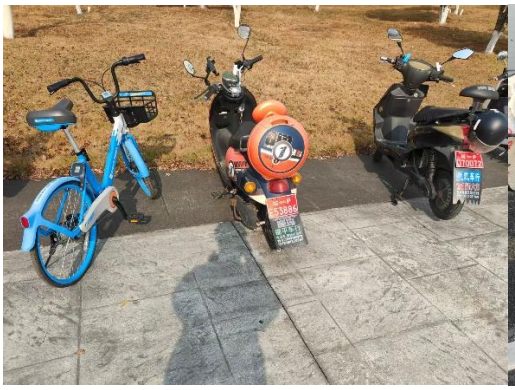
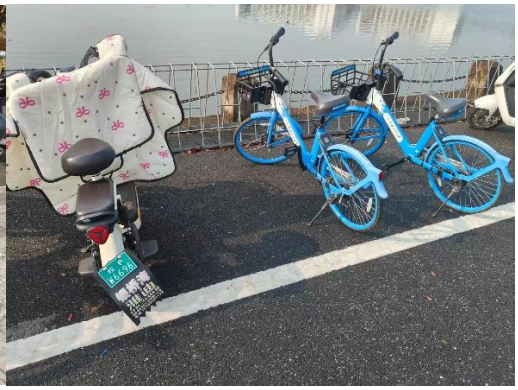
- **哈啰单车：**共享单车外观特征明显、可识别共享单车结构。
- **电动车：**具备电动两轮车典型结构（更厚的车身/脚踏板/电池仓/车座结构明显等）。
- **汽车**

