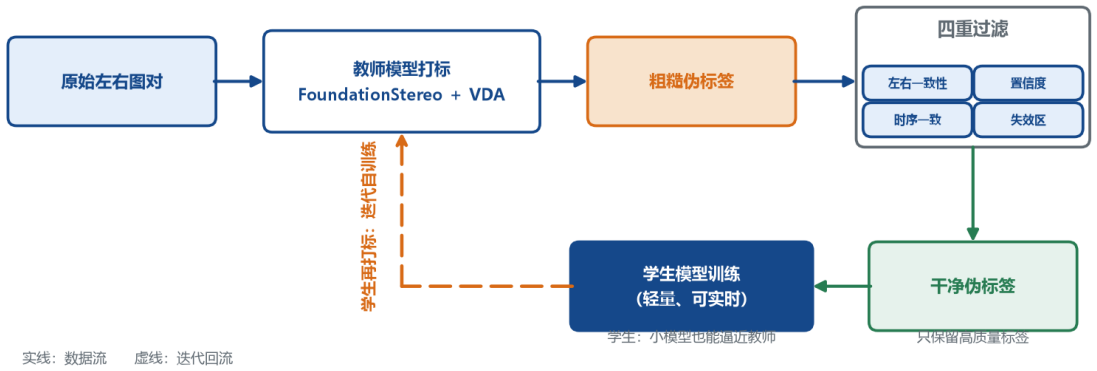


图 2.10　教师–学生伪标签闭环：大模型教师逐帧打标，多重过滤筛掉可疑标签，学生模型在幸存标签上训练；学生变强后回头重新打标，数据质量螺旋上升

第一步，**教师打标**。我们请两位“老师”给每一帧打分：**立体教师** FoundationStereo（见下面的前沿模型一览表，给逐像素视差和置信度）+ **视频深度教师** VDA（Video Depth Anything，单目深度大模型的视频版，深度序列在时间上更稳定）。两位老师各批一份作业，正好互相校对。

第二步，**四重过滤**。老师的作业不能照单全收——恰恰相反，**过滤是整条流水线的灵魂**。一条伪标签想活下来，得过四关：

1. **左右一致性**：用左目视差图把左图“搬”到右目，再算一次反向视差，两者应当互相吻合——对不上，说明匹配找错了对象。
2. **置信度**：教师自己会输出“这像素我有多确定”，低于门槛的直接淘汰。
3. **时序一致性**：视频是连续的，相邻帧的深度不该跳变——前后帧对不上号的，可疑。
4. **失效区剔除**：水体背景、无纹理区、OSD字符这些“本来就算不出深度”的区域，整块划掉，不许混进训练集。

第三步，**学生迭代**。幸存的伪标签拿来训练学生模型（学生正是我们要交付的、比教师小的网络）；学生训练好了，回头**重新给数据打标**——学生比原教师更懂本场景，打出的标签更干净，再过滤、再训练……循环几轮，**数据质量与学生水平螺旋上升**。这就是**自训练**（self-training），也称自我蒸馏。

2.4.3　最大的先例：NVIDIA 用 140 万对伪标签蒸出快 10 倍的学生

这条路线绝不是我们的发明，而是 2024–2026 年间业界公认的“阳谋”。最有分量的先例来自 NVIDIA：Fast-FoundationStereo（CVPR 2026）用“立体教师 + 单目深度教师 + 开放词表分割（划出天空等失效区）”给海量真实视频打伪标签，一致性校验后拿 **140 万对**伪标签蒸馏学生，得到比原模型**快 10 倍**、精度几乎不掉的实时版。看它的流水线图，与我们的设计思想如出一辙：

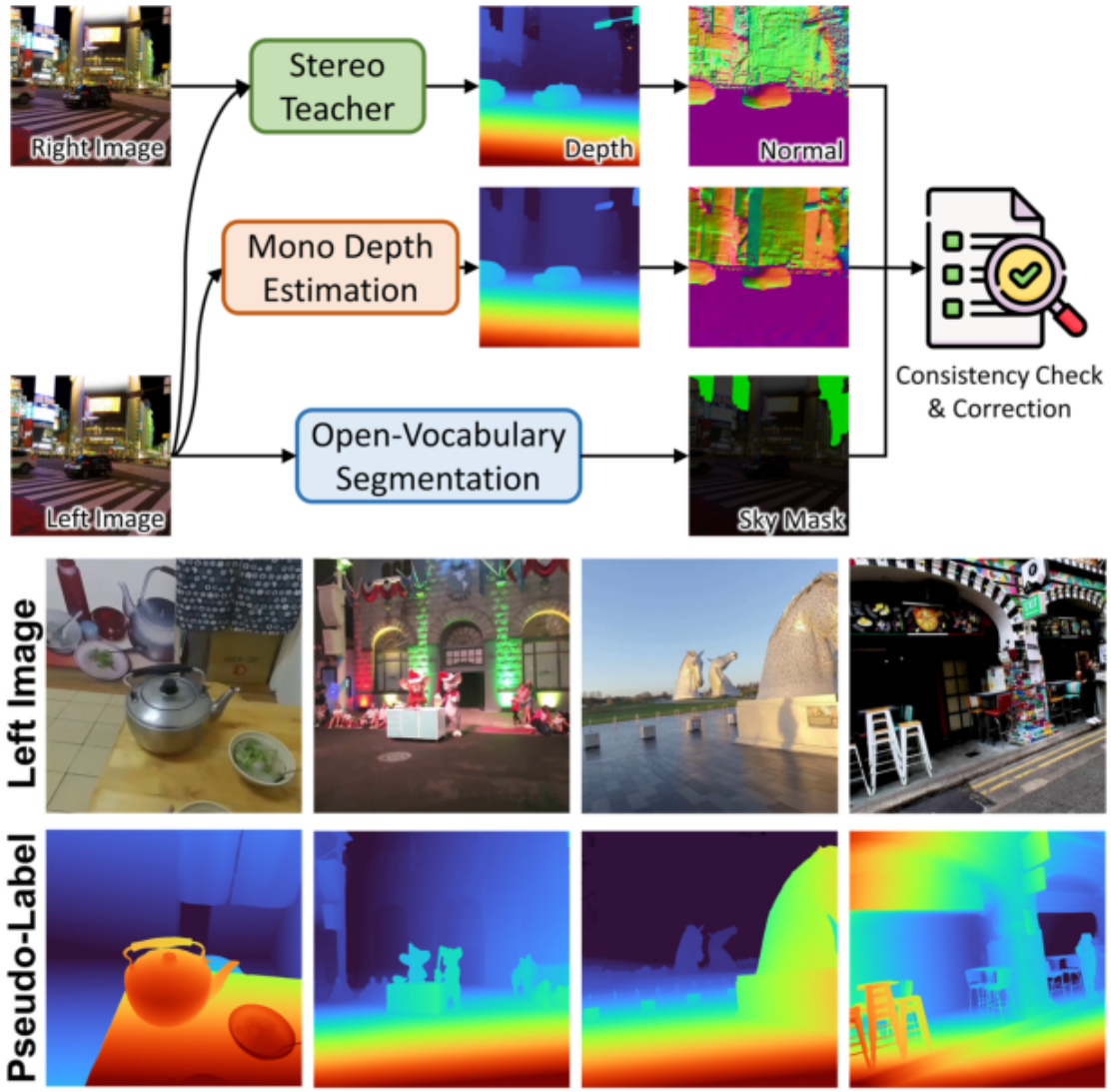


图 2.11　Fast-FoundationStereo 的伪标签流水线：左右图像分别经立体教师与单目深度教师打标，开放词表分割划出天空等失效区，一致性校验与修正后成为伪标签（下：多样真实图像与打出的伪标签）

图片来源：NVIDIA Fast-FoundationStereo 论文配图（CVPR 2026）

再往前追，单目深度大模型 Depth Anything V2 已经把“伪标签规模化”玩到极致：先用合成图训练出最大的教师，再用教师给 6,200 万张无标注真实图像打伪深度，最后用这些伪标签训练出各尺寸学生模型——见下图的三段式。**没有真值也能造数据**，它是最有力的证词。

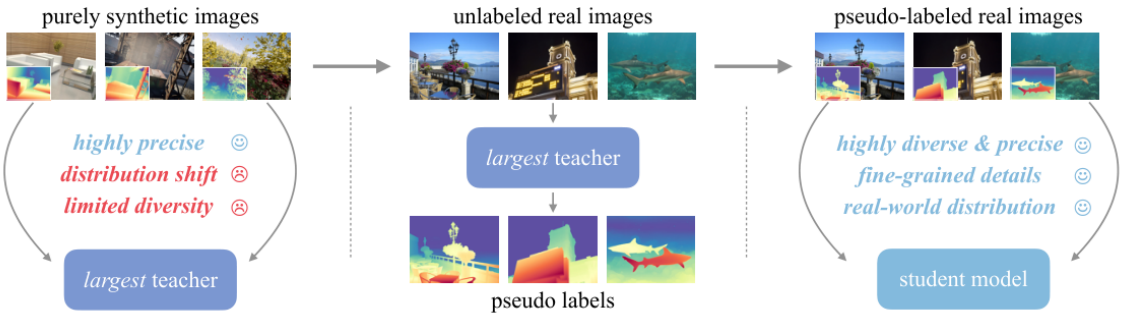


图 2.12　Depth Anything V2 的三段式蒸馏：合成图（精确但域窄）训练最大教师；教师给海量无标注真实图像打伪标签；伪标签（多样且精确）训练各尺寸学生模型

图片来源：Depth Anything V2 论文配图（Yang et al., 2024）

2.4.4　五条路线与前沿模型

伪标签不是唯一的路。下表把当前五条主流路线摆在一张表里，本项目走的是最后一条：

表 2.1　“没有真值”的五条主流路线对比

路线	代表	核心思路	短板
合成渲染	UE5 + EasySynth、UW-Stereo	游戏引擎渲染“自带真值”的水下世界	量大管饱，但与真实水下有域差距
单目伪标签	Depth Anything V2	单目深度大模型给真实帧打伪深度	尺度不确定，缺左右一致性约束
视频伪标签引擎	Fast-FoundationStereo	立体教师 + 单目教师 + 分割过滤，规模打标	算力门槛高
单目先验 × 立体	Stereo Anywhere、MonSter、DEFOM-Stereo	把单目大模型先验“嫁接”进立体匹配	依赖先验的质量与鲁棒性
自训练迭代（本项目）	AutoLabel-UW 流水线	双教师打标 + 四重过滤 + 学生重打标迭代	迭代要防“学偏”，过滤阈值需调优

这一领域 2024–2026 年的代表作，值得你每个都混个脸熟——它们不只是名词，后续实验里会真的出现在你的命令行里：

表 2.2　前沿模型一览（2024–2026）

模型	年份	一句话认识
FoundationStereo	2025	NVIDIA出品、CVPR 2025 最佳论文提名；当前最强零样本立体匹配基础模型——本项目的主线底座与“立体教师”
Depth Anything V2	2024	用 6,200 万张伪标签单目图像训练的单目深度大模型；单目伪标签路线的代表作
Video Depth Anything	2025	单目深度大模型的“视频版”，深度序列时序稳定——本项目第二位教师（VDA）
Fast-FoundationStereo	2026	NVIDIA 用 140 万对伪标签蒸馏出快 10 倍的学生；伪标签范式最大先例（CVPR 2026）
Stereo Anywhere	2025	单目先验与立体匹配“双分支”融合的代表
MonSter	2025	让单目与立体“互相救赎”：单目补纹理缺失，立体稳尺度
DEFOM-Stereo	2025	把单目基础模型（DINOv2）塞进立体迭代优化的代表

最后交代我们自己的主线算法与战绩。毕设生与研究助理在张洪进带领下，把三项传统数学方法——匹配代价泛函分析（DCFA）、模糊图关系（FGR）、置信度后处理（CFP）——改写成**可微模块**，嵌入 FoundationStereo，让经典数学与基础模型各补对方短板（其中 CFP 正对应大图景里的“每个像素有多可信”）。训练与验证在 UWStereo——目前最大的公开合成水下立体数据集（训练 26,612 对 + 测试 2,956 对，约 71 GB）——上微调后，视差误差 EPE 约 1.39 像素。这套工作支撑了一篇在投论文（Image and Vision Computing）与 2026 年青年基金申请。你们四个组的公开数据干跑与对照实验，就是在这套主线的延长线上干活。

知识补齐：EPE 与 bad-N——视差怎么算“错了几个像素”

评价视差/深度预测，通篇就两个指标。**EPE**（End-Point Error，端点误差）：预测视差与真值视差逐像素求差的绝对值，再对全图取平均，单位是像素——

$$\mathrm{EPE} = \frac{1}{N}\sum_{i=1}^{N}|\hat{d}_i - d_i|,$$

人话：全图平均“错几个像素”，越小越好（我们微调后约 1.39）。注意第 2.3 节的反比曲线：**视差错 1 个像素，近距离的深度误差会被放大**——所以 1.39 像素不等于“差 1.39 米”，深度误差还要看远近。**bad-N**：视差误差超过 N 像素的像素占全图的比例（常看 bad-1 / bad-2 / bad-4），衡量“错得离谱的部分”有多少。第 12 章评测组的指标白皮书，就围绕这两个数展开。

前沿速递：水下立体感知 2024–2026 与“无真值域”方法论

三件事值得记住。其一，**数据集**：UWStereo（最大公开合成水下立体数据集，约 71 GB）已收录为标准测试场；真实侧有 FLSea（海上真实双目）与 SQUID（水下真实立体对）——本项目四个组用这三个公开数据集做扩量与对照，真实数据（500 对）已在资产包里，原始视频依旧按保密红线管理。其二，**零样本能力**：FoundationStereo 这类基础模型在从没见过的场景上直接推理就有可用精度，“零样本”（zero-shot）正把三维视觉从“一个场景训一个模型”推向“一个模型吃遍天下”；但第 2.2 节的四道坎决定了水下仍需微调——这正是你们的活儿。其三，**范式**：“没有真值的领域，用基础模型教师打伪标签 + 一致性过滤 + 学生自训练”已经是机器人界的通用方法论——水下、矿山、仓储、深空，逻辑通用。学会它，你带走的不是一个项目，是一把万能钥匙。

项目连线：自动打标签，四个组各守一环

本项目：这条流水线把四个组焊在一条链上——组 1 造出能打标的数据，组 2 让教师在真实 GPU 上跑起来，组 3 训练学生并守住“零样本 → 微调”对照，组 4 用 τ（过滤阈值）扫描和误差分桶回答“标签能不能信”；毕设生的 DCFA/FGR/CFP 可微化是学生的“内功”。**六目拼接**（第 一 章）：它的“九个参数头”是“老经验 + 新网络”的混合体，与“传统数学方法可微化嵌入基础模型”是同一个思想的两副面孔。**数据基座**（第 三 章）：那边用大模型给文档打结构化标签、用门控过滤幻觉，这边用大模型给视频打深度标签、用一致性过滤伪标签——“教师打标 + 过滤”跨域通用。

2.5　我们的 500 对数据集：一次真实的翻车与自救

讲完方法论，讲我们自己的数据——这一节是真实工程叙事，有翻车、有自救，比成功更值得读。甲方给的是几个硬盘的检查视频（就是本章开头那种带 OSD 遥测的 ROV 画面）。第一步干什么？不是训练，是**给视频做体检**：搞清每段视频“动了没有、拍了什么、能不能用”。

2.5.1　翻车：八个“典型场景”全军覆没

最初我们挑了 8 段看起来最典型的视频段，信心满满开始构建。一算指标，傻眼：这些段几乎全是 **ROV 定点悬停段**——ROV 悬停在原地看链条，相机几乎纹丝不动，画面里像素的平均位移（净光流）不到 1 像素。人话翻译：**双目测距靠“从两个位置看同一物体”，相机不动，两个位置就重合成一个**——等于你睁眼闭眼看到的画面毫无差别（回顾动手盒子），基线近似为零，深度在数学上不可解。八个“典型场景”，一个能用立体数据的都没有。教训：**挑数据不能看“画面好看”，要看“几何有用”**——这个反直觉，就是数据组实验一要你亲手体会的。

2.5.2　自救：两条硬指标把数据“挖”出来

改进后的筛选是双重的：第一，**活跃度 $\geq$ 2 像素/秒**——ROV 得沿链移动，画面有足够的净运动，立体基线才“长得出来”；第二，**剔除带底部遥测 OSD 的画面**——OSD 是后“盖章”上去的字符，不是真实场景，会污染匹配与训练。两条过一遍，终于从视频里挖出 8 个可用的源窗口，清一色锚链场景：链环特写、海生物、藤壶、卸扣（连接链环的“大销子”）。

零基础扫盲：OSD——画面上的“盖章”字符

OSD（On-Screen Display，屏上显示）是设备把遥测数字（深度、坐标、时间戳）直接“印”在视频画面上的功能，常见于行车记录仪与监控。对人是信息，对算法是污染：它随屏幕而动、不随世界而动，网络若把它当真实场景学进去，学到的是“数字水印的排布规律”。所以本项目两条铁律：**筛选时剔除带 OSD 的帧**；**训练时把 OSD 区域划为失效区**。

2.5.3　没有标定板的自标定

有了画面，还差相机的“体检参数”：焦距 f、畸变系数。水下 ROV 不可能给你架标定板，怎么办？**自标定**：用 `pycolmap` 对视频帧做全局 SfM（Structure from Motion，运动恢复结构——从画面自身的运动反推相机参数与场景结构），解出焦距 $f_x = f_y = 925.9$ 像素，重投影误差 0.62–0.71 像素（人话：反推的参数把所有特征点“放回”画面时，平均偏不到一个像素——参数可信）。

2.5.4　一个价值 7 倍的工程决策

更隐蔽的坑在畸变。ROV相机多为广角，画面边缘有明显的镜头畸变（直线被拍弯）。实测发现：**如果在带畸变的原始像素坐标上直接做几何计算，画面旋转会被“揉”进视差里，把视差虚高约 7 倍**——深度直接错 7 倍，全盘皆输。团队因此定下一条决策：**全部几何在去畸变坐标上进行**——先把每帧图像“掰直”到无畸变的理想坐标，再谈校正、再谈视差。这类决策教科书不会写，只有亲手翻过车才会懂。

2.5.5　验收：数字说话

最终交付的数据集长这样：目录 `stereo_dataset` 下，每对样本含 `left.png`、`right.png` 与一份 `meta.json` 元数据，另有总账 `manifest.csv`。产出经手写 Bouguet 校正（含 Q 矩阵）共 507 对，组装交付 **500 对校正立体图对**。质量验收按下表执行——每组一个硬指标、一个门槛，全部达标才算交付：

表 2.3　stereo_dataset 质量验收单（全部通过）

指标	数值	门槛与说明
可用源窗口	8个	双重筛选后全部为锚链场景（链环特写/海生物/藤壶/卸扣）
校正立体图对	507 对 → 交付 500 对	手写 Bouguet 校正 + Q 矩阵
垂直极线误差（中位数）	0.254 px	门槛 < 1.0 px，对应第 2.3 节的校正质量
视差（中位数）	19.2 px	场景目标可测深，供 $z = bf/d$ 换算
匹配内点（中位数）	492	门槛 ≥ 100，衡量左右图可配对的稳健程度
自标定焦距 $f_x = f_y$	925.9 px	`pycolmap` 全局 SfM，重投影误差 0.62–0.71 px