3.3.2 数据采集：（采用宣城校区内图片）

- **场景覆盖：**道路、停车区、教学楼周边等典型校园区域。
- **多样性覆盖：**不同光照（晴天/阴天/夜间）、不同距离与角度、遮挡/密集停放等情况。





3.3.3 数据清理：

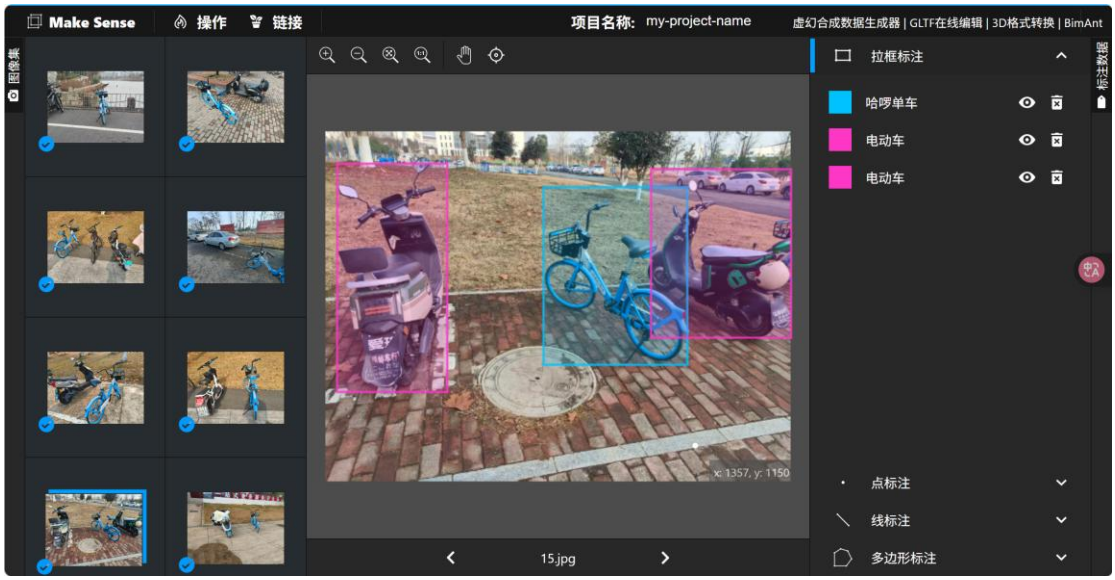
利用 python 脚本进行图像编号的调整

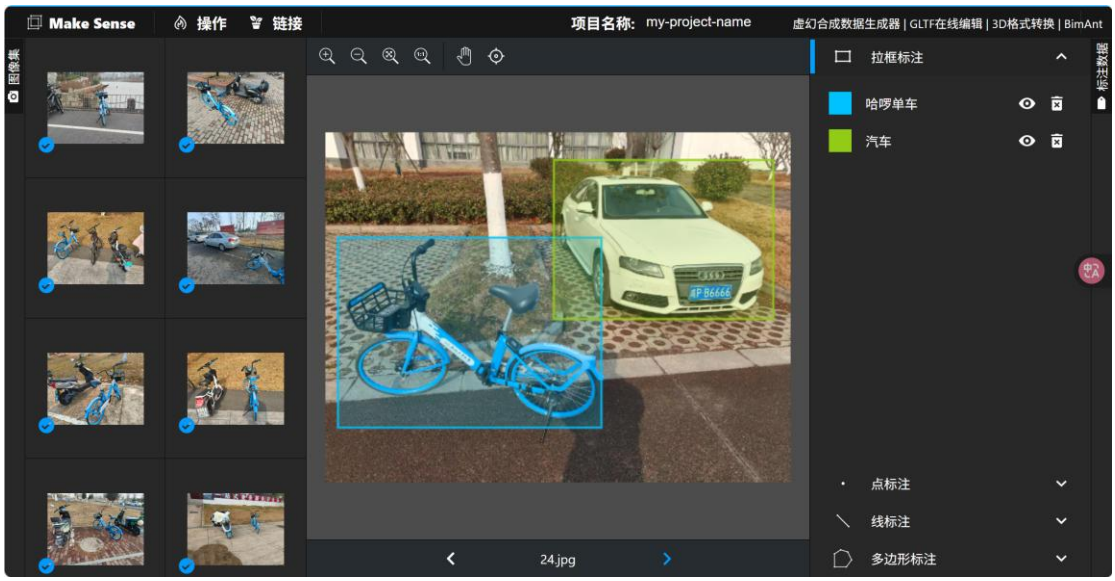
```python
def rename_images_in_subfolders(base_dir: Path):
    for sub in sorted([p for p in base_dir.iterdir() if p.is_dir()]):
        imgs = sorted(
            [p for p in sub.iterdir() if p.is_file() and p.suffix.lower() in IMAGE_EXTS],
            key=lambda p: p.name.lower()
        )

        if not imgs:
            continue

        # 先改成临时名，避免 1.jpg 已存在导致冲突
        tmp_paths = []
        for i, p in enumerate(imgs, start=1):
            tmp = sub / f"__tmp__{i:06d}{p.suffix.lower()}"
            p.rename(tmp)
            tmp_paths.append(tmp)

        # 再改成最终名：1.xxx / 2.xxx ...
        for i, tmp in enumerate(tmp_paths, start=1):
            final = sub / f"{i}{tmp.suffix.lower()}"
            tmp.rename(final)

        print(f"[OK] {sub.name}: renamed {len(imgs)} images")

if __name__ == "__main__":
    rename_images_in_subfolders(Path.cwd())
```

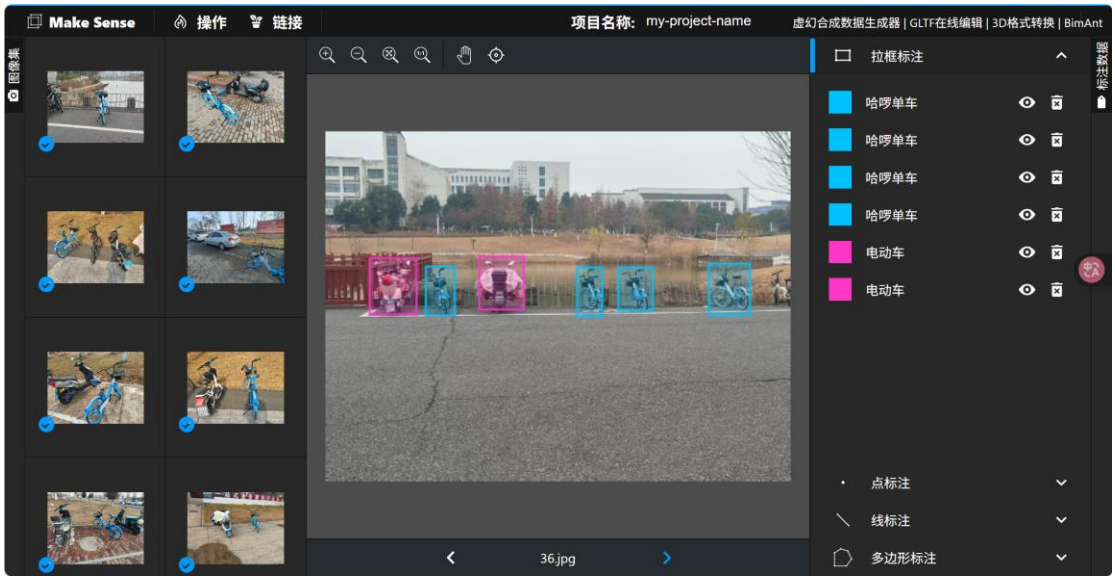
调整图像编号后

3.3.4 数据标注：

- 使用标注工具对目标绘制 bbox。
- 保存为 YOLO 格式标签：class x_center y_center width height（坐标归一化）。
- 确保类别 id 只出现 0、 1 和 2，与两类定义一致。


Make Sense
操作
链接
项目名称: my-project-name
虚幻合成数据生成器 | GLTF在线编辑 | 3D格式转换 | BimAnt
图像集
标注数据
拉框标注
哈啰单车
电动车
电动车
点标注
线标注
多边形标注
15.jpg