看看辛苦换来的数据长什么样。下图是 seg0 窗口的校正左目图像：一根长满海生物（白色斑点多为藤壶壳体）的锚链沿对角线横穿画面，背景是近乎纯色的深蓝海水，右缘还残留少量 OSD 字符——低纹理、弱对比、杂字符，水下立体匹配的“地狱难度”一眼可见：

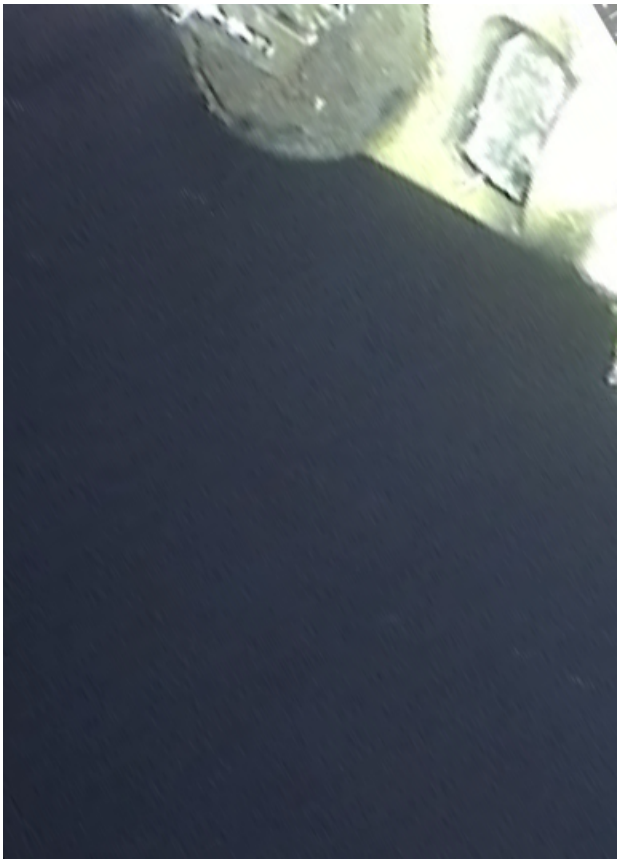


图 2.13　校正后的左目图像（seg0 首帧）：长满海生物的锚链沿对角线横穿画面，背景近乎纯色，右缘残留少量 OSD 遥测字符——低纹理、弱对比的典型水下场景

图片来源：本项目 stereo_dataset（E:\锚链视频\stereo_dataset，seg0 样本）

下图是 seg2 窗口：镜头贴近锚链部件拍摄，画面上部可见黄褐色的金属链环/卸扣，大部分区域

沉入阴影——光照不均也是水下常态。两幅图的“校正”效果，就看同一根链条在左右图中是否处于同一行：



图 2.14　校正后的左目图像（seg2 首帧）：镜头贴近锚链金属部件（链环/卸扣）拍摄，上部可见黄褐色金属与海生物附着，大面积阴影展示水下光照不均

图片来源：本项目 stereo_dataset（E:\锚链视频\stereo_dataset，seg2 样本）

最后一张，是从原始视频里做“体检”时的抽帧联系表：每 10 秒取一帧、共 36 帧。亮色的锚链在画面中缓缓摆动，OSD 里的深度读数从约 185.5 m 缓慢变到约 187.5 m、离底高度从约 11.8 m 降到约 10 m——运动有了、场景清楚了，“活跃度筛选”挑的就是这样的段：

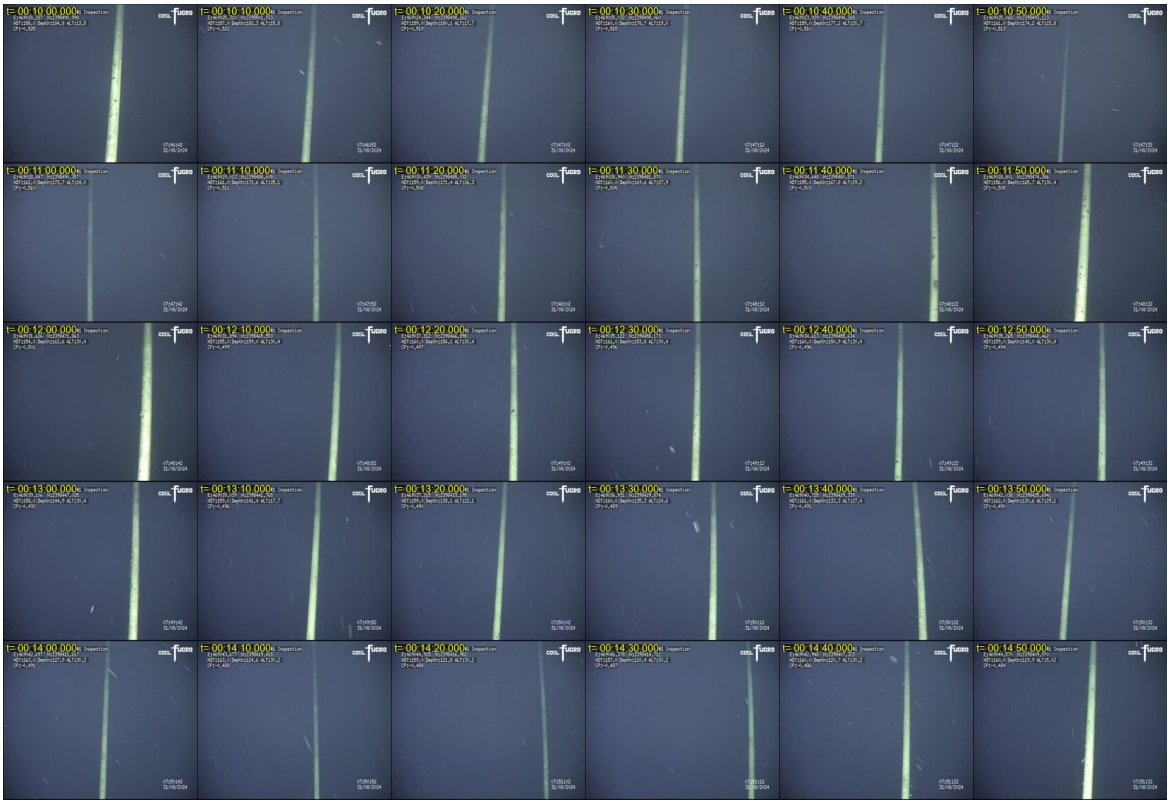


图 2.15　检查视频抽帧联系表（每 10 秒一帧，共 36 帧）：锚链亮条在画面中缓缓移动，OSD 遥测中深度约 185.5–187.5 m、离底高度约 10–12 m 缓慢变化——“视频体检”量化的正是这些运动与场景信息

图片来源：本项目分析产物（E:\锚链视频\analysis）

项目连线：500 对数据从哪来、到哪去

本项目：本节就是组 1 数据组的“前传”——视频体检、抽帧去冗、双目预处理、数据集组装四步，第 9 章给了命令级的实验安排；前期最小实现已把这套流程跑通：500 对真实图对已在资产包里，四个组从第一周就在真实资产上工作；UWStereo/FLSea/SQUID 用于扩量与对照实验。**六目拼接**（第 一 章）：同样吃过“画面好看但几何没用”的亏——全景拼接也要求相机有足够视差，两个项目对“运动”的理解殊途同归。**数据基座**（第 三 章）：manifest.csv 总账、meta.json 元数据、验收门槛表——“先建台账、再谈智能”的数据治理纪律，两边一模一样。

2.6　四个组怎么分工：四条并行赛道，实验四会师

方法有了、数据有了，剩下的是组织问题：4个本科生组 + 1 名毕设生，怎么把上面这些活儿分掉？先交个底：前期的最小实现已经完成——FoundationStereo 权重在 4080 上部署可跑、500 对锚链校正立体图对就绪、用 UWStereo 初步微调的基线模型 v0 已产出。所以组织方式不是接力，而是**四条并行赛道：组 1 管粮草、组 2 管炉子、组 3 管炒菜、组 4 管试菜**，共享同一份资产包，实验一至三各自离线自研、谁也不等谁，实验四（12 月 21 日）合并联调。一句话各说清：

表 2.4　四个小组的分工与四次实验（第 5–19 周，每周 6–8 小时）

组别	一句话使命	四次实验（一至三离线自研，四联调）	人工必做
组 1 数据组	把海量视频变成能训练的数据集	① agent 配置 + 视频体检脚本；② ffmpeg / PySceneDetect / OpenCV 抽帧去冗工具链；③小规模微调亲眼看 loss 下降；④数据配比对比	抽 30 帧亲眼核对体检指标
组 2 部署组	让大模型在真实 GPU 上跑起来、跑得快	① 4080 环境清单；② FoundationStereo 权重部署 + `run_demo.py` + UWStereo 50 对零样本冒烟 + 降级链 vitl → vits → IGEV++；③显存占用曲线与吞吐；④批量/分辨率/fp16 对比 + ONNX 导出初探	亲手装机、实录环境坑
组 3 训练组	从零样本基线到 UW-Stereo 微调复现、超参优化	①配置说明书；②数据加载冒烟；③复现“零样本 → 微调”对照，画 loss/EPE 曲线；④学习率/增强对比 2–3 组	epoch 闸门必须经小助手批准
组 4 评测组	给所有模型打分：EPE/bad-N → 误差分桶 → 可视化 → 阈值扫描	①指标白皮书 + 手工验证 1 对；②评测全链路；③每模型 ≥ 50 对建基线数字表；④伪标签过滤阈值 τ 扫描 + 分桶归因	看误差可视化 $\geq$ 20 张，写“错在哪”

看懂这张表的关键是**并行不乱套**：四个组开局就拿到同一份资产包（FoundationStereo环境、500 对图对、UWStereo 子集、零样本基线与初步模型 v0），实验一至三各干各的离线自研，产出仍然互为嵌入：组 1 的数据集是组 3 的粮、组 2 的环境是全队的地基、组 4 的指标给组 3/组 1 判卷——但谁也不必等谁先交付。特别点两处“人工闸门”：组 3 的 epoch 推进必须经小助手批准，组 4 必须亲眼看至少 20 张误差可视化并写归因（AI 不可代看）。到了实验四，四条赛道在同一台机器、同一套考卷上会师：新配比接入重训、新权重补测、统一口径总评测干活。特别点两处“人工闸门”：组 3 的 epoch 推进必须经小助手批准（防止过拟合了还傻训）；组 4 必须亲眼看至少 20 张误差可视化并写下“错在哪”（AI 可以算指标，不能替你长眼睛）。

纪律方面，本项目有四条红线（下方红盒），外加一个固定节点：**第 19 周周五（2027 年 1 月 22 日）全员离场，四组合并提交《四组交接总表》**——数据、代码、模型、指标数字表、遗留问题，一项都不能少。每组详细的任务书、命令级步骤与评分表，见第三篇第 9 章至第 12 章。

2.7　回头看：你将用到第一篇的哪些本事

这门课把项目编排在第一篇之后不是偶然——本项目的每一步，几乎都能在第一篇找到“那块积木”。下表是对照清单，学习第一篇时请带着右栏的问题去学：

表 2.5　第一篇知识在本项目中的落点

第一篇	讲什么	在本项目哪里用上
第 5–6 讲	卷积与 CNN 前向/训练	立体匹配网络的“眼睛”：特征提取靠卷积；微调就是第 6 讲的训练循环
第 7 讲	CNN 演化与基础模型骨干	FoundationStereo 的骨干源自这条演化线：懂骨干才看得懂结构图
第 8 讲	轻量化、部署与 FCN	组 2 的日常：fp16、ONNX、显存/吞吐，全是“部署量化”的现场版
第 9 讲	序列与时序建模	四重过滤里的“时序一致性”：视频是序列，逐帧独立必丢分
第 11 讲	注意力与 Transformer	基础模型的骨干与代价聚合都靠注意力：看懂 Q/K/V 才读得懂论文图
第 12 讲	生成对抗与蒸馏	教师–学生伪标签闭环就是蒸馏思想的工程化；“教师打标”与 GAN 的博弈思想同理

还有一条贯穿始终的“软本事”：**与 AI 协作**. 四个组的实验都在 AI 小助手辅助下进行——但本项目把话说在前面：AI 产出的每一行代码、每一个数字，都必须实跑验证；看图、看曲线、写“错在哪”，必须你亲自来。这不是不信任 AI，而是分清“谁能替你算、谁必须替你看”。

数据保密红线：四条，一条都不能碰

本项目使用甲方（中海油）真实生产数据，以下红线写入每个组的任务书，违者按不及格处理：(1) **原始数据不上云**——原始视频、抽出的帧、构建的数据集，**一律不上传任何网盘、不投喂任何在线 AI 服务**，全部处理在本地离线环境完成；(2) **主线代码只读**——毕设生维护的算法主线仓库，本科生组只许读取调用，不许修改，改动一律走自己组的分支并留下说明；(3) **不直接联系甲方**——一切需求与问题经项目负责人张洪进对外沟通，学生不得私自对接；(4) **AI 产出必须实跑验证**——AI 给的代码/结论，跑不通、对不上就不算数，禁止把 AI 的话直接写进报告。红线之外，大胆试错。

本章结束前，把开篇那句话还给你：输入一对左右画面，输出每个像素的深度，并知道每个像素的结果有多可信。现在你知道这句话背后的全部代价——四道物理的坎、一条 $z = bf/d$、一套伪标签流水线、500 对翻车又自救换来的数据、四个组的接力。接下来，去第三篇领你的任务书：数据在第 9 章，算力在第 10 章，训练在第 11 章，评分在第 12 章。海底世界等着你们的 AI 去看懂。

ROV（Remotely Operated Vehicle，遥控潜水器）

由母船通过缆绳供电与通信、人在船上遥控的水下机器人；本项目的“摄影师”，负责下水检查锚链。

FPSO（Floating Production Storage and Offloading，浮式生产储卸油装置）

海上“炼油厂 + 仓库”：接收海底油井的原油就地加工储存；本项目的甲方场景是“海洋石油 123”号 FPSO。

锚链（mooring chain）

把 FPSO/平台拴在海床上的巨型铁链；断裂即大事故，故需 ROV 每年下水巡检——本项目全部数据的拍摄对象。

OSD（On-Screen Display，屏上显示）

设备“盖章”在画面上的遥测字符（深度、坐标、时间戳等）；对人是信息，对算法是污染，须剔除或划为失效区。

视差（disparity）

同一空间点在左右两幅校正图像中的水平像素差 d；近大远小，是深度计算的原料，也是标签本体。

基线（baseline）

双目两相机光心之间的距离 b；基线为零（相机悬停不动）则深度在数学上不可解。

立体校正（stereo rectification）

把左右图像重新投影到“理想水平摆放”的坐标系，使对应点落在同一行，把匹配从全图搜索简化为逐行搜索。

垂直极线误差（vertical epipolar error）

校正后真实对应点偏离同一行的垂直像素数；衡量校正质量，本项目验收门槛为中位数 < 1.0 px，实测 0.254 px。

EPE（End-Point Error，端点误差）

预测视差与真值视差逐像素求差取绝对值再平均，单位像素；本项目 UWStereo 微调后约 1.39。

bad-N

视差误差超过 N 像素的像素占全图比例（常看 bad-1/bad-2/bad-4）；衡量“错得离谱”的部分有多少。

零样本（zero-shot）

模型不在目标场景数据上做任何训练、直接推理；FoundationStereo 的看家本领，但水下仍需微调。

伪标签（pseudo-label）

用模型（教师）的预测当训练标签；不是真值，须经一致性过滤，是“无真值域”的核心数据来源。

知识蒸馏（knowledge distillation，教师–学生范式）

让小“学生”模型模仿大“教师”模型的输出；伪标签自训练正是其工程化——FFS 用 140 万对伪标签蒸出快 10 倍的学生。

UWStereo

目前最大的公开合成水下立体数据集（训练 26,612 对 + 测试 2,956 对，约 71 GB）；本项目训练对照与干跑的“标准测试场”。

项目三　数据基座治理：让海量工程数据开口说话

本章地图

讲什么：项目三的“总剧本”——甲方是“海洋石油 123”号 FPSO 相关业务方，手上有 126 份 CSV 共 3.1 亿行运行数据，外加 6 份共 1,001 个物理页的成套工程资料。本章讲我们如何建成“工程资料结构化与本地数据查询系统”（ERAG-ST-20260928 Rev.3）：让机器按结构读懂文档；让你一句话问数、每个数字能点回原件；让 AI 抽取的知识先过门控再入库、再用图谱查关系；以及一场识别虚假内容的攻防战——包括一次把 0.88 改判成 0.74 的诚实审计。**为什么值得学**：前两个项目“造”数据，这个项目“治”数据。它把第一篇的评估、向量表示、训练循环、部署、Agent、对抗思想全部串成一条流水线，还示范了工程里最稀缺的品质：如实。**学完你能**：说清“文档变数据、问题变 SQL、知识过门控、假货现原形”四件事；看懂一张验收缺口表；第 5–19 周带着三个学生组的任务书（第三篇第 13 章至第 15 章）进场。

先设想一个场景。你是“海洋石油 123”号 FPSO（Floating Production Storage and Offloading，浮式生产储卸油装置，第 二 章介绍过的“海上炼油厂”）相关业务团队的一名普通工程师。某天快下班，领导路过你的工位，丢下一句话：“把 2023 年 11 月 18 日 20 点 27 分前后，平台的所有信息整理给我。”就这一句话，你面前立刻立起两座山。

第一座山叫**运行数据**：126 份 CSV 文件、共 3.1 亿行——航行、装载、机械、摇荡，各种传感器一分一秒地记。想找“20 点 27 分”，你得先弄清哪份文件管哪个系统、时间戳是哪个时区、列名是什么意思，然后在 3.1 亿行里大海捞针。第二座山叫**工程资料**：成套的计算书与设备手册，阶段材料就有 6 份、1,001 个 PDF 物理页。领导要的“所有信息”有一半写在纸面里——某台泵的额定排量、某个舱室的设计参数，你得一页一页翻，翻到了再用手抄。

过去这道题的标准答案是三个动作：**人工翻页誊录**（取一张表要照着 PDF 敲进 Excel）、**逐行比对**（核对一个数字要两份资料来回对照）、**人工把关**（就算让 AI 抽取，抽出来的东西入库前还得靠老师傅一条条过目）。一个“20 点 27 分”，足够一个熟练团队加班一整晚。本项目要做的，就是让这道题变成一句话的事。系统有个朴素的名字：**工程资料结构化与本地数据查询系统**（ERAG-ST-20260928 Rev.3）。它由几条能力流水线组成，见下图——工程资料与运行数据从左边进，经过结构化、问数、门控与图检索，最后变成“随口问、秒级答、可追溯”的服务。