Make Sense
操作
链接
项目名称: my-project-name
虚幻合成数据生成器 | GLTF在线编辑 | 3D格式转换 | BimAnt
图像集
标注数据
拉框标注
哈啰单车
汽车
点标注
线标注
多边形标注
24.jpg


Make Sense
操作
链接
项目名称: my-project-name
虚幻合成数据生成器 | GLTF在线编辑 | 3D格式转换 | BimAnt
图像集
标注数据
拉框标注
哈啰单车
哈啰单车
哈啰单车
哈啰单车
电动车
电动车
点标注
线标注
多边形标注
36.jpg

图像标注过程

3.4 数据集划分与数据增强：

1. **将数据集划分：**训练集/验证集

```python
# 自动拆分比例
VAL_RATIO = 0.2
SEED = 42

# 类别名（必须与标签 id：0/1/2 对齐）
NAMES = {
    0: "bicycle",
    1: "e-bike",
    2: "car",
}
```

2. **对图片进行必要预处理**（如统一输入尺寸 640×640）。
3. **进行数据增强以提升鲁棒性**
   - **几何增强：**随机缩放/裁剪、随机翻转
   - **颜色光照增强：**亮度/对比度/饱和度扰动
   - **模糊噪声增强：**提高鲁棒性

```python
# ========== 几何增强 ==========
A.HorizontalFlip(p=0.5),  # 随机水平翻转

# 平移/缩放/旋转（不会改变图片大小，空白区域用黑色填充）
A.ShiftScaleRotate(
    shift_limit=0.05,       # 平移范围（占宽/高比例）
    scale_limit=0.2,        # 缩放范围
    rotate_limit=10,        # 旋转角度范围（±10度）
    border_mode=cv2.BORDER_CONSTANT,
    value=0,
    p=0.7
),

# 随机裁剪再缩放到固定大小（模拟目标远近变化）
A.RandomResizedCrop(
    size=(img_size, img_size),
    scale=(0.8, 1.0),
    ratio=(0.9, 1.1),
    p=0.5
),
```

```python
# ========== 颜色/光照增强 ==========
A.ColorJitter(
    brightness=0.2, contrast=0.2, saturation=0.2, hue=0.05, p=0.7
),
A.RandomBrightnessContrast(p=0.3),

# ========== 模糊/噪声增强（提高鲁棒性）==========
A.GaussianBlur(blur_limit=(3, 5), p=0.15),
A.MotionBlur(blur_limit=5, p=0.15),
A.GaussNoise(var_limit=(10.0, 40.0), p=0.2),
```

数据增强代码展示

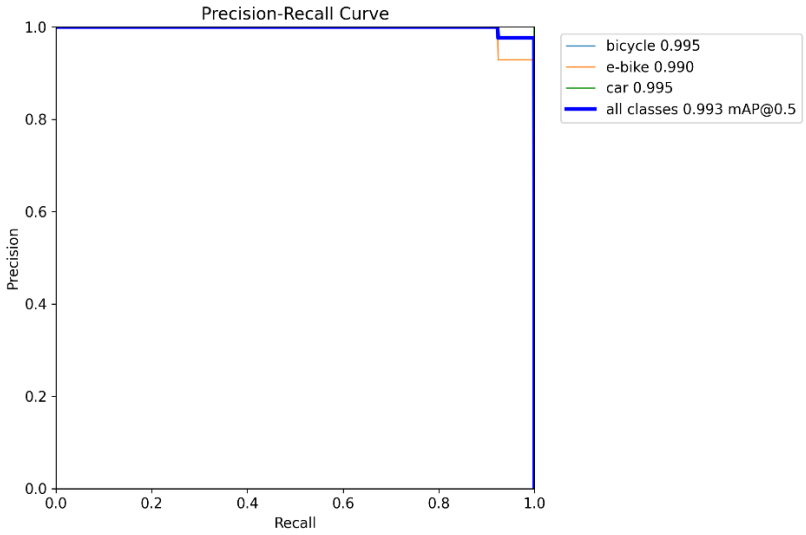
3.5 训练模型

```
results = model.train(
    data=str(dataset_yaml),

    epochs=EPOCHS,
    imgsz=IMGSZ,
    batch=BATCH,
    device=device,
    workers=4,
    cache=True,
    patience=PATIENCE,

    optimizer="AdamW",
    lr0=0.002,
    lrf=0.01,
    weight_decay=0.01,
    warmup_epochs=3,
    cos_lr=True,
    label_smoothing=0.05,

    # 小数据集建议冻结一部分层
    freeze=10,

    # 强增强（适合 60 张）
    hsv_h=0.02,
    hsv_s=0.7,
    hsv_v=0.4,
    degrees=10.0,
    translate=0.10,
    scale=0.50,
```

```
mosaic=1.0,
mixup=0.15,
copy_paste=0.10,
close_mosaic=15,

amp=True,
pretrained=True,

project=str(data_root / "runs_vehicle"),
name="yolo11_vehicle3_only",
```