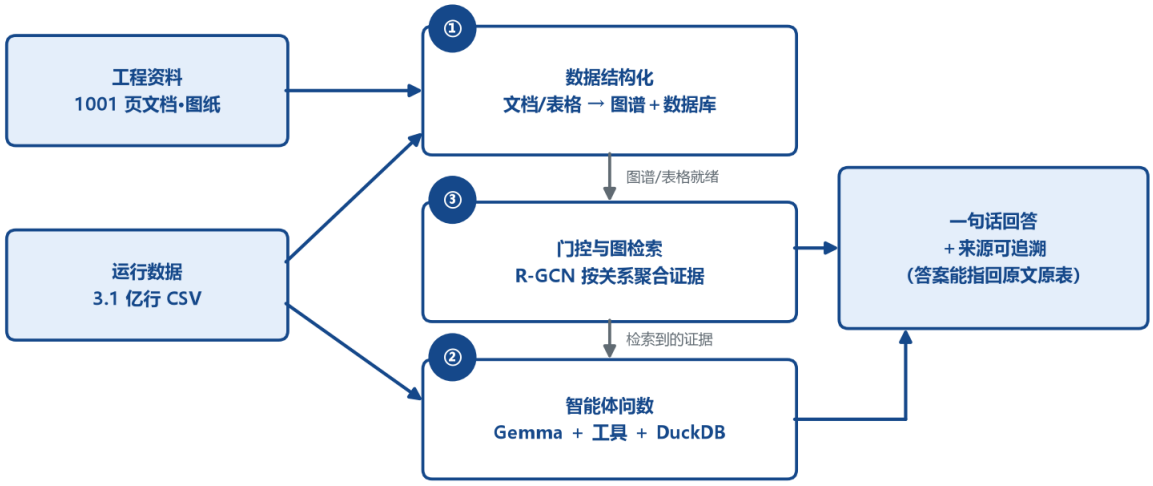


图 3.1　数据基座三大能力流水线总览：工程资料与运行数据先经“结构化”变成带锚点的表格、公式与片段，再由“智能体问数”变成一句话可查、每个数字可回溯的查询服务；“知识门控与图检索”为入库知识把一道关、把关系连成网；第四条独立的攻防流水线专门识别输入中的虚假内容——四条链各司其职，层层设卡

系统在技术上有六层：交互层（网页）→ 智能层（本地大模型 Gemma 加语义模型 Granite）→ 工具层（8 个固定工具）→ 计算层（SQLGlot / Wren / DuckDB 三道关卡）→ 数据层（CSV、Parquet 与只读原件）→ 知识层（门控与图检索）。正式环境跑在 8765 端口，测试环境跑在 8766——连端口都分开，演示数据永远混不进生产库。本章按流水线顺序逐层拆解：第 3.1 节先看清问题；第 3.2 节讲结构化；第 3.3 节讲一句话问数；第 3.4 节与第 3.5 节讲知识门控与图检索；第 3.6 节是本章高潮——假内容攻防战；第 3.7 节讲那次著名的“0.88 改判 0.74”审计；第 3.8 节讲三层身份与验收文化；第 3.9 节回头盘点你要用到的第一篇本领。

3.1　数据的洪水与资料的迷宫

先把两座山的“地质构造”看清楚，因为它们难的方式完全不同，解法也完全不同。

运行数据是“洪水”：量大、规整、没人看。3.1 亿行数据并不是乱码——CSV 嘛，行列分明——问题在于**量**：一个工程师一天最多核对几千行，3.1 亿行意味着几百个人年的工作量。而且这些数据过去“躺着”的姿态很不友好：126 份文件各自为政，列名不统一、时间精度不一，想跨文件问一个问题（“11 月 18 日晚上主机转速和纵摇角有没有联动?”），你得先自己把几份文件拼起来。

工程资料是“迷宫”：量不算大（1,001 页），但**结构**极其复杂。计算书里的表格动辄跨页，一列数据跨了页还要继续合并；公式不是图片就是扫描件，想抄进报告只能重新敲；图纸和“图 3-2”这样的标签散落在不同页；更坑的是，PDF 的页码还分“目录里的逻辑页码”和“文件里的物理页”——封面、扉页、目录加起来，逻辑第 12 页可能在物理第 20 页。人在这座迷宫里认路，靠的是记忆和经验；机器过去连“门牌号”都找不到。

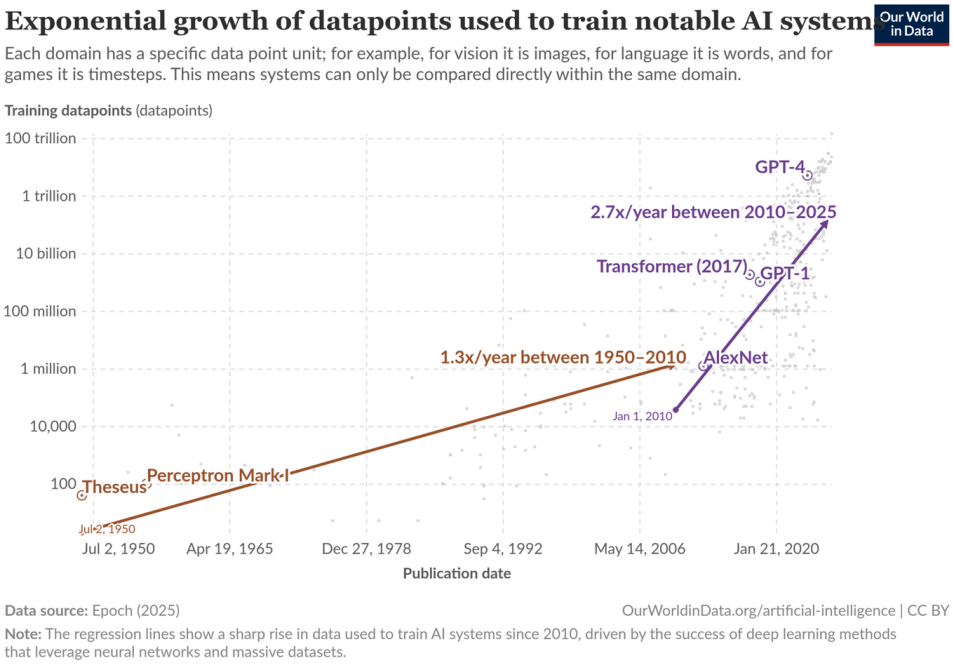


图 3.2　数据量增长示意：全球每年产生的数据量呈指数级攀升，工程行业“资料多到没人读得完”是普遍现状——本项目的 3.1 亿行只是这股洪水中的一滴水

图片来源：IDC Global DataSphere 全球数据量统计与预测（公开资料转载）

数据洪水不是海洋石油行业独有的。看上图，全球数据量近乎指数增长，而人类的阅读速度一万年没变过——**“数据多到没人读得完”会是你这一代人在所有行业里的常态**。出路只有一条：让机器替你读。但机器要能读，先得解决三个问题：第一，机器怎么把 PDF 里歪七扭八的表格**按结构**读出来（第 3.2 节）；第二，读出来的数据怎么让人**用一句话**查、查到的数字怎么保证不是模型编的（第 3.3 节）；第三，机器自己“读后感”式的知识产出，怎么防止它夹带私货（第 3.4 节）。

这三个问题对应系统的三类需求编号：**A 类**（查询与追溯）、**B 类**（文档结构恢复）、**C/D 类**（知识门控与图检索）。字母看着枯燥，其实是验收时的“军令状”——每类需求都挂着一串可判定、可复核的验收条款，这与前两个项目“先定指标再动手”的纪律一脉相承。

项目连线：一个甲方的三张考卷

三个项目共用同一个甲方背景（中海油海上油气业务），但考卷各不相同：**六目拼接**（第 一 章）处理**图像**，把六路相机画面拼成全景；**水下双目**（第 二 章）处理**视频**，把像素变成深度；**数据基座**（本章）处理**文档与表格**，把页面变成数据库。素材不同，方法论却惊人一致：都先把原始素材加工成“有身份、可追溯”的数据资产（全景图有拼接参数、立体图对有 meta.json、表格有锚点），再在上面谈智能。可以说，本章是你将来在任何一个行业做“AI 落地”时最先碰到的必修课——因为客户手里最不缺的就是文档和表格。

3.2　让机器“按结构读懂”：数据结构化

数据结构化对应 **B 类需求：文档结构恢复**。一句话说清目标：**把 1,001 页 PDF/Word 还原成“表格是表格、公式是公式、图是图、每一段都能指回原文”的结构化数据**。注意“按结构”三个字——不是拍张截图让大模型“看图说话”，而是像拆房子一样，把梁、柱、管线一根根登记造册。

先看两类输入的两条路。**PDF 走“真实物理页”**：系统记录的不是目录里的逻辑页码，而是内

容在文件里的物理页面与坐标位置——封面目录造成的页码错位，在这里不起作用。**DOCX 走“原生 XML”**：Word 文件内部本来就是一套结构化标记（XML，一种用标签组织数据的文本格式），系统直接读原生结构，不走“打印成 PDF 再识别”的弯路，表格、标题、脚注自带的信息一点不丢。

读出来之后，四样东西要“保得住”：

- **表格跨行合并保得住**：工程表格最怕跨页——一张表从第 37 页排到第 38 页，列头还得沿用上面的。系统把跨页的行重新拼回一张完整的表，合并单元格的关系也保留下来。
- **公式保存 TeX 源码**：数学公式不存截图，而是抽取成 LaTeX 源码（第 0 章里本讲义自己排版用的那套记号），查询时用 MathJax 在本地渲染——公式可复制、可检索、可再编辑。
- **图片与标签对应**：图纸/照片与它的图号、图题（“图 3-2 系泊布置图”）绑定登记，不会张冠李戴。
- **断句保留双锚点**：长句被切开成片段时，每个片段都带着指回原文位置的锚点，任何一段随时能拼回原文。

知识补齐：SHA-256——给每份内容发“内容身份证”

SHA-256（Secure Hash Algorithm 256，安全散列算法 256 位）是一种**指纹函数**：无论输入是一句话还是一部电影，它都算出一个 256 位的“指纹”，内容变一个字节，指纹就面目全非；反过来从指纹推不出原文。本项目用它给每份资料、每张表、每个片段登记**内容身份**。它解决一个细腻的问题：**同名不同内容**。两份文件都叫《主机操作手册.pdf》，一份是 Rev.2 一份是 Rev.3，按文件名登记必然撞车；按 SHA-256 登记，两份各自有各自的身份证，互不冒充。你在第 3.8 节会看到，这枚指纹是“三层身份”的第一层。

“锚点”（anchor）值得单独说，因为它是全系统可追溯性的地基。**锚点就是“出处坐标”**：一串形如“哪份资料、第几个物理页、哪一行、哪一格”的登记信息。19,893 个来源锚点，意味着结构化产物里的每个片段都带着自己的“户口”——后来智能体答的每个数字能“点回原件”，靠的就是它。下图的总体方案里，左列“文档处理”、右列“表格数据”两条链，最终都汇到底部同一句话：“核查结果后返回答案和出处”。

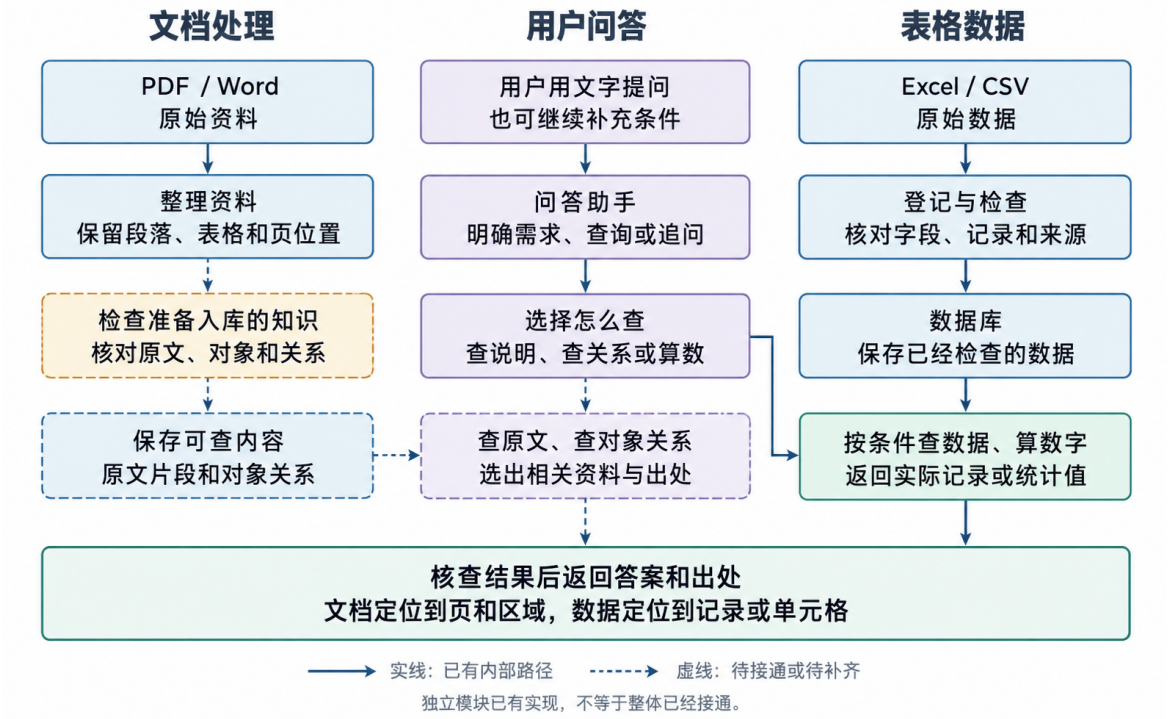


图 3.3　总体方案架构图：左列“文档处理”（PDF/Word 原始资料 → 保留段落、表格与页位置的整理 → 入库前的知识检查 → 保存可查内容），右列“表格数据”（Excel/CSV 原始数据 → 登记与检查 → 数据库 → 按条件查数据、算数字），中列“用户问答”把两条链汇到一起；底部是共同要求——核查后返回答案和出处，文档定位到页和区域、数据定位到记录或单元格。注意图例：实线是已有内部路径，虚线是待接通或待补齐——“独立模块已有实现，不等于整体已经接通”

图片来源：本项目《总体方案与验收计划》文档配图

结构化干了这么多活，成果与缺口都必须**拿数字说话**。下表是某一阶段的真实产物清单——注意最后三行，它们和成果一样重要：

表 3.1　数据结构化阶段产物与如实登记的缺口（1,001 个源物理页）

项目	数量	说明
恢复表格	484张	跨页跨行合并，保留合并单元格关系
数学公式源码片段	272 段	TeX 源码入库，本地 MathJax 渲染
来源锚点	19,893 个	每个片段可指回“哪页哪行哪格”
局部回退区域	39 处	结构过于复杂，自动恢复降级为保守模式，**如实登记待改进**
列宽证据不足的表	138 张	原始 PDF 无法证实原表列宽，按保守策略恢复
人工确认计数	0	全部产物为机器自动生成，**未冒充人工复核成果**

最后三行就是本项目反复出现的“**如实条款**”：39 个回退区域、138 张存疑的表，不藏、不抹、不改口；更难得的是“人工确认计数 0”——系统大大方方承认这些产物**没有**经过人工逐项确认。听起来像自曝家丑？恰恰相反：把“哪些是机器猜的、哪些是人核过的”分得清清楚楚，下游才敢放心用、也才知道用之前该查什么。这份诚实，第 3.7 节还会回来收利息。

3.3　一句话问数：智能体与可追溯查询

数据进了库，怎么查？过去你得会写 SQL（下一节的青盒里 30 秒教会你）——但平台工程师不该被迫学数据库语言。系统给出的答案是**智能体**（agent，能理解意图、调用工具、组织回答的程序）：你用人话问，“2023 年 11 月 18 日 20:27 的尾吃水是多少”，它替你查库、替你算数、替你组织答案。听上去像手机语音助手？往下滑你会发现，这套系统处处是“镣铐”——而**镣铐正是它能被信任的原因**。

先认识两位主角。**大脑**是 Gemma 12B——一个有 120 亿参数的本地大语言模型，用 llama.cpp 在本机 GPU 上量化部署（第 8 讲的部署套路），上下文窗口 8,192 个词元，温度设为 0（不许即兴发挥），并发固定为 1（一次只办一件事）。它负责**理解问题、挑工具、提交 SQL、组织回答**。**检索**由 Granite 语义模型负责：它把文字变成 768 维向量（第 4 讲讲过“向量是语义的坐标”），问题与表名、字段说明、历史案例就在这个向量空间里比相似度。一个管想，一个管找，谁也不越界。

架构的精髓在下图：用户问题交给本地大模型，模型只能**提出**一条 SQL；这条 SQL 先过 **SQLGlot** 检查（是不是只读？字段存在吗？有没有超过资源限制？），再由 **Wren** 做数据模型规划，最后才交给 **DuckDB**——一个单机就能跑的分析型数据库——去执行真正的筛选与聚合。校验不过，错误信息退回模型修改重试；条件不明，系统直接反问你，而不是替你瞎猜一个条件。

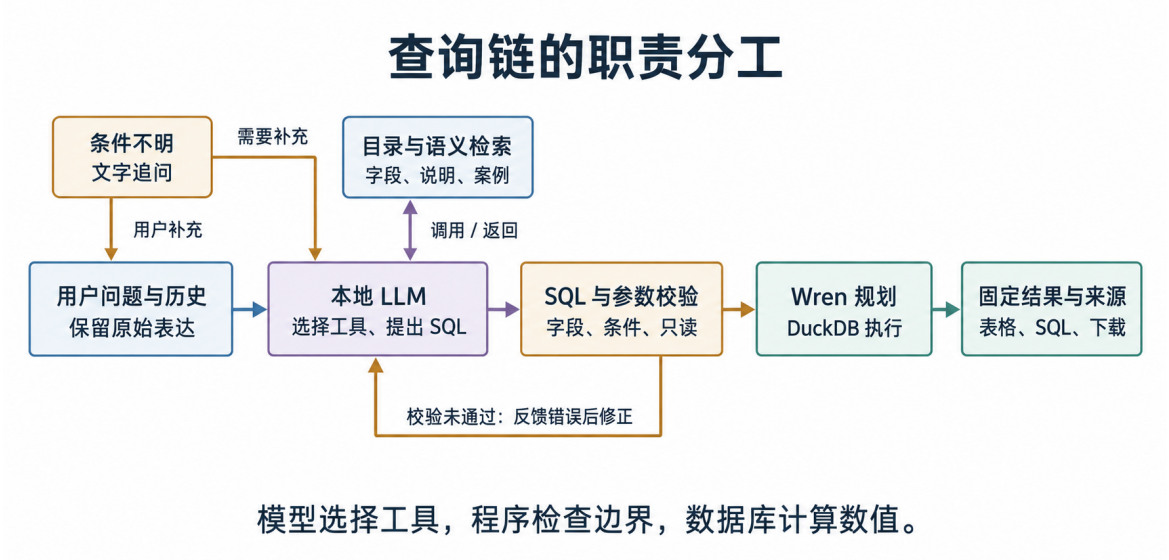


图 3.4　查询链的职责分工：本地 LLM 选择工具、提出 SQL，须经“SQL 与参数校验”（字段、条件、只读）通过后，才由 Wren 规划、DuckDB 执行，结果连同表格与 SQL 一起固定下来；条件不明走“文字追问”请用户补充，校验未通过则把错误反馈给模型修正——底部一句口号概括全部哲学：模型选择工具，程序检查边界，数据库计算数值

图片来源：本项目《总体方案与验收计划》文档配图

请把底部那句话读三遍：**模型选择工具，程序检查边界，数据库计算数值**。大模型在这条链里“只提出操作，不直接接触数据”——它碰不到原始 CSV，更碰不到原件；真正碰数据的只有被检查过的 SQL 和数据库引擎。这个设计叫**工具白名单**：智能体手里只有 8 个固定的工具，**没有**任意文件读写，**没有**命令行——模型再“聪明”，也翻不出这八件套的手掌心。这种“思考–行动–观察”的循环正是大模型 Agent 的标准范式（下图），区别在于：多数教学 demo 给 Agent 全权，本项目给 Agent 全副镣铐。