3.6 迁移学习与模型评估

3.6.1 PR 曲线



总体效果：mAP@0.5（所有类别）= 0.993

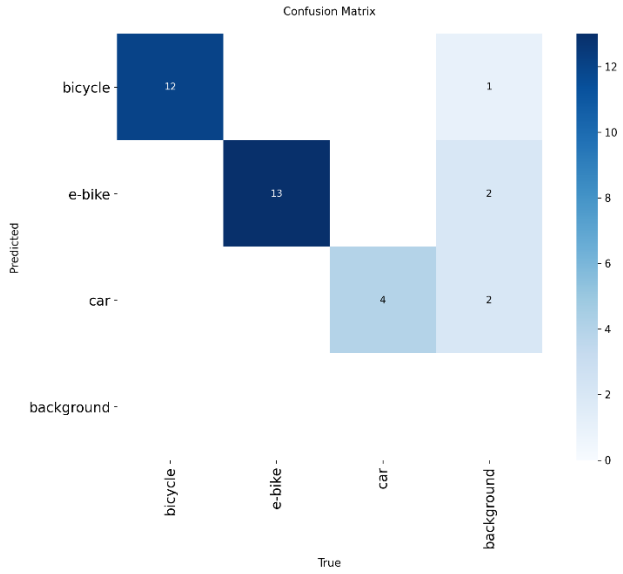
这说明模型在 IoU=0.5 的判定标准下，整体检测效果非常好，三类目标几乎都能稳定检测出来。