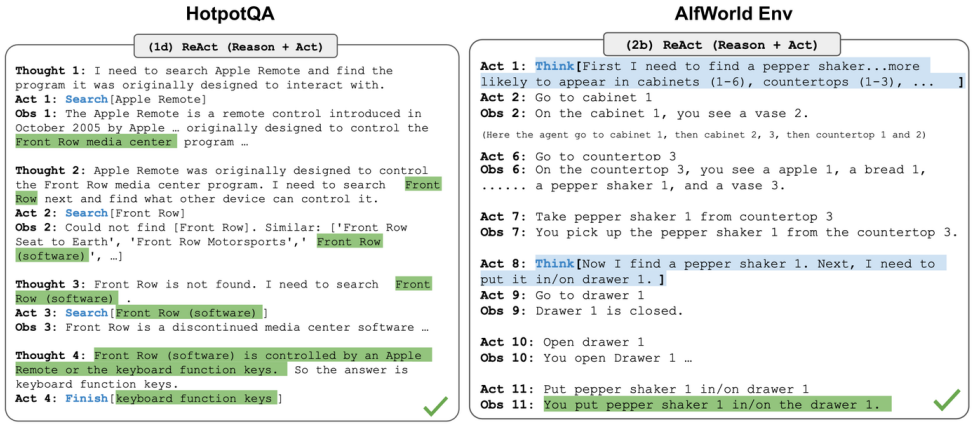


图 3.5　LLM Agent 循环示例：模型“思考”决定下一步行动，调用工具获得“观察”结果，再基于观察继续思考——循环直到给出最终回答；本项目的智能体就是这个循环，但每一步“行动”都被限制在 8 个白名单工具之内

图片来源：公开技术资料中的 ReAct 模式 Agent 循环示意（Yao et al., 2022）

零基础扫盲：SQL 是什么——数据库界的“填空题”

SQL（Structured Query Language，结构化查询语言）是跟数据库对话的官方语言。它的可读性好到惊人，比如“查 11 月 18 日 20 点 27 分的吃水”翻译成 SQL 大概是：`SELECT 尾吃水 FROM 运行数据 WHERE 时间 = '2023-11-18 20:27:00'`——SELECT 是“挑哪几列”，FROM 是“从哪张表”，WHERE 是“满足什么条件”。数据库收到后逐行筛出答案。它最大的好处是**没有歧义**：语句写定了，结果就定死了——这正是本章让大模型“只写 SQL、不算数字”的原因：人话有歧义，SQL 没有。

智能体对外承诺的能力，系统文档里编号 A1–A7。逐条翻成人话：

表 3.2　A 类查询与追溯的七项能力承诺

编号	名目	人话版
A1	一句话问数	直接说人话就能查，不用学 SQL
A2	时间精度到秒	“20:27”就是 20:27，不会给你“20 点档平均值”这种和稀泥的答案
A3	缺条件会追问	条件不足时反问你，而不是替你瞎猜一个答案
A4	数量与边界由程序保证	说“前五条”就只给 5 条；默认最多返回 100 行——截断是程序干的，不是模型嘴上说的
A5	数字由数据库算	所有求和、平均、计数都是 SQL 聚合的结果，模型只负责转述
A6	每个数字能点回原件	答案带锚点，可以一路点回 PDF 的哪页哪行哪格
A7	结果固定不串题	温度 0、并发 1：同一个问题今天问明天问，答案一致

A4 值得多说一句，它体现了一种**边界控制哲学**。“给我看前五条”这种话，若交给大模型自由发挥，它可能给你 6 条、4 条，甚至编 5 条——语言是模糊的，模型顺着模糊走。本项目的做法是：模型把“前五条”翻译成 SQL 里的 `LIMIT 5`，**由数据库引擎机械地执行截断**——说前五条就只给五条，一条不多，一条不少。把“模糊的语言承诺”变成“确定的程序行为”，是让用户敢信一个 AI 系统的第一块基石。

支撑问数的数据层也值得一窥。126 份 CSV 并不是简单倒进数据库了事：每份先**登记核验**（每份文件都要被完整地读两轮，通过核验才准入库），然后**按月分区**转成 Parquet 列式存储（按列组织

文件的格式，只查几列数据时不必读全表，快得多），最后在 DuckDB 里建成四个**查询模型**——把杂乱的原始表整理成四个“官方视图”：

表 3.3　四个查询模型：智能体能问什么，先由数据层划定

查询模型	列数	覆盖内容
`fpso_navigation`	29	航行相关：位置、航速、航向等
`fpso_load_status`	30	装载相关：各舱液位、吃水、卸载等
`fpso_machinery`	20	机械相关：主机、泵、发电机状态等
`fpso_motion`	27	运动姿态：纵摇、横摇、升沉等

能力不是吹出来的，是考出来的。项目组独立编写了 500 道题的回归考卷（“回归”指每改一版系统就重考一遍，防止改好一处坏三处），六种题型成绩如下：

表 3.4　智能体问数 500 题历史回归成绩（独立标准答案判卷）

题型	得分	说明
明细查询	219/220	查具体某时某刻某值；唯一失分题如实留档
普通聚合	100/100	求和、平均、计数、最值
连续对话	70/70	多轮追问，指代与上下文不丢
模糊歧义	60/60	“最近”“那次”等模糊说法，按 A3 追问或按约定消解
执行边界	28/30	LIMIT 截断、100 行上限等边界行为
复杂挑战	20/20	多表关联、嵌套条件

合计 497/500。速度上，60 道实测题的中位数耗时 4.93 秒——从“一个团队加班一晚”到“五秒钟一句话”，这就是开篇那道题的答案。还要强调一句验收纪律：这 500 道题的**标准答案是独立编写、先行复审**的，系统自己生成的 SQL 不许反过来当标准答案——否则等于让学生自己批自己的卷子。这句话在验收文化里还有更响亮的版本，第 3.8 节见。

3.4　知识门控：AI 抽的知识入库前先过检查

结构化和问数解决的是“资料里的东西怎么查”。但系统还有更进一步的野心：把资料里的**知识**（“P-101 泵的额定排量是 120 立方米每小时”这类事实条目）抽取出来、存进知识库，供检索与推理复用。这就引入了新的风险：**抽取靠大模型，而大模型会幻觉**——它会无中生有、会张冠李戴、会把“某工况下”悄悄放大成“任何工况下”。于是系统设了第三道关卡：**知识门控**（gating，入库前的闸门）。一句话概括其哲学：**机器抽的知识只是“候选”，过得了检查才配叫“知识”**。

门控流水线五步：**读取 → Chunk → 提取候选知识 → 执行门控 → 发布**。Chunk（分块）是把长文档切成带锚点的片段；提取步骤让模型从片段里抽出“主体–关系–客体”式的候选条目；然后是主角登场——每条候选都要过**三类检查**：

1. **是否支持**：候选条目说的内容，原文证据真的支持吗？还是模型脑补的？
2. **是否同一对象**：条目里的设备、系统，和证据里的是同一个吗？——三个平台都可能有一台叫“P-101”的泵。

3. **关系是否成立**：抽出的关系（“额定排量是”）与证据里的关系（“设计工况下排量可达”）是同一回事吗？

检查之后三个出口：**通过**（发布入库）、**转复核**（证据不够干净，交人工或专家裁定）、**拒绝**（打回并留下记录）。看下图的三岔口：

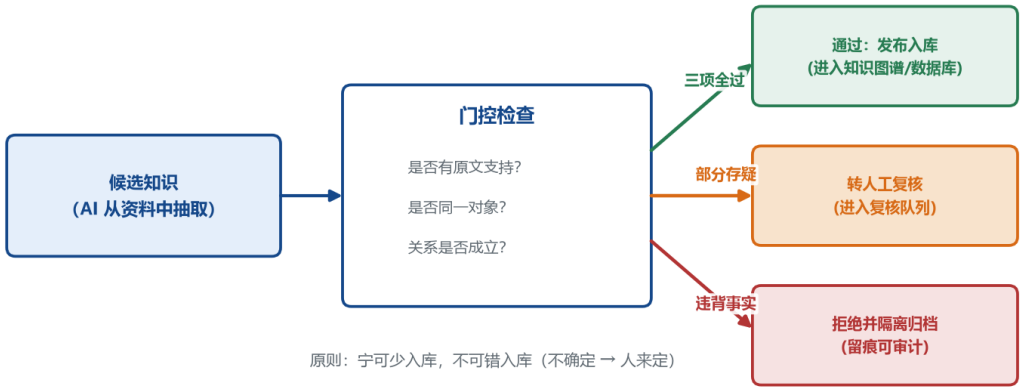


图 3.6　知识门控三出口：候选知识过三道检查（原文是否支持、是否同一对象、关系是否成立），按结论分流——通过（发布入库，带完整证据链）、转复核（证据存疑，交人工裁定）、拒绝（打回并留痕）；三个出口都有登记，没有一条候选能“悄悄消失”

光看流程图不刺激，看一个真实的教学例子。系统文档里有一张“P-101 表”，把泵 P-101 的五种典型情形和门控的五种处置一一对应——这张表是给新工程师的第一课，我们原样搬给你：

表 3.5　P-101 教学表：五种情形，五种处置（门控逻辑的人话版）

情形	候选知识 vs 原文证据	门控处置
内容一致	两条来源都写“P-101 额定排量 120 m^3/h”	**通过**：发布入库
数值冲突	手册写 120，计算书写 150	**拒绝**：证据打架，不许带病入库
身份不清	证据讲的是 3 号平台的 P-101，条目没说是哪个平台	**转复核**：交人确认对象
缺少证据	条目言之凿凿，却找不到原文出处	**拒绝**：孤证不立
扩大含义	原文说“设计工况下可达 120”，条目抽成“任何工况下 120”	**拒绝**：偷换条件，性质是歪曲

这套逻辑跑在真实数据上是什么样？某一阶段的运行数字：候选知识共 **9,505** 条，门控判下来——**接纳 2,988** 条、**待复核 582** 条、**拒绝 5,935** 条。看清楚：**六成以上的候选被拒之门外**。这不是模型太笨，恰恰是门控在工作——抽取模型天生“手松”，门控负责“手紧”。2,988 条通过者也不是直接上生产：先在**沙箱**（sandbox，与正式环境隔离的试运行区）分 20 批发布，每批观察无异常才转正。事后审计环节，**9,533 条事件链被完整重算一遍，结论一致**——每一条“为什么通过、为什么拒绝”都有账可查。对比一下老办法“AI 抽完人工把关”：人把关的瓶颈是人只有一双眼睛；门控把关的瓶颈是规则是否严谨——而规则，是可以审计、可以改进的。

3.5　图谱查关系：知识图谱与图检索

门控入库的知识是一条条孤立的事实。但工程师的真问题往往是**关系题**：“3号平台 P-101 的备泵是哪台？它的润滑系统和哪台主机共用？”——答案散在几十份资料里，靠关键词检索永远拼不齐。这就轮到**知识图谱**上场了。

知识补齐：实体、关系、三元组——知识图谱的三块积木

实体（entity）是世界里的“名词”：设备、系统、平台、文档；**关系**（relation）是名词之间的“动词”：“3 号平台 –拥有– P-101”、“P-101 –备泵是– P-102”。一条“主体–关系–客体”叫一个**三元组**（triple），把成千上万条三元组连起来，就是一张**知识图谱**（knowledge graph）：节点是实体，边是关系。它和向量检索的分工是：向量擅长“找相似”（这段话和那段话意思近），图谱擅长“查关系”（A 和 B 之间隔了几跳、走的什么关系）——你问“意义相近的段落”，用向量；你问“三亲等以内的亲戚”，用图谱。

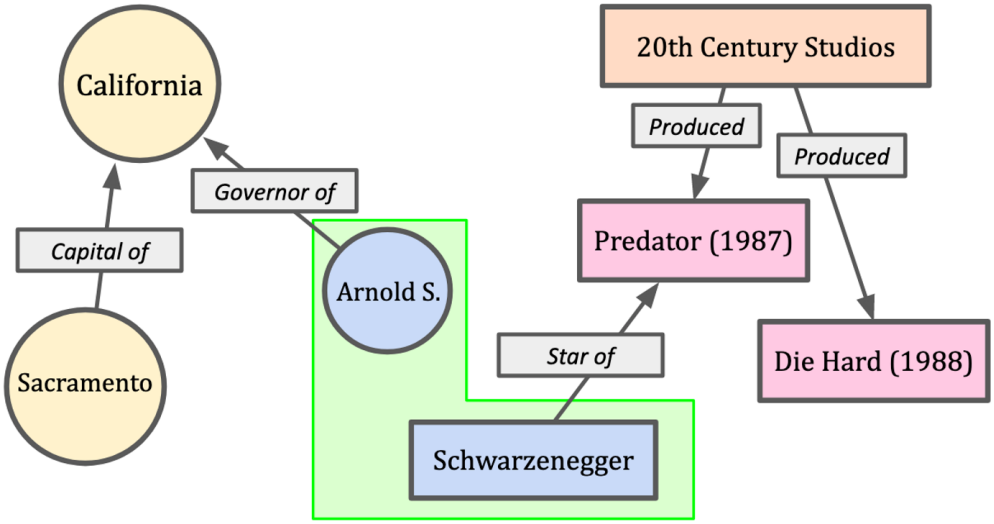


图 3.7　知识图谱可视化示例：节点是实体（设备、系统、文档），连线是关系；顺着边走就能回答“A 和 B 之间有什么联系”这类关键词检索答不了的题

图片来源：Wikimedia Commons，知识图谱网络可视化示例（CC 授权转载）

检索的具体打法叫 **GraphRAG**（Graph-based Retrieval-Augmented Generation，基于图的检索增强生成）：回答问题前，先沿着图谱把**相关的实体与关系当作证据**找出来，再把证据交给生成模型组织答案——比普通 RAG（只检索文本片段）多了“关系”这味药。而给证据**排序**的模型叫 **R-GCN**（Relational Graph Convolutional Network，关系图卷积网络），它的直觉看下图：一个节点该“信谁”，取决于它沿**不同类型的关系**连着谁——设备类关系连来的证据，和文档类关系连来的证据，权重应当不同。R-GCN 给每类关系各配一套聚合参数，本系统叠了三层，并把 BGE 模型输出的 1,024 维实体向量投影到 256 维排序空间——人话：把“语义相似”的向量，微调成“关系上也站得住”的排序。

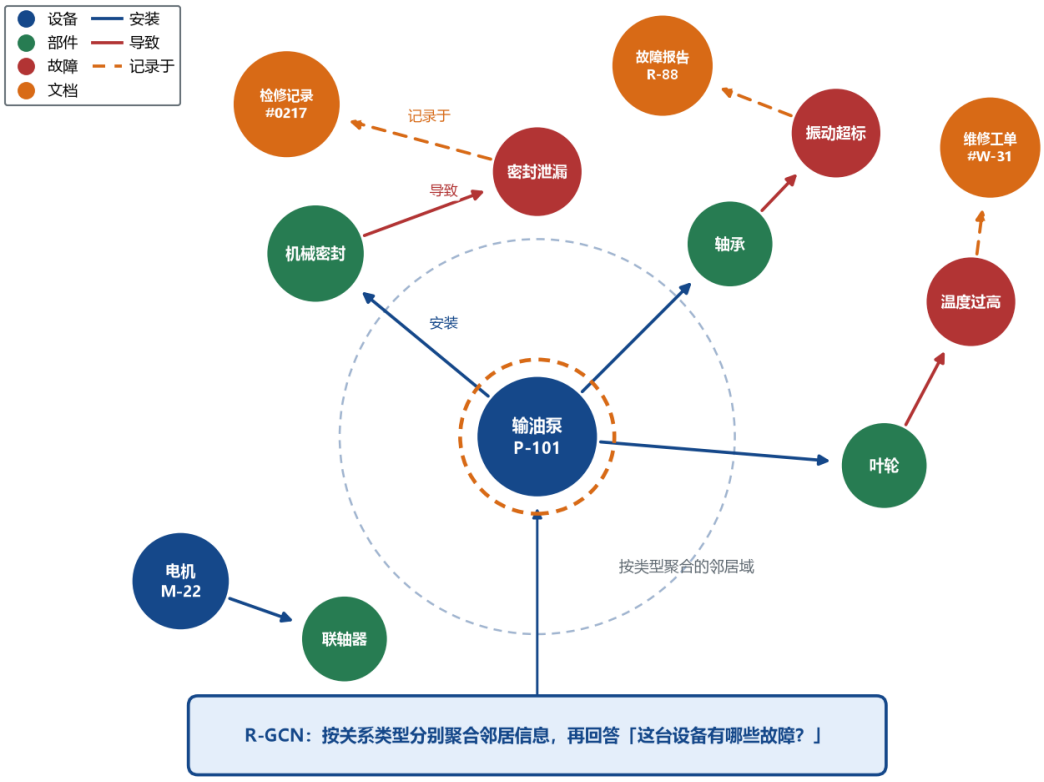


图 3.8　R-GCN 图检索示意：以查询抽中的实体为种子，沿知识图谱中不同类型的关系（不同颜色的边）逐层聚合邻居信息；三层之后每个节点都“见过”三跳以内的邻居，再投影到排序空间为候选证据打分——不同关系类型各有各的聚合权重，“谁说的话更可信”由结构说了算

在线检索的流程是一套严格的“节流阀”：一次提问最多取 **8 个种子实体** → 沿真实关系向外扩 **3 跳** → 裁剪到 **128 个候选** → R-GCN 聚合排序，选出最相关的证据交给生成端。为什么处处设上限？和 A4 的边界哲学同源：**检索范围必须是确定的、可审计的**——不设上限的图检索，一遇密集节点就会把无关证据卷进来，既慢又会“污染”答案。8/3/128 三个数字写在系统文档里，每次检索都照此执行，复盘时能精确说清“当时看了哪些证据”。

3.6　假内容攻防战（本章高潮）

门控很能干，但它有盲区。回看 P-101 表：门控的五连处置，管的全是**证据层面**的问题——没出处、证据打架、对象不清。可如果一条假知识**带着貌似齐全的证据、写着貌似合理的数值、语言毫无破绽**呢？随着 AI 生成内容渗入工程文档，这不再是抬杠：一份被 AI“润色”过的报告完全可能把 120 写成 135、把“试验值”写成“额定值”，语言通顺、格式规范、还带着参考文献式的装饰。**语义词造假**——无数值格式错误、无逻辑硬伤、纯粹在事实层面撒谎——是门控规则抓不住的。系统的回应，是一个独立的对抗训练 demo（接入 DeepSeek 云端接口，模型 `deepseek-flash`，即 DeepSeek-V4.1-Flash，使用时须关闭思考模式）：**让大模型扮演造假者，日夜不停地生产假条目，把防御系统练出“火眼金睛”**。

先立铁律，再谈攻防。本项目两条铁律，值得抄在扉页上：

- **“大模型只能改说法，永远改不了事实——目标假数值由程序给定。”**造假者拿到的假数值是程序写死的，大模型的任务只是把假数值“包装”得像真的；它无权自己发明数字。没有这条，造假者自己幻觉出来的“假货”会污染整个训练场。

- **“LLM 只做增强，离线逻辑永远是兜底。”**云端模型断网、涨价、下线，离线流水线照样出题、照样判卷——第 3.7 节的四跑对照表会给这条铁律一个漂亮的注脚。