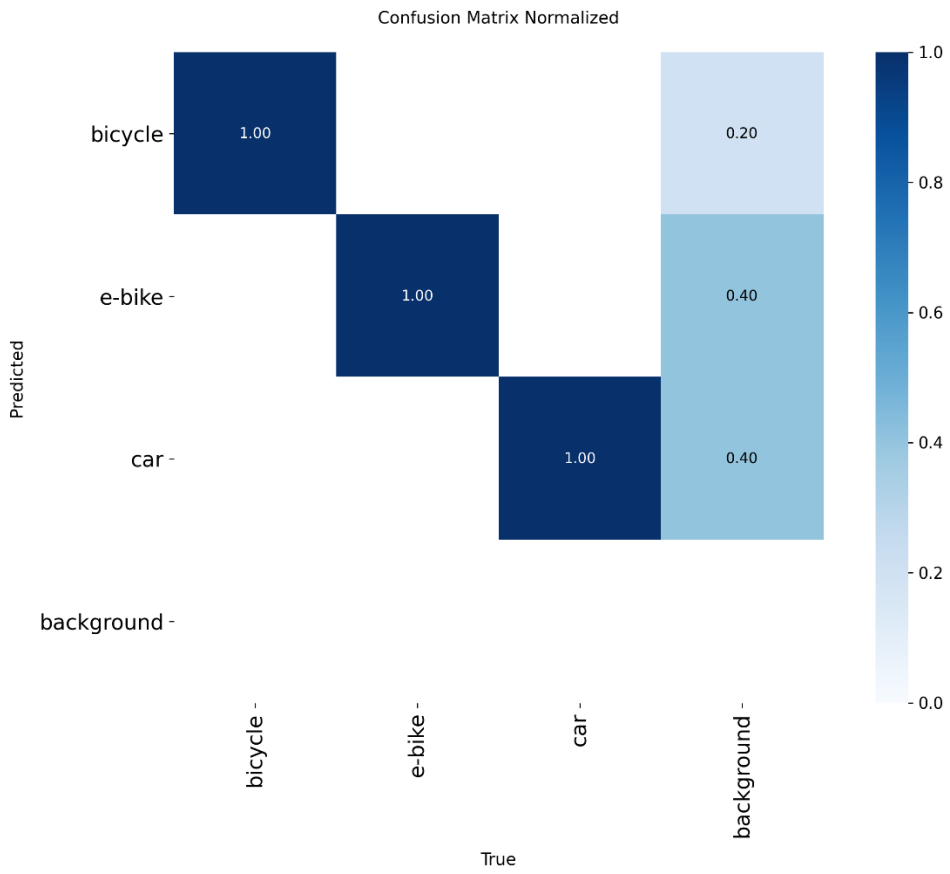
曲线基本贴着右上角走，说明 Precision 和 Recall 同时都很高。

e-bike 的 AP（0.990）略低于另外两类（0.995），说明电动车相对更容易出现：

- 少量误检（把背景/其他物体当电动车）
- 或者在某些难样本上置信度偏低、阈值提高后更容易漏检

3.6.2 混淆矩阵





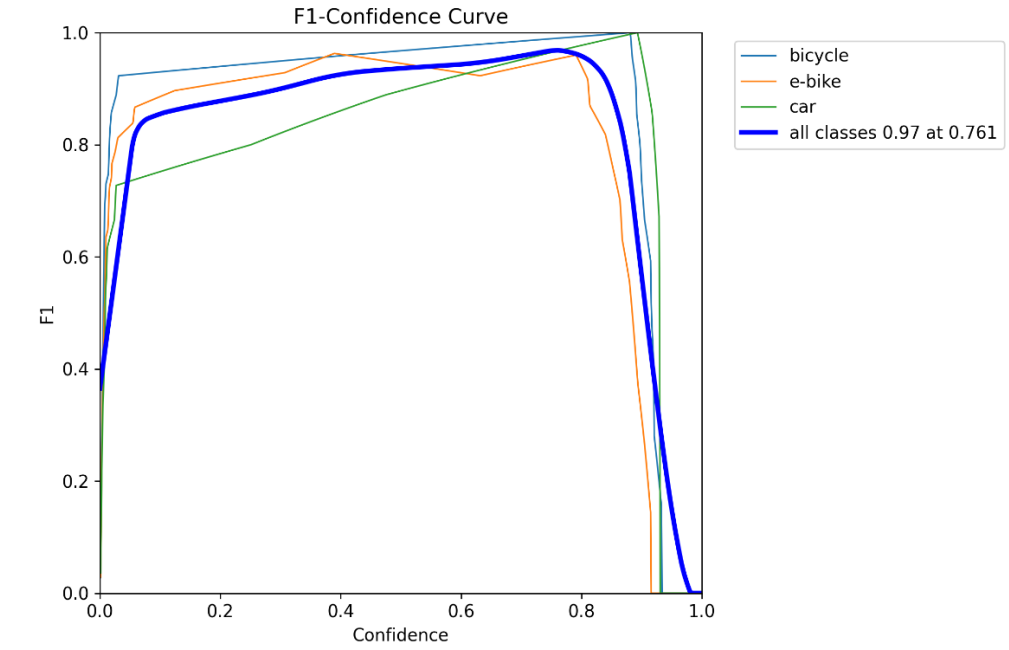
从混淆矩阵看：

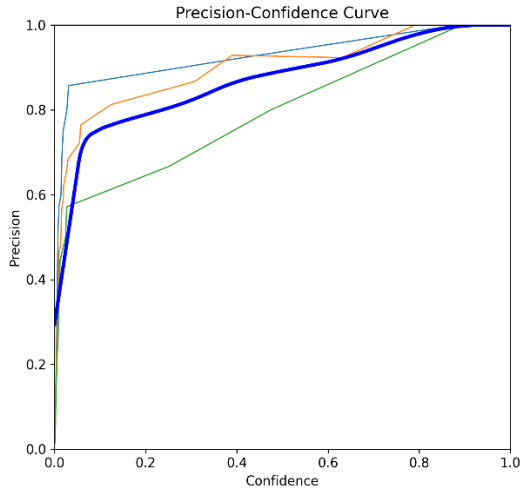
- **类间混淆几乎为 0**（例如 bicycle 基本不会被预测成 e-bike 或 car），说明模型已经学到了清晰的类别区分边界。
- **主要错误来自 background 列**（真实为背景，但被预测成某一类）：
  - bicycle：1 次误检
  - e-bike：2 次误检
  - car：2 次误检

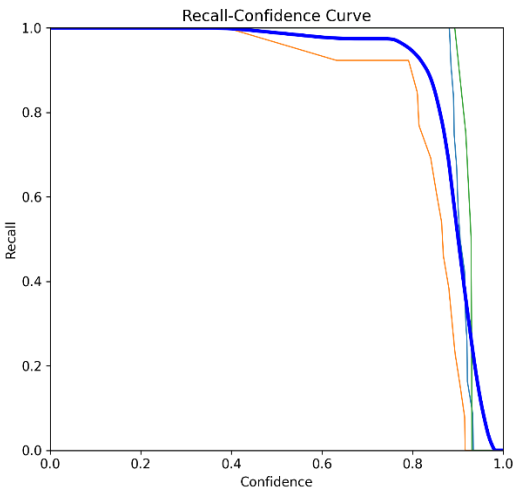
在归一化矩阵里，background 列对应比例是：

- 20% 被误判为 bicycle
- 40% 被误判为 e-bike
- 40% 被误判为 car

3.6.3 置信度相关曲线

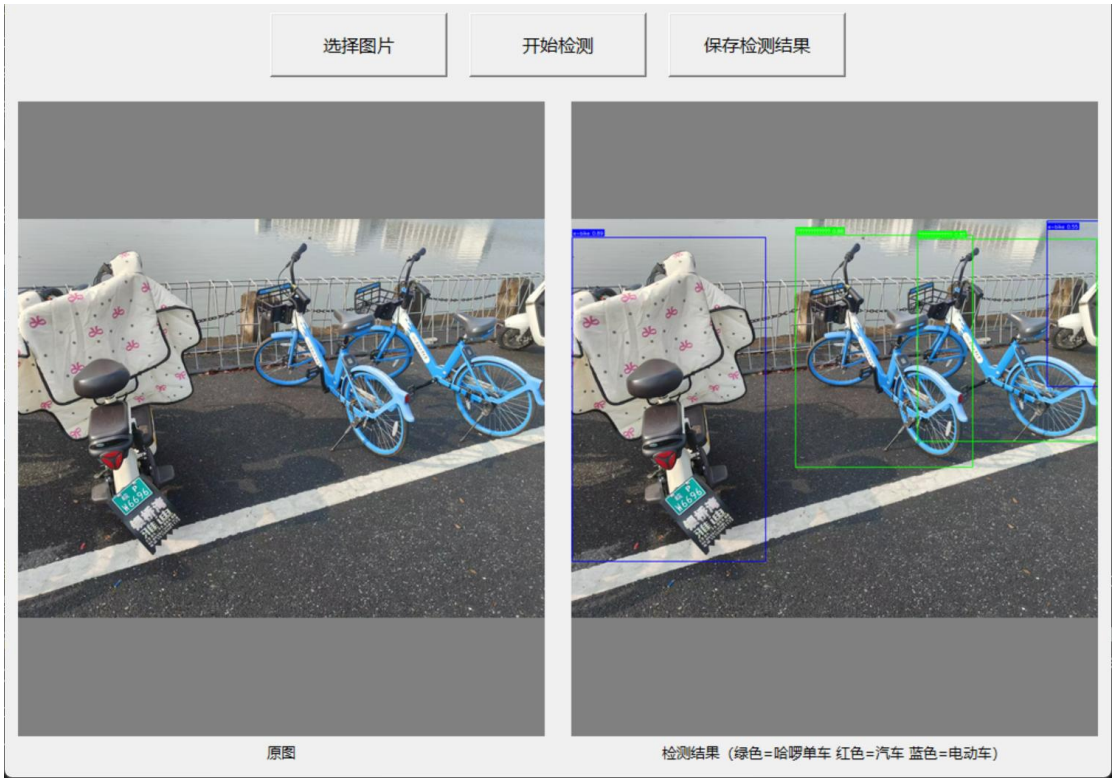






本实验训练得到的 YOLO 模型在测试集上取得了较高的检测性能，**整体 mAP@0.5 达到 0.993，各类别 AP@0.5 均接近 0.99。**PR 曲线基本贴近右上角，表明模型具有较高的准确率与召回率；类别间混淆较少，主要错误来源于少量背景误检。由 F1-Confidence 曲线可知，当置信度阈值约为 0.76 时综合 F1 最优（约 0.97），因此推理阶段将阈值设置在 0.75 左右可在精确率与召回率间取得较好平衡。

3.7 结果展示



区分电动车与哈啰单车（比基线好）

区分哈啰单车与汽车

四、心得体会

通过本次“校园共享单车检测”实验，**我对目标检测任务从数据到模型再到评估的完整流程有了更系统的理解**。本实验基于 YOLO 框架完成三类目标（bicycle、e-bike、car）的检测，最终训练结果在验证/测试阶段表现良好，PR 曲线整体贴近右上角，mAP@0.5 达到约 0.993，说明模型能够较稳定地完成“定位 + 分类”的联合任务，并具备较高的检测精度与召回能力。

实验过程中我最深刻的体会是：**目标检测的效果上限很大程度由数据质量与标签定义决定**。检测任务不仅需要区分类别，还需要准确回归目标边界框，因此数据的多样性（场景、角度、距离、光照、遮挡程度）以及标注的一致性（边界框是否贴合、类别是否统一）都会直接影响模型学习到的决策边界。尤其在 bicycle 与 e-bike 的区分上，二者在外观结构上存在相似性，若标注规则不够明确或存在“边界样本”（例如部分助力车/相似车型），模型容易在这类样本上产生置信度波动或误检。因此在数据集构建阶段，先建立清晰的类别定义与标注规范，并保持全数据标注一致，是提升模型性能与稳定性的关键。

在训练与调参方面，本实验让我理解了从“能训练出模型”到“能训练出可用模型”之间的差异。**Precision、Recall、mAP 等指标能够更直接反映模型泛化能力。通过对 PR 曲线、混淆矩阵和置信度相关曲线（Precision-Confidence、Recall-Confidence、F1-Confidence）的观察，我认识到推理阶段的参数设置同样重要：置信度阈值提高可以减少误检，但会带来漏检增多；阈值降低则提升召回但可能引入更多背景误检。**结合 F1-Confidence 曲线，本实验在置信度约 0.76 时取得较优的综合 F1（约 0.97），因此该阈值可作为实际部署的推荐取值；若应用场景更强调“少误报”，则可进一步提高阈值，但需要接受召回下降的代价。