整个攻防闭环由四个角色组成，看下图与角色表：

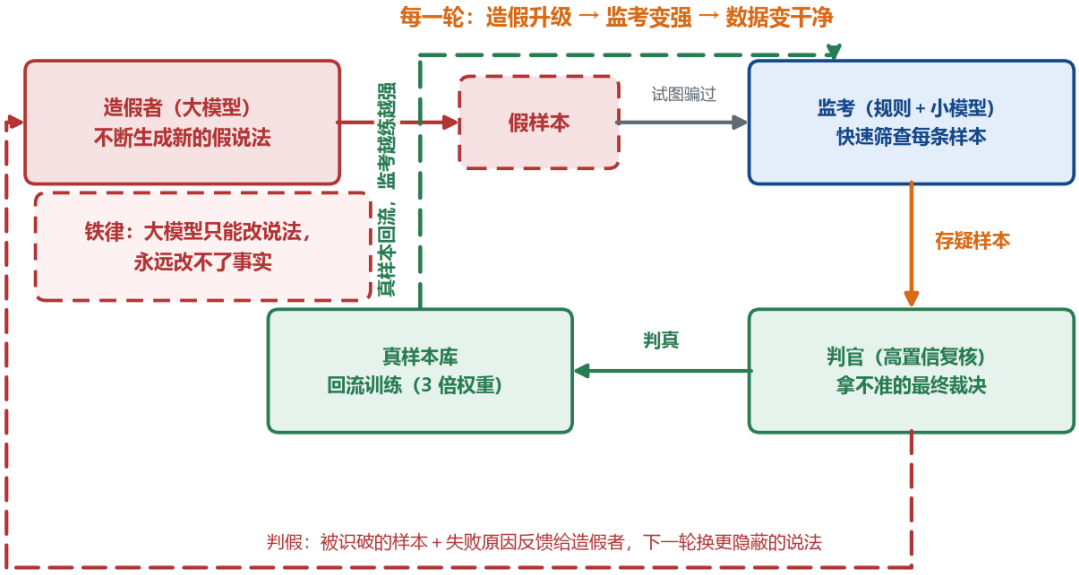


图 3.9　真假内容攻防闭环：造假者（大模型 + 程序给定假数值）批量生产以假乱真的条目；监考对不确定样本追问、对放行样本抽检；判官只认数值与单位等硬证据定生死；库存管家看管家底、扫描“中毒”样本；复核结论沉淀为蒸馏样本与先例库回流——四方在同一张考卷上螺旋进化

表 3.6　攻防四角色：人设与关键数字

角色	人设	关键规则与数字
造假者	不写出题范围的老千选手	每轮 30 条直接造假 + 10 条语言造假 + 10 条纯文字造假，最多 5 条润色；骗过的题目以 few-shot 注入下一轮“进化”，每 4 轮强制策略变异
监考	不确定必问的场务	不确定必追问，每轮不超过 25 条；纯文字样本走三分类复核；放行样本抽检 10%
判官	只认硬证据的主考	存疑样本复核每轮不超过 25 条；判“真”须置信度 ≥ 0.75 且有数值/单位等硬证据，光“读着像真的”不算数
库存管家	看家底的仓库主任	管蒸馏库（2,654/5,000）、监考先例库（6,891/8,000）、中毒隔离（72 条）；支持缓存报告、中毒扫描、一键回滚快照

进化机制是这场军备竞赛的灵魂。造假者每轮产出的题目，凡骗过防御的，会被作为 few-shot 示例“喂”回它自己下一轮的提示词——“照这个水平继续造假”；每 4 轮还强制策略变异，防止它在一种套路上一路走到黑。防御方则把每次复核的结论沉淀成两种资产：**蒸馏样本**（以 3 倍权重回流训练）和**监考先例库**（“上次遇到这种话术是这么判的”）。于是你攻我守、你守我攻，双方水平螺旋上升——这正是第 12 讲 GAN“生成器与判别器互相逼强”思想的工程化落地，只是博弈的赌注从“以假乱真的图像”换成了“以假乱真的工程数据”。

考卷本身也一丝不苟：3,300 道题的封存考卷中含 300 道 **OOD**（out-of-distribution，分布外——出题时故意换上训练时没见过的造假手法，专测“遇上新骗术会不会抓瞎”）。造假手法登记了 **26 个**

造假家族（G1–G19 编号，其中 PURE_TEXT 纯文字家族下分 7 个亚型），并按“易、中、难”三层分账——易档改个数字，难档通篇语义撒谎、只字不差。下面两张图是这场攻防战的驾驶舱：

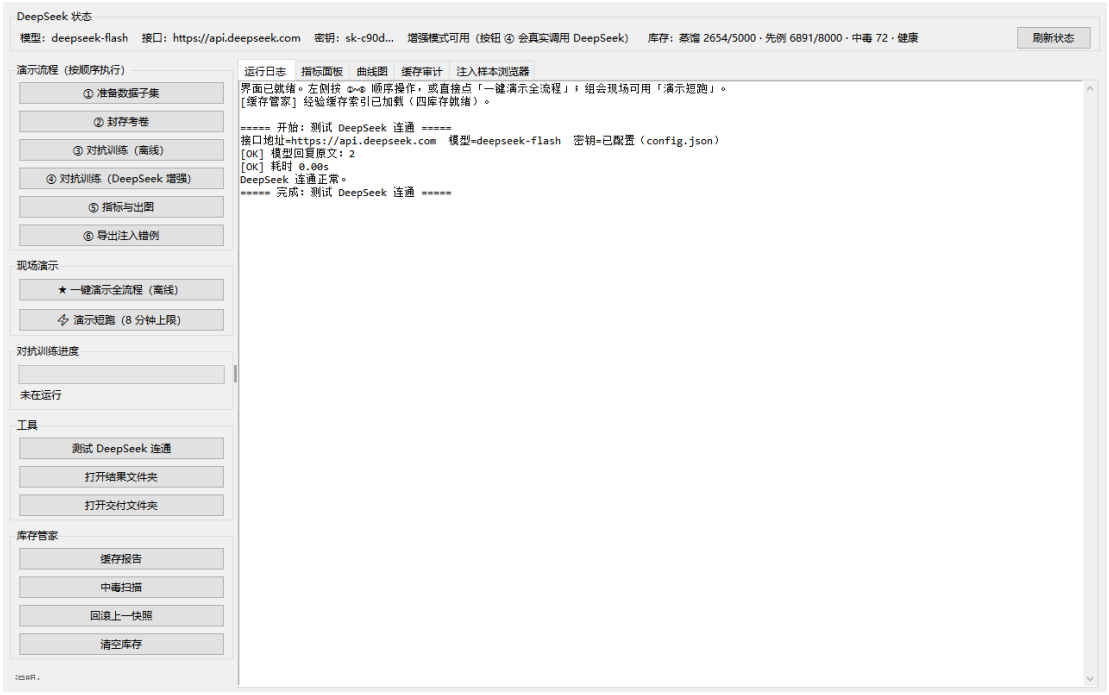


图 3.10　对抗训练 demo 主界面：左侧自上而下是六步演示流程（①准备数据子集 → ②封存考卷 → ③对抗训练（离线）→ ④对抗训练（DeepSeek 增强）→ ⑤指标与出图 → ⑥导出注入错例）、现场演示按钮（一键演示全流程/8 分钟演示短跑）、库存管家维护按钮（缓存报告/中毒扫描/回滚上一快照/清空库存）；顶部状态栏常驻库存水位（蒸馏 2,654/5,000、监考先例 6,891/8,000、中毒隔离 72）；右侧是多页签日志区，图示为一次 DeepSeek 连通性测试（模型回复 OK，用时 0.06 秒）

图片来源：本项目对抗训练 demo 截图（E:\中海油数据，demo_ 对抗训练 results）



图 3.11　对抗训练 demo 的指标面板（离线模式最近一次结果）：宏平均 F1 0.742（95% 置信区间 0.728–0.756）、综合区分度 0.937、监考抓包率 0.947、金标字段全量 13,475 条零缺失、交付 2,000 条分层齐全（易 300/中 400/难 300），下方是 16 项自检逐条“通过”的清单——每个成绩旁边都挂着可核查的证据，而不是一句“效果良好”

图片来源：本项目对抗训练 demo 截图（E:\中海油数据，demo_ 对抗训练 results）

攻防的最终交付物是一份 **2,000 条真假混装数据集**：假条目 1,000 条（易 300、中 400、难 300）加真条目 1,000 条（五个真话亚族），每条带双金标——`gold_label`（客观真假）与 `disc_label`（判官当时的判定口径）。往后任何人想检验“我的模型能不能识别工程数据造假”，拿这份数据集一测便知，不必从零再造一场攻防。

动手时间：当一回监考——三道题，请人工判定

下面 3 条候选知识，先别看答案，和同桌各自判定“通过/拒绝/转复核”，再对照第 3.4 节的 P-101 表逻辑对账：(1) “P-101 泵额定排量 120 m^3/h（摘自《设备操作手册》物理页 45，表 3-2）”；(2) “P-101 泵额定排量 135 m^3/h”；(3) “P-101 泵在冬季工况下排量可提升约 20%”。**参考判定**：(1) 出处具体、数值有据——通过；(2) 无出处，且数值与手册冲突——拒绝（数值冲突）；(3) 没说是哪个平台的 P-101（身份不清），“约 20%”也无从核实（证据不足）——转复核。判对了几条？如果你在 (3) 上犹豫了，你就亲身理解了为什么门控需要“转复核”这个出口、监考为什么“不确定必问”——**犹豫本身不是 bug，把犹豫假装成确定才是。**

3.7　0.88 的谎言与 0.74 的诚实

这一节讲本项目的“名场面”。先看下面这条曲线，它是 2026 年 9 月中旬以前系统的“官方战绩”：75 轮对抗，判官在封存考卷上的 macro-F1 始终稳稳停在 0.87–0.88，一条漂亮得像用尺子画出来的水平线。

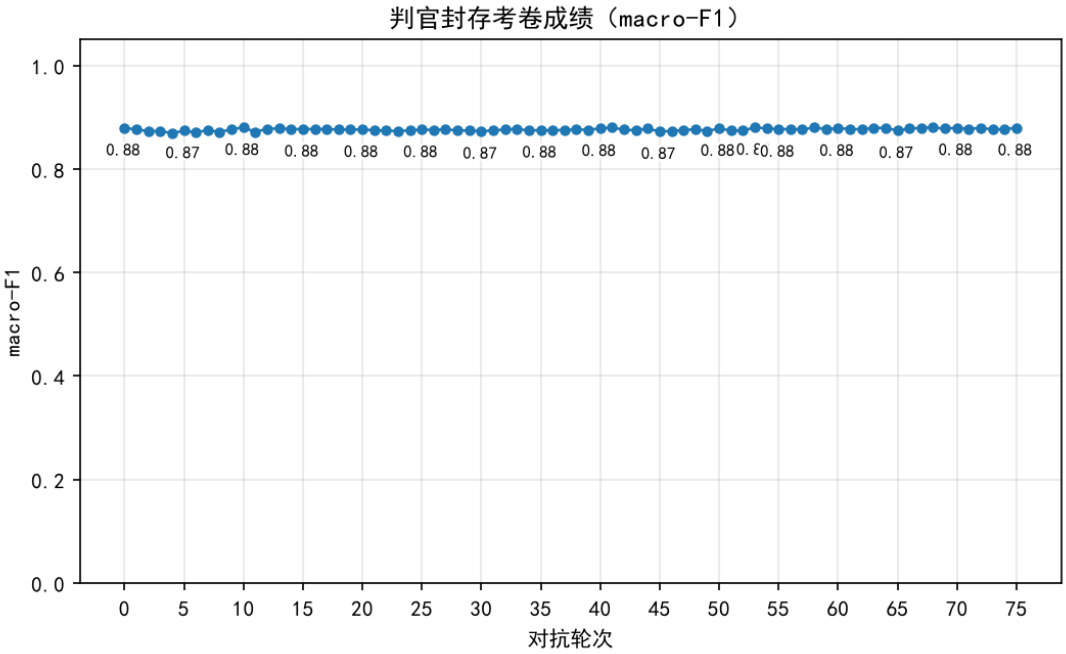


图 3.12　旧口径下的“判官封存考卷成绩（macro-F1）”曲线：75 轮对抗，成绩纹丝不动地钉在 0.87–0.88——这条漂亮得过分的水平线，恰恰成了 2026 年 9 月 15 日那次审计的起点：真实系统里，几乎没有哪条指标能这样一动不动

图片来源：本项目对抗训练 demo 产物（E:\中海油数据，demo_ 对抗训练 results）

2026 年 9 月 15 日，项目组对考卷本身动了审计——查的不是“系统考得多好”，而是“考卷配不配考”。结果三大水分，条条触目：

1. **考卷与训练数据事实级重叠 78%**：七成八的考题，其考查的事实在训练数据里出现过“同一事实的变体”——相当于期末考试考的全是课后原题的换皮。
2. **逐字重复率 15.7%**：一部分考题与训练样本存在整句级别的雷同——连换皮都懒得换了。
3. **判官 17 维特征中 11 维可查表**：判官赖以判卷的 17 个特征里，有 11 个能直接查表得到——等于开卷考试，还允许带标准答案进场。

结论只有一个：**旧口径下的 0.84–0.88 全部作废**。项目组重建考卷（斩断事实级重叠）、重建特征（砍掉可查表维度），把系统拉回真实水位重新考试——新的成绩落在 0.73 档。项目组在报告里写下一句后来被反复引用的话：“**0.72–0.74 是去掉水分后的真实水位，不是性能回退**。”数字变难看了，但数字第一次变得可信。

先补一句人话：本节反复出现的 macro-F1（宏平均 F1）是三分类（真话/造假/核不了）的综合成绩单：

$$\text{macro-F1} = \frac{1}{K}\sum_{k=1}^{K}\frac{2\,P_k R_k}{P_k + R_k}, \tag{3.1}$$

人话：K 是类别数（这里 $K=3$），P_k、R_k 是第 k 类各自的查准率（判为第 k 类的里面判对了多少）与查全率（真是第 k 类的里面抓住了多少）。“宏平均”就是先给每一类各算一份 $F1$ 再取平均——少数类（比如“核不了”）不会被多数类淹没，谁差拉谁的平均，偏科藏不住。

重建之后，项目组做了四组对照实验，把每个组件的贡献摊在桌面上：

表 3.7　四跑对照：每个组件的贡献一次看清（新口径考卷）

轮次	配置	macro-F1	备注
Run A	离线逻辑全开，64轮	0.7417	规则基线为 0.7261，离线闭环有效
Run B	DeepSeek 增强对抗	0.7401	与 Run A 不可区分：增强是锦上添花，不是救命稻草
Run C	蒸馏后离线复测	0.7403	零 API 调用：云端能力已“蒸”进本地
Run D	全部关闭（消融）	0.7190	跌破规则基线；ECE 0.210 全场最差

读这张表有三个收获。其一，Run A 比 0.7261 的规则基线高约 1.5 个百分点——离线闭环（蒸馏回流 + 先例库）真实有效。其二，Run B（0.7401）与 Run C（0.7403）几乎等于 Run A（0.7417）：云端大模型的增益**与离线不可区分**——铁律“LLM 只做增强，离线逻辑永远是兜底”不是口号，是可复现的实验结论。其三，Run D 把组件全关后成绩**跌破基线**，且 ECE 达到全场最差的 0.210（ECE，Expected Calibration Error，期望校准误差：模型“自信程度”与“实际对错”的偏差，越小越好——一个 ECE 很差的系统会一边出错一边理直气壮）。

还有一份更痛的诚实，写在 OOD 赛道上：300 道“纯文字/拼接造假”的分布外考题，F1 只有**0.071–0.233**。项目组没有把这几行数字藏进附录，而是原样写进验收材料，并附上一句诊断：“**系统对无数值痕迹的语义造假确实是盲的**。”品一品这句话的分量：当前防御体系依赖数值与单位等硬证据，一旦造假者只玩文字游戏，系统近乎失明——这正是组 3 的同学（第 3.8 节）接下来要啃的硬骨头。

工程伦理：一份漂亮成绩单的死亡与重生——三条教训

这次 0.88 → 0.74 的改判，值得每个工程人当伦理案例读。**教训一：掺水的地图比没有地图更危险。**指标是给决策者看的地图——拿着 0.88 的地图，甲方会放心把系统接进生产流程；水分挤破那天，信任一起破产。宁可交一份诚实的 0.74，不交一份虚胖的 0.88。**教训二：过于漂亮的曲线本身就是事故征兆。**75 轮纹丝不动的 0.88，与其说是成绩，不如说是“考卷漏了”的报警器——看到完美得反常的指标，第一反应应该是怀疑测量，而不是庆祝。**教训三：审计要审“考卷”，不只审“考生”。**系统没作弊、模型没作弊，出问题的是数据集与特征的构造方式——这类“无主观恶意的水分”只有靠制度化审计（查重叠、查重复、查特征泄漏）才能揪出。本项目从此立下规矩：**凡报指标，必附考卷说明与水分检查记录**——“如实条款”就是这样一章章攒出来的。

3.8　三层身份与验收文化

如果你只从本章带走一个工程习惯，我们推荐这一个：**给数据发身份证**。系统里任何一条信息，都要能回答三个“你是谁”：

- **内容身份**：这份内容本身是什么？——SHA-256指纹登记，同名不同内容分别记录（第 3.2 节的知识盒子）。
- **来源身份**：它从哪来？——锚点精确到“哪份资料、哪一页、哪一行、哪一格”，19,893个锚点就是 19,893 张户口页。
- **结果身份**：谁算出了它？——每次查询有唯一查询编号，答案、SQL、数据快照绑定存档，事后任何争议都能原样重放。

三层身份层层衔接，串成一条完整的责任链：用户问的每个数字（结果身份）能查到它来自哪张表（来源身份），那张表的每个字节又能验明正身（内容身份）。这套身份体系支撑着一套近乎偏执的验收文化。系统文档里写了三条格言，我们原文抄录：

- **“不以智能化、准确、高效等形容词代替说明。”**
- **“标准答案须独立编写并先行复审，系统生成的 SQL 不得反过来充当标准答案。”**
- **“静态截图、预填答案或人工拼接两个项目的结果不替代连接。”**

第一条治“吹”——形容词不算证据，数字才算；第二条治“自证”——考生不能自己批卷；第三条治“摆拍”——demo 里两个模块各自转得再漂亮，没真正接起来的管线就是没接起来。2026 年 10 月 12 日的验收就是按这套文化执行的，分三步：**贯通验收**——四条链（结构化链、问数链、门控链、图检索链）逐链判定，不许打总分和稀泥；**现场演示**——35 分钟五段，真数据、真提问；**三层核验**——按内容/来源/结果三层身份逐项对账。

验收的产出物里，最珍贵的一份是**缺口表**：G1–G11 共 11 条“系统还没做到的事”，逐条如实列出、逐条给出位置与建议。摘其中与本课程最相关的三条：**G8**——正式知识图谱尚未发布（现在沙箱里的图谱还没转正）；**G9**——R-GCN 尚未与业务数据联合验收（模型在测试集上能跑，还没在真实业务问题上验明正身）；**G10**——难负样本与对抗训练尚未形成成果（第 3.7 节那个 OOD 盲区，正是缺难负样本练手的症状）。缺口表不丢人——它恰恰是一份系统“活着”的证明；丢人的是那种通篇“达到国际先进水平”的验收报告。