此外，通过**混淆矩阵的分析**我发现，本实验模型在三类之间的互相混淆较少，说明模型已经学习到了较清晰的类别区分特征；主要误差更多来自背景误检（False Positive）或困难场景下的漏检，而不是“把 bicycle 识别成 e-bike”这类典型混淆。这一点提示我：后续提升性能的重点应放在困难样本与负样本补充上，例如远距离小目标、遮挡严重、夜间/逆光条件、密集停放或背景中存在相似结构物体的场景。针对这些问题，可以从以下方向改进：一是补充更丰富的

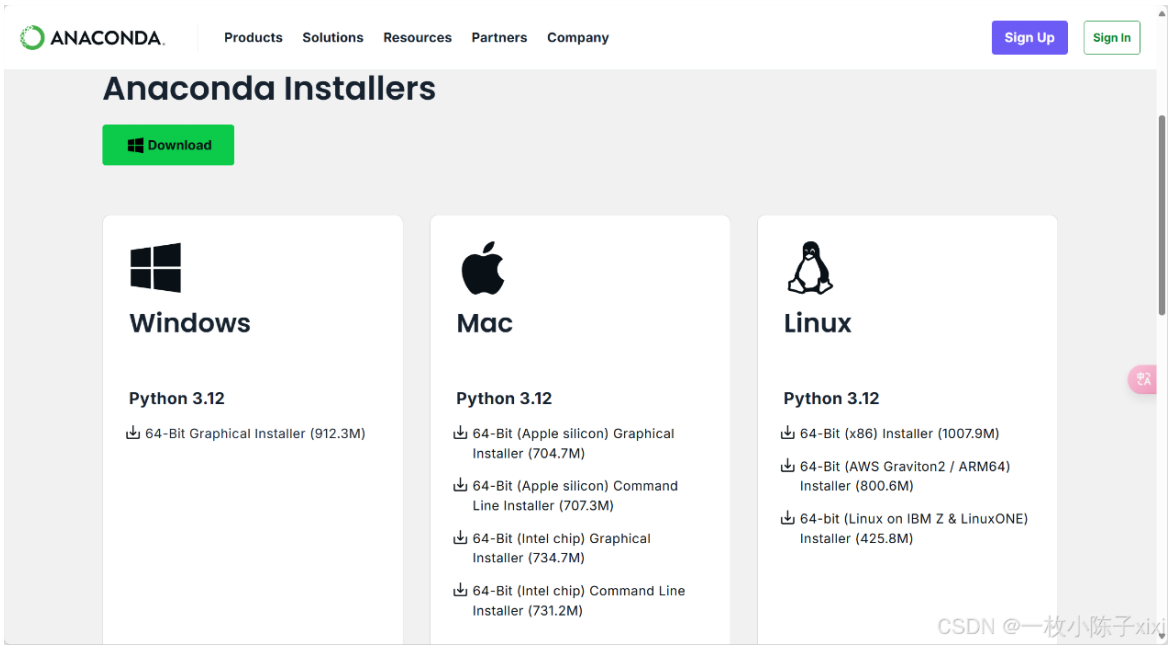
困难样本与纯背景负样本，降低背景误检；二是提高 car 类样本量（若存在类别不平衡）并增强多场景覆盖；三是优化数据增强策略，使模型对尺度变化、遮挡、光照变化更鲁棒；四是推理阶段结合阈值与 NMS 参数调节，以适配不同应用目标（高精度或高召回）。

总体而言，本次实验让我真正掌握了目标检测任务中“**数据—模型—评估—改进”的闭环思维**：不仅能够完成 YOLO 模型训练与测试，还能基于指标与可视化结果定位问题，并提出可执行的改进方案。这些经验对后续进行更复杂的检测与细粒度分类任务具有直接参考价值。

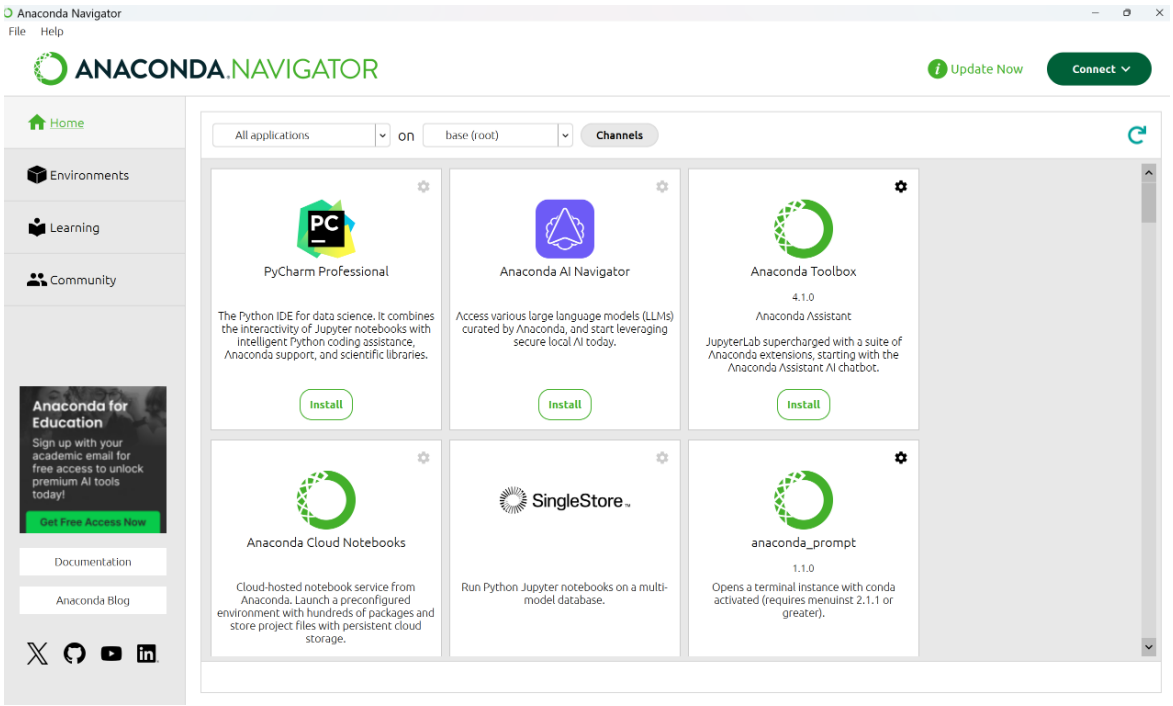
环境配置过程。

anaconda 安装与使用

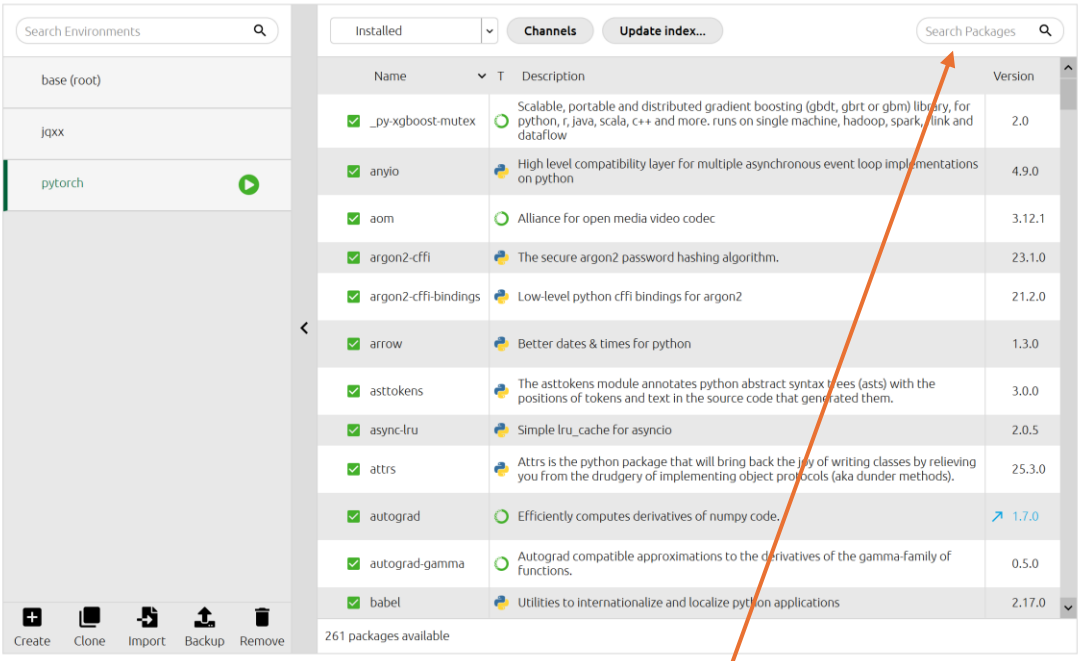
1、下载安装包



2、打开 Anaconda Navigator

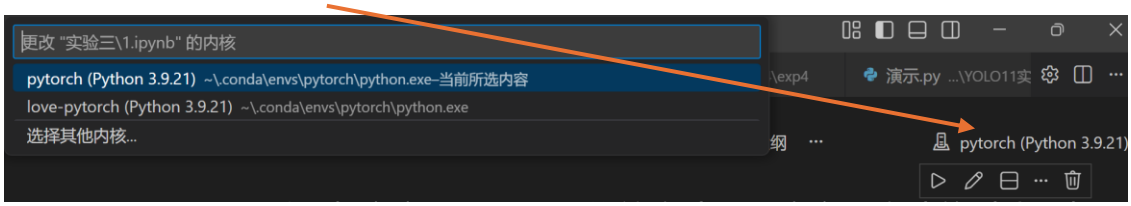


3、创建环境



这里我创建的是名字为 pytorch 的环境，并且可以在右上角搜索要下载的包，如果需要下载什么包可以自由下。

4、打开 vscode 并且切换环境



5、可以在 jupyter 里敲代码了