而这三个缺口，恰好就是你们三个学生组的进场口。第 5–19 周，每组 5 人、每周投入 6–8 小时：

表 3.8　三个学生组与缺口表的对接（详见第三篇第 13 章至第 15 章）

组别	名称	对接的缺口与主线任务
组 1	数据结构化组	对接结构化侧缺口：39个局部回退区域、138 张列宽证据不足的表（第 3.2 节）——提升 B 类结构恢复的覆盖与质量
组 2	智能问答与图检索组	对接 G8/G9：推动正式图谱发布，完成 R-GCN 与业务数据的联合验收
组 3	虚假内容识别与对抗训练组	对接 G10：把难负样本与对抗训练做成成果，正面强攻 OOD 语义造假盲区

项目连线：三个项目，同一条纪律

把三个项目并排放，你会看到同一条纪律的三种长相。**数据资产化**：六目拼接的全景图带拼接参数（第 一 章）、水下双目的图对带 meta.json（第 二 章）、数据基座的表格带锚点与 SHA-256——都坚持“没有身份的数据不是资产”。**先过滤再入库**：伪标签要过四重过滤、知识要过门控、假条目要过监考与判官——都拒绝“生米直接下锅”。**如实文化**：拼接项目登记配准失败情形、双目项目公布误差的分布尾巴、数据基座发布缺口表 G1–G11——都把“没做到的”写进交付物。这套纪律不属于某个行业，它属于所有想把 AI 用在正经地方的人。

3.9　回头看：你将用到第一篇的哪些本事

本章出现了大量“黑话惊魂”：向量检索、量化部署、Agent、对抗训练、macro-F1……别慌，它们全部能在第一篇找到老巢。下表是对照清单——学第一篇时带着右栏的问题去学，到第 5 周进场时你就是组里“看得懂全局”的那个人：

表 3.9　第一篇知识在本项目中的落点

第一篇	讲什么	在本项目哪里用上
第 3 讲	训练与评估	500 题回归考卷、macro-F1（式 (3.1)）、置信区间、ECE——“先定怎么考，再谈学得好”是全项目的方法论底座
第 4 讲	自编码器与“表示”	Granite 的 768 维语义向量、BGE 的 1,024 维实体向量、投影到 256 维排序空间——“用向量表示意义”一以贯之
第 6 讲	CNN 训练循环	蒸馏样本 3 倍权重回流、复盘再训练——“数据回流、模型更新”就是训练循环思想在数据侧的投影
第 8 讲	轻量化与部署	Gemma 12B 用 llama.cpp 本地量化部署，上下文 8,192、并发 1——“本地小模型 + 严格边界”是涉密场景的务实路线
第 11 讲	注意力与大模型 Agent	智能体“思考–行动–观察”循环、8 个白名单工具、8,192 上下文预算的精打细算
第 12 讲	生成对抗与蒸馏	造假者 vs 监考/判官是 GAN 博弈思想的“数据质检版”；蒸馏回流与判官先例库是博弈成果的沉淀方式

前沿速递：2024–2026，数据治理撞上大模型

本章的每一环都在 2024–2026 年经历着快速演化，四条线索值得关注。其一，**RAG → GraphRAG**：2024 年起，业界把知识图谱引入检索增强生成（GraphRAG），用社区摘要与多跳关系回答“全局性、关系性”问题——本章第 3.5 节正是这条路线的工程化迷你版。其二，**Text2SQL 的评测革命**：学术界 Text2SQL（自然语言转 SQL）基准从 Spider 演进到 Spider 2.0 与 BIRD，专门考“真实企业级数据库”上大模型的本事——结论很有趣：模型写 SQL 的“雅思分数”年年涨，但在真实库上“问数”远没到让人类放手的程度，本章 SQLGlot/Wren/DuckDB 的多关卡设计正是对这一现实的回应。其三，**LLM 标注的质检革命**：用大模型给数据打标已是行业常态，而 2024–2026 年的重心从“LLM 当裁判”转向质检——金标抽检、一致性审计、标注泄漏检查，与本章门控/监考/判官的三层设卡同构。其四，**合成数据的双刃剑**：Shumailov 等人在 Nature（2024）警告“模型在递归生成的数据上反复训练会崩溃”（model collapse）——合成数据省标注，但必须像本章这样“程序给定真值 + 双金标 + 水分审计”，否则是在用自己产生的回声训练自己。

章末把开篇那道题收个尾。“把 2023 年 11 月 18 日 20 点 27 分前后的所有信息给我”——现在你知道这句话背后的全部代价：1,001 页要拆出 484 张表、272 段公式、19,893 个锚点；3.1 亿行要登记、分区、建成四个视图；一个五秒钟的答案背后，是 8 个白名单工具、三道计算关卡、六成候选知识被拒之门外、26 个造假家族日夜攻防，以及一次把 0.88 改判成 0.74 的审计。三个组的位置已经留好：结构化在第 13 章，问答与图谱在第 14 章，攻防在第 15 章。去把缺口表上属于你们的那几行，划掉。

数据结构化（data structuring）

把 PDF/Word 等非结构化文档还原成“表格是表格、公式是公式、段落有出处”的结构化数据；本项目 1,001 页产出 484 张表、272 段公式源码、19,893 个锚点。

锚点（anchor）

“哪份资料、哪一页、哪一行、哪一格”式的出处坐标；全系统可追溯性的地基，每个数字“点回原件”靠它。

SHA-256

一种 256 位指纹函数：内容变一个字节、指纹面目全非；本项目给每份内容登记“内容身份证”，同名不同内容分别记录。

智能体（agent）

能理解意图、调用工具、组织回答的程序；本项目的大脑是本地部署的 Gemma 12B（上下文 8,192，温度 0，并发 1）。

工具白名单（tool whitelist）

只允许智能体调用事先登记的固定工具（本项目 8 个），无任意文件读写、无命令行——能力边界由程序而非“自觉”保证。

Text2SQL

自然语言转 SQL 的技术；本项目由大模型起草 SQL，但须经 SQLGlot 校验、Wren 规划后才由 DuckDB 执行。

DuckDB

单机分析型数据库，擅长对大表做筛选与聚合；本项目的“算数员”——所有数字由它算，模型只转述。

Parquet

列式存储文件格式，按列组织数据；本项目把 CSV 核验后按月分区转成 Parquet，查询时不必读全表。

知识图谱（knowledge graph）与三元组

用“主体–关系–客体”三元组把事实连成的网络：节点是实体（设备/系统/平台），边是关系；擅长回答“关系题”而非“相似题”。

R-GCN（Relational Graph Convolutional Network，关系图卷积网络）

按关系类型分别聚合邻居信息的图神经网络；本项目叠三层，把 1,024 维实体向量投影到 256 维排序空间给证据打分。

GraphRAG

基于图的检索增强生成：回答前先沿知识图谱取“实体与关系证据”，再交给生成模型组织答案。

知识门控（knowledge gating）

AI 抽取的候选知识入库前过三道检查（是否支持/是否同一对象/关系是否成立），按“通过/转复核/拒绝”三出口分流；本项目 9,505 条候选接纳 2,988、待复核 582、拒绝 5,935。

对抗训练（adversarial training）

让“造假者”与“防御方”持续博弈、互相逼强的训练范式；本项目四角色闭环——造假者、监考、判官、库存管家，登记 26 个造假家族。

OOD（out-of-distribution，分布外）

训练时没见过的数据分布；本项目考卷含 300 道 OOD 题，现成绩 F1 仅 0.071–0.233——“语义造假盲区”的如实公告。

第三部分

第三篇　任务书篇：15 个小组的任务、实验与验收

第三篇使用说明：任务书怎么读、怎么交、怎么评

第三篇地图

讲什么：15份任务书共同的游戏规则——任务书的固定结构怎么读、证据包三件套怎么留、AI与人怎么分工、成绩怎么算、第 19 周周五交什么，外加一张 15 组速查表。**为什么值得读**：本章是全篇的“总则”。先花二十分钟读通这里，再翻自己组的章，验收线上每一条你都见过同类，不会临场发懵。**学完你能**：说清“无证据 = 未完成”“AI 可以用、结果自己扛”两条铁律，把自己的提交清单与四个实验日期抄进日历。

第二篇讲的是三个项目“为什么做”（六目拼接见第 一 章，水下双目见第 二 章，数据基座见第 三 章），这一篇回答“你具体做什么”。全班 15 个小组（每组 5 人，每周合计投入 6–8 小时，组内自行排班），每组一章任务书。本章是 15 份任务书的总则：先读这里，再看自己组的章，两章合起来才是你这学期的完整说明书。

0.10 任务书怎么读：固定七节，先通则后本组

每份任务书的内部结构完全相同，固定七节：**组档案**（你是谁、跟谁交接）、**项目背景与你的位置**（为什么是你）、**核心任务**（要做什么）、**第 19 周周五提交清单**（最后交什么）、**AI 分工与人工必做**（谁干什么）、**防糊弄指南与满分标准**（怎么算分）、**四次实验安排**（什么时候做到哪）。建议的读法是“先两头后中间”：第一遍只读组档案和提交清单，五分钟知道自己组的边界；之后跟着四个实验节点的节奏，逐节精读其余部分。

全项目共用一套代码基座。六目拼接的八个组工作在 `stitcher_cpp` 目录（VS2022 工程，入口 `360stitcher.sln`），端到端路线在 `e2e_gpu` 目录；各组的训练代码与数据放在 `stitcher_cpp/training/groupXX`，训练好的模型统一交付到 `stitcher_cpp/build/Release/models/`。水下双目与数据基座项目的目录约定，见各自任务书的组档案一节。你的四个实验日期是全班统一的：实验一 2026 年 10 月 12 日、实验二 11 月 2 日、实验三 11 月 23 日、实验四 12 月 21 日；完全提交离场是 2027 年 1 月 22 日（第 19 周周五）。

0.11 证据怎么留：三件套、交付契约与求助通道

第一件事，**证据包三件套**。凡声称“我做出来了”，必须同时拿出三样东西：**命令全文**（你敲的是哪条命令，原样复制）、**完整日志**（终端从开始到结束的完整输出，不许截选）、**产物路径**（跑出来的文件在哪、叫什么）。三件缺任何一件，一律按“无证据 = 未完成”处理——截图可以摆拍，口说无凭，只有三件套能被别人原样复跑。

第二件事，六目拼接组还要记住**九头交付契约**，见下表。它设计的全部目的，是让“装样子”在技术上过不了关：探针会真去加载你的模型跑一遍，模型内容一变、输出必须跟着变，这就是“探针防绕过”。

表 10　六目拼接九头交付契约（逐条对照，缺一不过）

契约条款	具体内容
交付物	每个参数头一个`.onnx` 模型文件，配一个同名的`.metrics.json` 指标文件（文件名中的头名必须与头名完全一致，探针按文件名找文件）
交付位置	`stitcher_cpp/build/Release/models/`
模型探针	`python tools/nine_head_probe.py`，期望 `PASS (12/12)`
引擎探针	`python tools/engine_probe_test.py`，期望 `PASS (5/5)`
A/B 对比	`python tools/ab_compare.py before.json after.json`：同一批数据，新模型（after）与基准（before）同场比武
回退保护	模型损坏或输出异常时系统经 `[ProviderFallback]` 自动退回人工参数，不崩溃——坏模型只会没用，不会坏事

第三件事，**求助通道**：组内互助 → 项目助教（研究生）→ 导师，逐级上升。求助必须附“完整命令 + 完整输出”——这不是刁难，而是排查问题的唯一捷径：只发一句“老师我跑不通”，助教只能让你重发；带上完整输出，八成的坑一眼可见。

0.12　AI 怎么用：分工通则与两条红线

这门课不禁止 AI，反而**要求**你用 AI——但用得明白。通则一张表：左列的事可以放心交给 AI，右列的事永远自己来。

表 11　AI 分工通则：左列交给 AI，右列必须亲自做

AI擅长（可直接交给它）	人必做（不许外包）
解读项目代码、解释每段在干什么	在自己电脑上运行每一条命令
写解析脚本、画图代码、文档草稿	核对 AI 给出的每个结论与数字
解释报错信息、给出排查建议	亲眼看输出、看图、看训练曲线
复述论文与文档、翻译术语	在证据包上签字，对每个数字担责

两条红线之下，各有一份防糊弄指南，先在这里立此存照，各组任务书里还会结合本组任务重申。

防 AI 糊弄指南（通则）

(1) AI 给的数字必须本机复跑：它报的 loss、指标、耗时，只有你自己跑出来的那一份是真的。**(2) AI 引用的库版本与函数签名要核实**：它张口就来的 API 十有八九带着幻觉。**(3) AI 说“应该可以”“一般没问题”不等于跑通了**：跑没跑通，只认你的终端输出。**(4) 禁止把 AI 的回答直接粘进实验记录**：实验记录里只许出现你亲手跑出来的内容。

防人工糊弄指南（通则）

(1) 跑一次成功就截图当十次的证据，不行：每个节点各有验收线，一次表演补不齐四季庄稼。**(2) 改日志时间戳、拼接不同日期的输出 = 作弊**，按如实条款处理。**(3) 组内“代跑”必须留运行者姓名与时间**：谁跑的谁负责。**(4) 编造训练精度 = 该节点成绩作废**：这是如实条款，全

课程无弹性——工程世界里造假不是扣分，是出局。

0.13 成绩怎么算：建议方案（教师可调）

成绩构成的建议方案是：**实验一不计分（入场达标实验，必须通过验收）；实验二、三、四各 10 分，共 30 分；平时作业 20 分——学期两次、每次 10 分，从十二讲（第 1–12 讲）章后习题中任选 2 题作答；期末项目成果 50 分。总评满分 100 分。**实验分看你在二、三、四三个节点上是否达成当次验收线；作业分看两次习题作答的完成质量；成果分看第 19 周周五交出的整体质量。期末成果 50 分的典型构成见下表，各组任务书里的细则与此同构。

表 12 期末项目成果 50 分的典型构成

分值	项目	看什么
20	功能达成	任务书里“功能达成”一栏的动作是否全部落地（如模型交付、探针通过、标签达标）
10	证据完整	四个节点的证据包三件套是否齐全、能否被他人原样复跑
10	报告质量	组报告按模板撰写，方法讲得清、图表齐全、结论有数据支撑
10	协作与台账	排班执行、组内互助与求助记录规范、台账条目无缺失

再次声明：**以上为建议方案，教师可调。**评定的唯一依据是各组维护的台账与证据包——不是谁嗓门大，是谁的记录全。

0.14 第 19 周周五交什么：总清单模板

2027年 1 月 22 日（第 19 周周五）完全提交离场。每个组都要交齐下表四件东西；各组任务书的提交清单表，是本表加上组内特有交付物（如六目各组的三头 ONNX 模型）之后的版本，以任务书为准。

表 13 第 19 周周五提交总清单模板（15 组通用）

内容	格式	说明
项目报告	PDF，8–15 页	模板六段：摘要 / 背景 / 方法 / 实验 / 结论 / 分工声明
代码包	zip，含 README	README 写清环境与运行步骤，别人拿到能跑
证据包	目录或 zip	按四个实验节点分四包，命令 + 日志 + 产物三件套
AI 使用声明	随报告附一页	哪些事 AI 做了、人做了什么核验，逐条列清

其中“分工声明”一节请认真写：谁负责哪部分、谁运行了哪些关键实验——它既是协作分的依据，也是你个人的项目履历。AI 使用声明则与防糊弄指南配套：用 AI 不是错，说不清 AI 做了什么、你核验了什么才是错。

0.15 15 个小组速查表

最后一张地图。找到自己组的章号，翻过去之前，先把本章的总则过一遍。

表 14　15 个小组与任务书章号速查

组号	组名	任务书
第 1 组	六目拼接 • 匹配决策组	第 1 章
第 2 组	六目拼接 • 特征质量组	第 2 章
第 3 组	六目拼接 • 缝合融合组	第 3 章
第 4 组	六目拼接 • 数据与标签组	第 4 章
第 5 组	六目拼接 • 测试集与评测组	第 5 章
第 6 组	六目拼接 • 端到端数据组	第 6 章
第 7 组	六目拼接 • 端到端训练组	第 7 章
第 8 组	六目拼接 • 端到端优化组	第 8 章
第 9 组	水下双目 • 数据组	第 9 章
第 10 组	水下双目 • 部署组	第 10 章
第 11 组	水下双目 • 训练组	第 11 章
第 12 组	水下双目 • 评测组	第 12 章
第 13 组	数据基座 • 数据结构化组	第 13 章
第 14 组	数据基座 • 智能问答与图检索组	第 14 章
第 15 组	数据基座 • 虚假内容识别与对抗训练组	第 15 章

第 1 组　六目拼接 · 匹配决策组（九头之一：匹配与几何）

本章地图

讲什么：第 1 组任务书——policy / mc / geometry 三个参数头各管什么、四个月里每个实验做到什么程度、最后交哪几样东西。**为什么值得读**：探针 12/12、A/B 对比、metrics.json 这些验收线，每一条都对应成绩单上的一栏；先读懂再动手，不做无用功。**学完你能**：向任何人说清三个头的输入输出与训练目标，照命令级步骤独立完成四次实验，知道每一步该留下什么证据。

一句话概括你们组：拼接流水线的“第二步半”——特征匹配之后、几何对齐前后，三只小网络替老师傅做三个决定：这一对画面**怎么拼**（policy）、**信几分**（mc）、**几何参数给多少**（geometry）。九个参数头里最靠前的一环，归你。

1.1　组档案

表 1.1　第 1 组档案

项目	六目拼接（项目一，360Stitcher实时全景系统，详见第 一 章）
组号与组名	第 1 组 • 匹配决策组
人数与投入	5 人；每周合计 6–8 小时（组内自行排班，建议每周固定 2 次）
负责的头	policy（匹配策略）、mc（匹配质量分）、geometry（几何参数）
上游依赖	开局即可训练：组 4 组的公开数据仿真标签（UDIS-D/COCO 图对喂 simulation 管线产出）；真机标签实验四接入
下游依赖	第 5 组：你的三个头交付后指标入双台账
工作目录	`stitcher_cpp/training/group01_policy`、`group04_mc`、`group05_geometry`
硬件要求	本机无 GPU 亦可：三个头都是几百个参数的小网络，CPU 训得动

先排一颗雷：三个工作目录名里的数字（group01 / group04 / group05）是源工程里参数头的**原始编号**，与本课程“第 1 组”无关——别看到 `group04_mc` 就以为自己走错了门。第 1 组的三个头分散在三个目录里，每个目录自带 `train.py`，结构相同，学会一个就会三个。

1.2　项目背景与你的位置

第 一 章讲过来龙去脉：360Stitcher 把六路视频实时拼成 360° 全景，五步曲的第二步“特征提取与匹配”、第三步“几何对齐”，至今靠人工经验参数撑着——ORB 取 1500 点、比例阈值多大、RANSAC 距离阈值多严，全是老师傅手调的写死的数。这学期你们的任务，就是把这三个决定交给数据。你在流水线里的位置：六路画面完成畸变校正、提完 ORB 特征之后，**先由 mc 给这对画面的匹配质量打分**（分太低就不硬拼，自动退回老参数），**再由 policy 三选一定拼接策略**，**最后由 geometry 给出几何参数**供对齐使用——三个头首尾相接，是整条九头链路的入口。下图圈出了你的三块地盘。

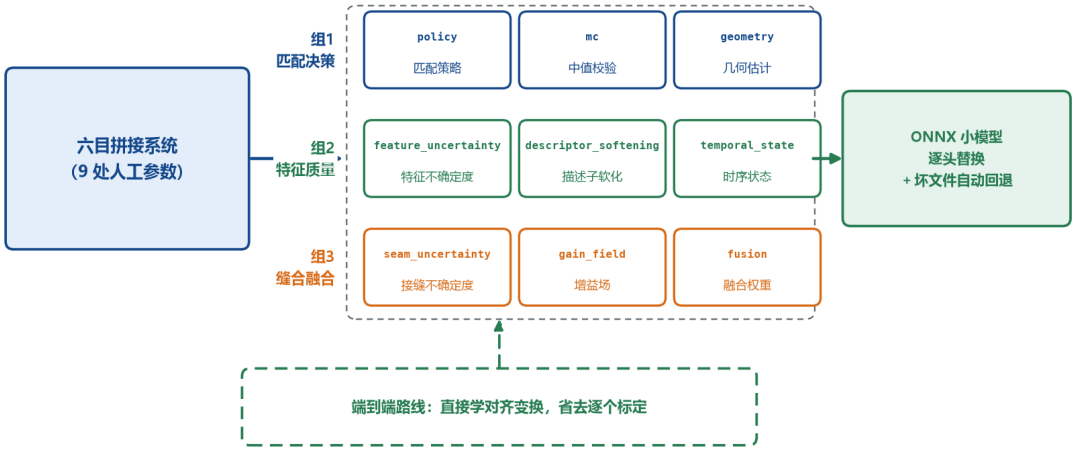


图 1.1　九头地图上的第 1 组：policy / mc / geometry 三个头（匹配决策组）站在九头链路的最前端，吃匹配统计量、吐策略与参数，每个头独立训练、独立交付

1.3　核心任务

三个头一张表看明白，随后逐头说人话。

表 1.2　第 1 组三个参数头：输入 → 输出与人话职责

头名	形状（输入 → 输出）	一句话职责	第一篇血缘
policy	[B,8] → [B,3]	读 8 项匹配统计量，三选一决定这条接缝走哪条拼接策略	第 1–3 讲
mc	[B,10] → [B,1]	给本轮特征匹配打一个置信分，分低就别硬拼	第 0–3 讲
geometry	[B,16] → [B,5]	从 16 项统计特征回归 5 个几何参数，接管手调阈值	第 2–3 讲

方括号是张量形状：B 是一次一起送进网络的一批样本数（batch），[B,10] 就是“一批若干行、每行 10 个数”。逐头展开：**policy** 是一道三分类题——同一个重叠区，可以用常规策略拼、用保守策略拼、或者干脆放弃这帧，网络的输出是三选一；**mc** 是一道回归题——输入 10 项匹配统计量（内点数量、内点率、得分分布等），输出一个 0 到 1 的质量分，它是系统的“信心开关”；**geometry** 还是回归题——从 16 项统计特征预测 5 个几何参数，把“老师傅手感”变成确定数。每个输入列具体是什么字段，以 `train.py` 读到的数据列与 `metrics.json` 的字段为准——这正是实验一要你亲手弄清楚的事。

项目连线：你的三个头与第一篇的哪几讲

三个头都是“统计量进、几个数出”的小型全连接网络：骨架是第 1 讲的感知器，引擎是第 2 讲的反向传播；训没训好、会不会过拟合，全部按第 3 讲的验证集方法论执行；mc 打分的直觉原点是第 0 讲的“点积打分器”。参数量比第 6 讲的 LeNet 还小一个量级，所以本机无 GPU、CPU 训练毫无压力——训练瓶颈从来不是算力，而是数据与耐心。

1.4　第 19 周周五提交清单

提交线：2027年 1 月 22 日（第 19 周周五）完全提交离场。你的组交齐下表五样，每样都与第 三 章的总清单模板对应。

表 1.3　第 1 组提交清单（2027 年 1 月 22 日）

交付物	格式	说明
三头模型与指标	6 个文件	`mc.onnx` + `mc.metrics.json`、`geometry.onnx` + `geometry.metrics.json`、`policy.onnx` + `policy.metrics.json`；文件名 = 头名，拷入 `stitcher_cpp/build/Release/models/`，交付后探针须仍 `PASS (12/12)`
训练证据包	zip	四次实验各一包，命令 + 日志 + 产物三件套，日志为终端完整输出
A/B 对比报告	PDF，2–4 页	`ab_compare.py` 的 before/after 数据表与结论，含 mc 三版对比
小组项目报告	PDF，8–15 页	按第 三 章模板：摘要/背景/方法/实验/结论/分工声明
AI 使用声明	一页	哪些事 AI 做了、人做了什么核验，逐条列清

1.5　AI 分工与人工必做

分工原则一句话：AI出草稿，人出签字——签字的人对数字负全责。

表 1.4　第 1 组 AI 分工与人工必做

AI可以自动做	人必须亲自做
解读 `train.py` 与 `metrics.json` 字段含义	本机跑通全部命令：编译、训练、探针、A/B
解释报错信息、给排查建议	逐条核对 AI 结论（在代码里找到对应行才算数）
写数据预处理与画图脚本草稿	拷贝交付物到 `models/` 并重跑探针确认
起草报告与 A/B 报告的文字段落	在 metrics.json 与证据包上签字，对数字担责

1.6　防糊弄指南与满分标准

防人工糊弄指南（第 1 组版）

(1) 跑一次成功就截图当十次的证据，不行——四个节点各有各的验收线，一次表演补不齐全部证据。**(2)** 改日志时间戳、拼接不同日期的输出 = 作弊。**(3)** 组内“代跑”必须留运行者姓名与时间，谁跑的谁负责。**(4)** 编造训练精度 = 该节点成绩作废：metrics.json 里的每个数字都要能与训练日志对上账，对不上就是作废，没有商量。

防 AI 糊弄指南（第 1 组版）

(1) AI 给的数字必须本机复跑——它随口报的 loss 与准确率，只有你终端里的那份是真的。**(2) AI 解释字段爱“脑补”**：8 项匹配统计量到底是哪 8 项，以代码与数据文件为准，它背的教科书不算。**(3) AI 说“这样应该可以跑通”不等于跑通了**，跑没跑通只看你的终端。**(4)** 禁止把 AI 的回答直接粘进实验记录当结论。

成果满分 50 分，怎么拿每一分见下表。

表 1.5　第 1 组成果满分标准（50 分制）

分值	项目	怎么拿满
20	功能达成	实验三交付 mc 头并完成 A/B（8 分）；实验四交付 geometry、policy 两头（7 分）；每次交付后重跑探针仍 `PASS (12/12)`（5 分）
10	证据完整	四次实验证据包三件套齐全、可复现；缺一件扣 5 分
8	报告质量	组报告按模板 8–15 页，方法讲得清、图表齐全、结论有 A/B 数据支撑
7	协作与台账	排班执行有记录；组 5 双台账中本组条目无缺失；求助记录规范
5	加分	如：为 mc 设计新的统计特征并用 A/B 证明提升；对三个头做联合分析并给出数据

1.7　四次实验安排

四个节点与全班同频，但每场都有本组的命令级动作。先看总览表，再逐场展开；每场实验按“目标—步骤—验收证据—常见坑”执行：步骤照抄就能跑，证据一栏就是成绩。

表 1.6　第 1 组四次实验总览

节点	日期	本组动作	核心验收证据
实验一	10月 12 日	agent 解读 `train.py` 三问，人工核对	解读 + 核对记录一页
实验二	11 月 2 日	本机部署 + 基准 + 探针	三段日志 + `PASS (12/12)`
实验三	11 月 23 日	`--smoke`、`--telemetry` 训练并交付 mc，A/B	训练日志、`metrics.json`、A/B
实验四	12 月 21 日	交付 geometry、policy，真机标签复训 mc 联调	两头交付 + 三版 mc 对比表

1.7.1　实验一（10 月 12 日）：让大模型替你读代码，你来当裁判

目标：配好大模型编程助手的 agent 模式，让它解读 `group04_mc/train.py`，你负责核对它说得对不对——这是全组整个学期训练功课的第一份预习。

步骤：(1) 按课堂演示装好编程助手并切换到 agent 模式；(2) 打开 `stitcher_cpp/training/group04_mc/train.py`，让 AI 回答三个问题：数据从哪来（读哪个目录、哪些文件、哪些列）、网络结构长什么样（几层、每层多少神经元、什么激活函数）、输出是什么（形状与每个数的含义）；(3) 把 AI 的回答

逐条与代码对照，标注“对 / 错 / 存疑”；(4) 至少纠正 AI 一处错误，写明你在代码里找到的依据。

验收证据：一页“AI解读 + 人工核对”记录，含 AI 原话、你的核对结论、纠正错误至少一处。

常见坑：AI 会顺手“脑补”——比如说脚本用了 GPU 或 dropout，而它其实就是 CPU 上的小网络；提示词里不写清文件名，它就给你背教科书；只问一遍就信是最大禁忌——换个问法再问一遍，两次答案打架的地方，就是你必须翻代码的地方。

1.7.2　实验二（11 月 2 日）：本机部署，亲眼看见九个头上线

目标：在自己电脑上把 360Stitcher 编译出来，看到九个头以 ONNX 形式挂上系统，并用探针确认部署无误。

步骤：(1) 用 VS2022 打开 `stitcher_cpp/build/360stitcher.sln`，把配置切到 `Release` 后整体生成；(2) 命令行进入 `build/Release` 目录，运行 `360stitcher.exe --benchmark-stitch=deploy/simulation --provider=onnx`，日志应出现九行 `[Provider]` 加载记录（形如 `wired: orb=1500`）；(3) 再运行 `python tools/nine_head_probe.py`，期望输出 `PASS (12/12)`。

验收证据：编译成功末段日志、九行 [Provider] 日志、探针 `PASS (12/12)`，三段终端完整输出。

常见坑：配置忘了切 Release，Debug 版一路报错找不到 dll；探针 FAIL 先查 `models/` 下模型文件名是否与头名完全一致（探针按文件名找文件）；杀毒软件可能静默隔离 onnx 文件；不在 `build/Release` 目录下运行，程序找不到 `deploy/simulation` 基准图。

1.7.3　实验三（11 月 23 日）：mc 头真训练、真交付、真对比

目标：完整走通第一个头的全流程——先冒烟验证，再真数据训练，交付上系统，用 A/B 数据说话。

步骤：(1) 进入 `training/group04_mc`，先跑 `python train.py --smoke`——小数据几秒钟跑通全链路，验证代码与数据格式无误；(2) 再跑 `python train.py --telemetry`，用组 4 的公开仿真标签训练（真机标签实验四替换），记下最终指标并生成 `metrics.json`；(3) 把 `mc.onnx` 与 `mc.metrics.json` 拷入 `stitcher_cpp/build/Release/models/`，重跑探针必须仍 `PASS (12/12)`；(4) A/B 对比：`python tools/ab_compare.py before.json after.json`，before 是换模型前的基准结果，after 是换模型后的。

验收证据：`--smoke` 与 `--telemetry` 两段完整训练日志、`metrics.json`、探针 PASS、A/B 对比结果，四个证据缺一不可。

常见坑：跳过 `--smoke` 直接真训，数据少一列报错能排查半小时——冒烟 5 秒钟，救你一晚上；A/B 的文件路径写成相对路径而命令行不在对应目录，工具找不到文件；交付时顺手改了文件名（如 `mc_v2.onnx`），探针立刻 FAIL——文件名 = 头名，版本信息写进 `metrics.json`，别写进文件名。

1.7.4　实验四（12 月 21 日）：补齐两头，并用新标签复训 mc

目标：交付 geometry 与 policy 两个头；真机标签联调——组 4 录制的真实标签接入，重训 mc 得第三版，公开版/真机版 A/B 对账，用三版数据回答“数据变多到底管不管用”。

步骤：(1) 按实验三的流程（先 `--smoke` 再 `--telemetry`）先后训练并交付 geometry、policy；(2) 组 4 录制的真实标签（真机标签）接入，重训 mc 得到第三版；(3) 做三版 mc 对比表：基线（人工参数）/ 实验三公开仿真标签版 / 真机标签复训版，各列关键指标与 A/B 结果，公开版/真机版逐项对账，并写一段不少于 100 字的结论。

验收证据：两个头的训练日志与 `metrics.json`、交付后探针 PASS、三版 mc 对比表加结论段。

常见坑：复训覆盖旧模型前忘了保留 before.json——没有旧版数据就没有对比；三个头想一口吃成胖子，最后一头都没训完——按 mc → geometry → policy 串行推进，每个头交付验收后再开下一个。

第 2 组　六目拼接 · 特征质量组（九头之二：特征与时序）

本章地图

讲什么：第 2 组任务书——三个“特征质量”参数头的规格怎么读、时序头怎么训、两个图像头在数据到位前该干什么。**为什么值得读**：你们组的三个头形状最杂（有一维统计量、有 32×32 特征图、还有时序窗口），对应三种不同的网络——任务书把每种怎么落地写成了命令级步骤。**学完你能**：独立读懂 `head_specs.py` 的规格定义，训好并交付 temporal_state 头，把两个图像头从设计文档一路带到真训练交付。

一句话概括你们组：特征环节的“质检科”。第 1 组的 mc 头给“一对画面”打分，你们管得更细：一个头给**特征图本身**挑毛病（feature_uncertainty），一个头给**二进制指纹**学会看场合（descriptor_softening），一个头盯着**最近一段时间**判断跟踪状态（temporal_state）。

2.1　组档案

表 2.1　第 2 组档案

项目	六目拼接（项目一，360Stitcher实时全景系统，详见第 一 章）
组号与组名	第 2 组 • 特征质量组
人数与投入	5 人；每周合计 6–8 小时（组内自行排班，建议每周固定 2 次）
负责的头	feature_uncertainty、descriptor_softening、temporal_state
上游依赖	开局即可训练：组 4 组公开仿真时序标签 + HPatches/MegaDepth 公开 patch（32×32，与正式规格一致）；总体组真 patch 约第 11 周到位，实验四迁移转真
下游依赖	第 5 组：三个头交付后指标入双台账
工作目录	`stitcher_cpp/training/` 下本组三个头的子目录；三头规格的单一事实来源是 `training/common/head_specs.py`
硬件要求	本机无 GPU 亦可：三个头都是小网络，CPU 训得动

特别记一笔你们组的“数据节奏”：三个头的训练数据开局就有——temporal_state 吃组 4 组的公开仿真时序标签，两个图像输入头用 HPatches/MegaDepth 公开 patch 先行训练（32×32 与正式规格一致）；总体组在第 11 周前后提供的真 patch 数据，用于实验四的迁移转真。所以你的三条线从第一周就是“训练态”——公开版与真机版的指标在报告里分开写、如实注明数据来源。

2.2　项目背景与你的位置

第 一 章讲过：360Stitcher 的第二步用 ORB 提特征、做匹配，而特征这层一直是“盲盒”——提取出来的特征图质量如何、256 位二进制指纹可不可信、帧间跟踪的惯性还灵不灵，系统全凭经验假设。你们的任务就是给这个盲盒装上仪表盘。你在流水线里的位置：五步曲第二步的“质检岗”——

feature_uncertainty 扫一眼特征图，报出这帧特征有多不可靠；descriptor_softening 给指纹比对加软权重；temporal_state 读最近 T 帧，告诉跟踪器“该信还是要重置”。三个头的意见汇总起来，决定这一帧的特征信息被采信几分。

2.3　核心任务

三个头一张表，随后逐头说人话。

表 2.2　第 2 组三个参数头：输入 → 输出与人话职责

头名	形状（输入 → 输出）	一句话职责	网络类型
feature_uncertainty	[B,1,32,32] → [B,3]	扫一遍 32×32 特征图，报出 3 个“这帧特征有多不可靠”的指标	CNN
descriptor_softening	[B,1,32,32] → [B,256]	给 256 位描述子的每一位输出“是 1 的概率”，让生硬的位比对学会看场合	CNN
temporal_state	[B,T,24] → [B,4]×2	读最近 T 帧的 24 维时序特征，输出两个 4 维状态：跟踪该信谁、要不要重置	时序网络

descriptor_softening 最值得细品：BRIEF 指纹是 256 位 0/1，传统比对是“数不同位”（汉明距离）——硬碰硬，一位就是一位，没有商量。软化头给每一位输出一个 0 到 1 的概率 $P(\text{bit}=1)$，比对时按置信度加权：纹理清晰区域的位多说话，模糊区域的位少说话。匹配从“非黑即白”变成“看场合说话”。**temporal_state** 则是把第 9–10 讲的东西搬进真系统：滑动窗口读最近 T 帧的 24 维统计特征，输出两个 4 维向量，一个告诉帧间跟踪“惯性还剩几分”，一个告诉系统“要不要按下重置键”。三个头的精确规格——输入尺寸、输出含义、损失函数——全部写在 `training/common/head_specs.py`，那是规格的**单一事实来源**：AI 说的、你记的，都以它为准。

项目连线：一组人吃两种网络，正好把两条课线走通

temporal_state 是时序网络，第 9 讲循环神经网络的“按时间展开”就是它的骨架，第 10 讲的门控记忆告诉你怎么“该记的记、该忘的忘”；feature_uncertainty 与 descriptor_softening 是 CNN，第 5 讲的卷积与特征图、第 6 讲的训练技巧全覆盖。还有个巧合值得会心一笑：两个图像头的输入恰好是 32×32 patch——正是第 6 讲 LeNet 的原生输入尺寸。你组等于把“卷积 + 时序”两条课线各落地一遍。

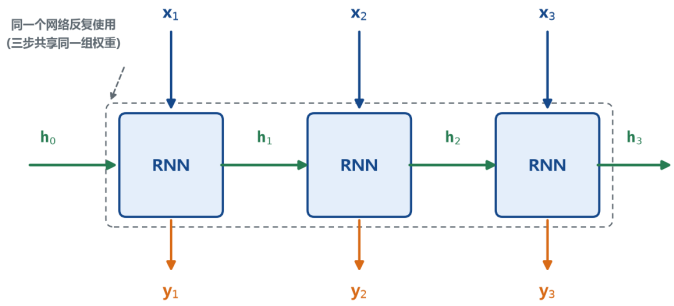


图 2.1　temporal_state 头的骨架：同一模块按时间步展开，隐藏状态把“最近发生了什么”一路传下去——第 9 讲的知识直接上岗

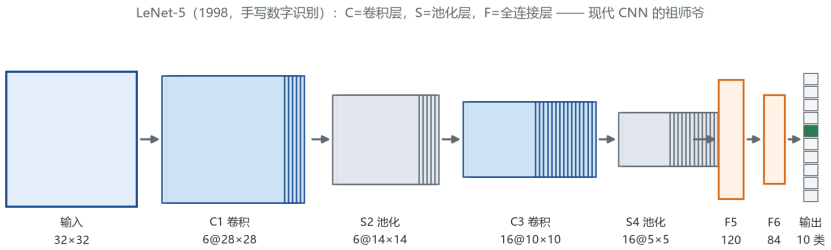


图 2.2　两个图像头的参照物 LeNet-5：输入恰好是 32×32——你的 patch 头与经典 CNN 同尺寸起步

2.4　第 19 周周五提交清单

提交线：2027年 1 月 22 日（第 19 周周五）完全提交离场。你的组交齐下表六样。

表 2.3　第 2 组提交清单（2027 年 1 月 22 日）

交付物	格式	说明
三头模型与指标	6 个文件	三个头各一套 `.onnx` + 同名 `.metrics.json`；文件名 = 头名，拷入 `stitcher_cpp/build/Release/models/`，交付后探针须仍 `PASS (12/12)`
规格说明书	一页	三个头的输入尺寸/输出含义/损失函数，以 `head_specs.py` 为准（实验一产物，随报告附上）
图像头设计文档	每头一页	feature_uncertainty 与 descriptor_softening 的网络设计（实验三产物）
训练证据包	zip	四次实验各一包，命令 + 日志 + 产物三件套
A/B 对比报告	PDF，2–4 页	`ab_compare.py` 的 before/after 数据表与结论
组报告与 AI 声明	PDF 8–15 页 + 一页	按第 三 章模板；AI 使用声明逐条列清

2.5　AI 分工与人工必做

分工原则一句话：AI出草稿，人出签字——签字的人对数字负全责。

表 2.4　第 2 组 AI 分工与人工必做

AI可以自动做	人必须亲自做
解读 `head_specs.py` 三头规格并起草说明书	逐条核对规格（与代码逐行对照后才能写进文档）
解释报错、起草设计文档与报告段落	本机跑通编译、`--smoke`、训练、探针、A/B
写时序滑窗与画曲线的脚本草稿	亲眼看训练曲线、抽检 patch 样本
复述 RNN/CNN 原理、翻译术语	拷贝交付物并重跑探针；在证据包上签字担责

2.6　防糊弄指南与满分标准

防人工糊弄指南（第 2 组版）

(1) 图像头数据没到位的那几周，验收标准白纸黑字是“链路通 + 设计成文”——照实写就是满分答案；反过来，在报告里虚构训练精度 = 该节点成绩作废，这是你们组最容易踩的雷。**(2)** 跑一次成功就截图当十次的证据，不行；改日志时间戳 = 作弊。**(3)** 组内“代跑”必须留运行者姓名与时间。**(4)** 时序数据“先打乱再训练”是隐形造假——打乱的应该是滑窗样本，不是帧，打错了训练集验证集就串了，指标再高也是假的。

防 AI 糊弄指南（第 2 组版）

(1) AI 念张量形状经常出错：把 [B,1,32,32] 念成“批量 13232”它做得出来——每个维度什么含义，以 `head_specs.py` 为准。**(2) AI 给的数字必须本机复跑：**loss、指标，只有你终端里的那份是真的。**(3) AI 说“应该可以跑通”不等于跑通了，**跑没跑通只看你的终端输出。**(4)** 禁止把 AI 的回答直接粘进实验记录与规格说明书——说明书里的每句话都要能在代码里指出出处。

成果满分 50 分，怎么拿每一分见下表。

表 2.5　第 2 组成果满分标准（50 分制）

分值	项目	怎么拿满
20	功能达成	实验三交付 temporal_state 并完成 A/B（8 分）；实验四交付两个图像头（7 分）；每次交付后探针仍 `PASS (12/12)`（5 分）
10	证据完整	四次实验证据包齐全可复现，含规格说明书与两页设计文档；缺一件扣 5 分
8	报告质量	组报告按模板 8–15 页；规格说明书与设计文档被助教判定“能照着实现”
7	协作与台账	排班执行有记录；组 5 双台账中本组条目无缺失；求助记录规范
5	加分	如：时序窗口长度 T 的消融对比有数据；给软化头设计新的评测角度并用 A/B 验证

2.7　四次实验安排

四个节点与全班同频，每场按“目标—步骤—验收证据—常见坑”执行。先看总览表，再逐场展开。

表 2.6　第 2 组四次实验总览

节点	日期	本组动作	核心验收证据
实验一	10月 12 日	agent 解读 head_specs.py，写规格说明书	说明书一页 + 核对记录
实验二	11 月 2 日	本机部署 + 探针 + 占位模型对照	三段日志 + wired 值对照
实验三	11 月 23 日	temporal_state 训练交付；图像头公开 patch 先行训练 + 交设计	训练证据 + 两页设计文档
实验四	12 月 21 日	真机 patch 联调：图像头迁移转真交付	两头交付 + 公开版/真机版 A/B

2.7.1　实验一（10 月 12 日）：读懂规格书，写出规格书

目标：用大模型 agent 解读 training/common/head_specs.py 中本组三个头的规格，人手留下一页“规格说明书”——它是你们组后面三个多月随查随用的字典。

步骤：(1) 装好编程助手并切到 agent 模式；(2) 打开 head_specs.py，让 AI 逐头回答三个问题：输入张量每一维是什么、输出的每个数什么含义、用什么损失函数；(3) 对照代码逐条核对，标“对 / 错 / 存疑”，至少纠正 AI 一处错误；(4) 汇成一页规格说明书。

验收证据：一页规格说明书 + AI 解读与人工核对记录。

常见坑：AI 把 [B,T,24] 的 T 说成“固定值”而不解释滑窗怎么来；[B,4]×2 是“两个输出”不是“八维向量”，AI 混着说的不少；提示词不点名文件路径，它就去背教科书——每个问题都把 head_specs.py 写进提示词。

2.7.2　实验二（11 月 2 日）：本机部署，外加一次“占位模型”对照

目标：部署流程与第 1 组一致——编译、看九头上线、跑探针；你们组多做一步：装一遍占位模型，验证“换模型”这件事本身干净无副作用。

步骤：(1) VS2022 打开 stitcher_cpp/build/360stitcher.sln，切 Release 编译；(2) 运行 360stitcher.exe --benchmark-stitch=deploy/simulation --provider=onnx，看到九行 [Provider]（形如 wired: orb=1500）；(3) python tools/nine_head_probe.py 期望 PASS (12/12)；(4) 运行 python tools/make_placeholder_models.py 生成九个占位模型，重跑同一基准，对照装前装后的 wired 值——应当**完全一致**（占位模型是输出恒定的空壳，它不该改变任何行为）。

验收证据：编译与九行 Provider 与探针三段日志；装占位模型前、后两份 wired 值对照记录。

常见坑：占位模型生成后忘了重跑探针；wired 值只截了装完的、没截装前的——对照实验两头都要留证据；忘了切 Release 报错找不到 dll。

2.7.3　实验三（11 月 23 日）：temporal_state 真训练交付，图像头交设计

目标：交付 temporal_state 头；两个图像头用 HPatches/MegaDepth 公开 patch 先行训练（真机 patch 未到位——如实注明数据来源，不冒充真机指标）。

步骤：(1) temporal_state：先 python train.py --smoke 验证链路，再 --telemetry 真数据训练，交付.onnx + .metrics.json 到 build/Release/models/，重跑探针；(2) 两个图像头：各交一页设计文档（网络几层、为什么这么设计、损失函数、打算怎么验证），并用 HPatches/MegaDepth 公开 patch 先

行训练一版（交付前仍跑 `python train.py --smoke` 验证链路）——**不等真数据，公开数据先跑通**，报告里如实标注训练数据来源。

验收证据：temporal_state 的训练日志、`metrics.json`、探针 PASS、A/B 结果；两页设计文档；两段 `--smoke` 完整日志。

常见坑：时序数据“先打乱再切分”——正确做法是先滑窗成样本再切训练/验证集，打乱的是窗口不是帧；图像头开局就在训练——公开 patch 先行训练与设计文档正是这几周的作业，真数据一到就能迁移。

2.7.4　实验四（12 月 21 日）：真机 patch 联调，图像头迁移转真

目标：总体组真机 patch 数据到位（合并联调）：把两个图像头从“公开 patch 预训版”迁移成“真数据版”，并做公开版/真机版 A/B 对账。

步骤：(1) 先抽检 20 张 patch 肉眼过目——确认是有效样本而不是空图与重复图；(2) 两个头先后 `--smoke`、`--telemetry` 训练；(3) 交付 `.onnx` + `.metrics.json` 并重跑探针仍 `PASS (12/12)`；(4) 各跑 A/B：`python tools/ab_compare.py before.json after.json`。

验收证据：两个头的训练日志、`metrics.json`、探针 PASS、A/B 结果。

常见坑：数据刚到位就全量训练——先抽检再放量；样本量小先跑 `--smoke` 确认 batch 维度对得上，别到真训练才发现形状不匹配。

第 3 组　六目拼接 · 缝合融合组（九头之三：缝合与融合）

本章地图

讲什么：第 3 组任务书——缝合线、增益补偿、融合三处“老师傅手艺”怎么换成网络，以及为什么 fusion 头是必须压轴的最后一块拼图。**为什么值得读**：你们组要读懂的代码从 Python 换成了 C++，还要管一次全项目最复杂的上游依赖（fusion 头要等其余八个头全部上岗才能开训）。**学完你能**：读懂 `stitch_initialize.cpp` 里的补偿与融合逻辑，交付 gain_field 与 fusion 两个头，为 seam 头画好图纸。

一句话概括你们组：流水线“最后两步”的质检与手感——缝合线**走哪**（seam_uncertainty 给动态规划递眼色）、画面亮度**怎么拉齐**（gain_field 接管补偿）、两幅画面**怎么混**（fusion 定权重）。前两步的老师傅手艺，这学期开始一件件换成学出来的参数。

3.1　组档案

表 3.1　第 3 组档案

项目	六目拼接（项目一，360Stitcher实时全景系统，详见第 一 章）
组号与组名	第 3 组 • 缝合融合组
人数与投入	5 人；每周合计 6–8 小时（组内自行排班，建议每周固定 2 次）
负责的头	seam_uncertainty、gain_field、fusion
上游依赖	开局即可开工：gain_field/seam 用组 4 组公开仿真标签；fusion 彩排用公开数据，正式版等八头齐 + 总体组第 13 周前后放行
下游依赖	第 5 组：三个头交付后指标入双台账
工作目录	`stitcher_cpp/training/` 下本组三个头的子目录；实验一解读对象是 `stitcher_cpp/src/stitch/stitch_initialize.cpp`
硬件要求	本机无 GPU 亦可：三个头都是小网络，CPU 训得动

你们组的排班是“两头开局就跑、压轴先彩排”：gain_field 与 seam_uncertainty 用公开仿真标签第 1 周就能训练；fusion 虽然正式版要等八头齐 + 总体组放行（约第 13 周），但放行前用公开数据把训练管线与超参先彩排一遍——放行当天就能开工。

3.2　项目背景与你的位置

第 一 章讲过五步曲的后两步：第四步用动态规划在重叠区里找一条差异最小的**缝合线**，第五步用增益补偿加多频段/feather 融合把接缝抹到看不见——这两步至今是写死在 C++ 里的手艺：缝合代价怎么算、增益系数怎么调、过渡带多宽，全是工程师凭经验定的数。你们的任务，就是让这三处由数据说话。你在流水线里的位置：全系统的**出口段**——匹配靠不靠得住由第 1 组打分，特征靠不靠

得住由第 2 组把关，而观众最终看到的画面好不好，接缝穿不穿帮、亮度匀不匀，都从你们手里过。三个头按下表分工。

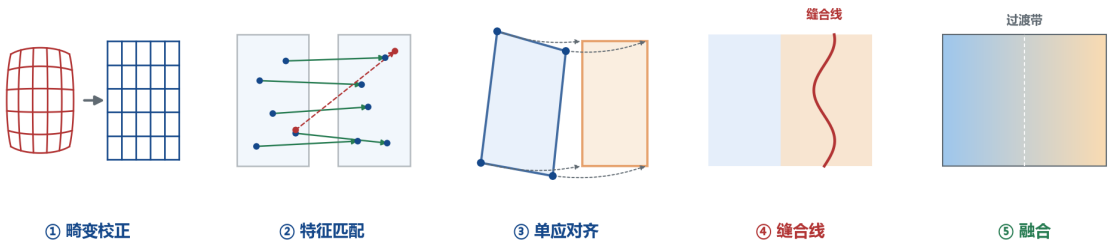


图 3.1　全景拼接五步曲上的第 3 组：seam_uncertainty 给第四步的缝合线“递眼色”，gain_field 与 fusion 接管第五步的增益补偿与融合——五步曲的最后两步归你

3.3　核心任务

表 3.2　第 3 组三个参数头：输入 → 输出与人话职责

头名	形状（输入 → 输出）	一句话职责	网络类型
seam_uncertainty	[B,2,128,256] → [B,1,128,256]	逐像素标注重叠区里“哪里走缝容易穿帮”，给动态规划的代价图加上学习式权重	CNN
gain_field	[B,12] → [B,6]	从 12 个亮度统计量算 6 个增益系数，接管手写补偿逻辑	小回归网络
fusion	[B,12] → [B,3]	从 12 项统计量算 3 个融合权重，决定多频段与 feather 怎么混	小回归网络

逐头展开。**seam_uncertainty** 是三个头里唯一的“图像进、图像出”：输入重叠带（128×256 的区域，2 通道，具体含义以规格与代码为准），输出一张同尺寸的“穿帮风险图”——动态规划找缝合线时照着它绕道，行人、车灯这些高频穿帮点被自动避开。它和第 8 讲的 FCN 是同一思路：逐像素预测。**gain_field** 与 **fusion** 是两只“统计量进、几个数出”的小回归头，与第 1 组的头是同一物种，只是吃的统计量不同：前者吃 12 个亮度统计量（各路画面的均值方差与重叠区亮度差等），后者吃 12 项融合相关统计量。**fusion 还有个特殊身份：九头链路的最后一块拼图**——它的输入统计量要算在“其余八个头都上岗”的新系统上；头一换，统计量的分布就变，所以 fusion 的**正式版**必须等八头齐、总体组第 13 周前后放行后才能开训。但放行前允许用公开数据**彩排**：把训练管线、超参、验收动作先跑顺，彩排版标注“公开数据彩排”、不进台账成绩——区分彩排与抢跑的标就是标注与用途，不是做不做。

项目连线：出口段的三个头与第一篇的哪几讲

seam_uncertainty 是逐像素预测，第 5–6 讲的卷积网络打底，第 8 讲 FCN 的“热图输出”就是它的样子；gain_field 与 fusion 是第 1–3 讲的小回归头，方法论的根在第 3 讲：留验证集、看曲线、防过拟合。而“fusion 等八头”这件事本身，是第 3 讲“数据分布要一致”原则的工程版：训练数据长什么样，上线时就该长什么样。

3.4　第 19 周周五提交清单

提交线：2027年 1 月 22 日（第 19 周周五）完全提交离场。你的组交齐下表六样。

表 3.3　第 3 组提交清单（2027 年 1 月 22 日）

交付物	格式	说明
三头模型与指标	6 个文件	三个头各一套 `.onnx` + 同名 `.metrics.json`；文件名 = 头名，拷入 `stitcher_cpp/build/Release/models/`，交付后探针须仍 `PASS (12/12)`
seam 头设计文档	一页	网络结构与依据、损失函数、验证方案（实验三产物）
训练证据包	zip	四次实验各一包，命令 + 日志 + 产物三件套；实验二的基线计时一并入包
A/B 对比报告	PDF，2–4 页	`ab_compare.py` 的 before/after 数据表与结论，覆盖 gain_field 与 fusion
组报告与 AI 声明	PDF 8–15 页 + 一页	按第 三 章模板；AI 使用声明逐条列清

3.5　AI 分工与人工必做

分工原则一句话：AI出草稿，人出签字——签字的人对数字负全责。

表 3.4　第 3 组 AI 分工与人工必做

AI可以自动做	人必须亲自做
解读 `stitch_initialize.cpp` 的补偿与融合段落	本机跑通编译、基准、训练、探针、A/B
起草 seam 头设计文档与报告段落	核对 AI 对 C++ 的每句解释（落到行号才算数）
写日志解析与计时画图脚本草稿	亲眼记录 luminance/blend 基线计时数字
解释报错、复述 DP 与融合原理	拷贝交付物并重跑探针；在证据包上签字担责

3.6　防糊弄指南与满分标准

防人工糊弄指南（第 3 组版）

(1) 实验二的基线计时（luminance、blend）是实验四 A/B 的“before”，**必须至少跑三遍取稳定读数**，只跑一遍就当基线 = 埋雷。**(2) 区分 fusion 的“彩排”与“正式”**：放行前用公开数据彩排不仅允许、还鼓励，但彩排版必须标注“公开数据彩排”且不得冒充正式版交付——把彩排版当正式版交，按未完成处理。**(3)** 跑一次截图当十次证据不行；改日志时间戳 = 作弊；组内“代跑”必须留运行者姓名。**(4) 编造训练精度 = 该节点成绩作废**——seam 头没数据就照实写“链路通 + 设计成文”，写个编出来的精度反而全丢。

防 AI 糊弄指南（第 3 组版）

(1) AI 读 C++ 爱脑补：变量名、行号、模板参数张口就来——核对时要它引用行号，你打开文件对上才算数。**(2) AI 给的数字必须本机复跑**：计时、指标，只有你终端里的那份是真的。**(3) AI 说“应该可以跑通”不等于跑通了**，跑没跑通只看你的终端输出。**(4)** 禁止把 AI 的回答

直接粘进实验记录与设计文档——文档里每句结论都要有代码行号或日志页码撑腰。

成果满分 50 分，怎么拿每一分见下表。

表 3.5　第 3 组成果满分标准（50 分制）

分值	项目	怎么拿满
20	功能达成	实验三交付 gain_field（公开仿真标签版）并完成 A/B（8 分）；实验四放行后交付 fusion 正式版，附彩排版与正式版两版记录（7 分）；每次交付后探针仍 `PASS (12/12)`（5 分）
10	证据完整	四次实验证据包齐全可复现，基线计时三遍记录在包；缺一件扣 5 分
8	报告质量	组报告按模板 8–15 页；seam 设计文档达到“能照着实现”的水平
7	协作与台账	排班执行有记录；对第 1 组、第 2 组交付进度的跟踪记录完整；组 5 台账无缺失
5	加分	如：seam 设计文档被总体组采纳进实现；对 fusion 做八头联动分析并给出数据

3.7　四次实验安排

四个节点与全班同频，每场按“目标—步骤—验收证据—常见坑”执行。先看总览表，再逐场展开。

表 3.6　第 3 组四次实验总览

节点	日期	本组动作	核心验收证据
实验一	10月 12 日	agent 解读 `stitch_initialize.cpp` 增益补偿段	解读 + 核对记录 + 参数清单
实验二	11 月 2 日	本机部署 + 探针 + 记录基线计时	三段日志 + 三遍基线表
实验三	11 月 23 日	gain_field 训练交付 + A/B；seam 交设计 + `--smoke`	训练证据 + 设计文档
实验四	12 月 21 日	fusion 正式版：八头放行后用真实输出训练交付（联调）	上游确认 + 彩排/正式两版记录

3.7.1　实验一（10 月 12 日）：让大模型替你读 C++，你来当裁判

目标：用大模型 agent 解读 `src/stitch/stitch_initialize.cpp` 的增益补偿段落，回答三个问题——为什么亮度不均、现在的补偿逻辑是什么、哪些是写死的数。你们的改造对象，今天先看个通透。

步骤：(1) 装好编程助手并切到 agent 模式；(2) 让 AI 定位增益补偿段落，回答三问：为什么各路画面亮度不均（曝光、增益、朝向差异）、现在的补偿按什么逻辑算系数、哪些常数是拍脑袋写死的；(3) 逐条核对——让 AI 引用行号，你打开文件对上才算数；(4) 把“写死的数”列成一张清单，它们就是 gain_field 头未来要接管的对象。

验收证据：一页“AI 解读 + 人工核对”记录，含行号引用、三问答案、写死参数清单。

常见坑：C++ 的模板与宏最容易让 AI 看走眼，别让它把标定参数当成运行时补偿系数；提示词不贴文件路径，它就去讲教科书式的“增益补偿综述”；只问一遍就信——换个问法再问一遍，答案打架的地方翻代码。

3.7.2　实验二（11 月 2 日）：本机部署，顺手记下基线数字

目标：部署流程与第 1 组一致；你们组多做一件只有你们组用得上的事——把日志里增益补偿与融合两段的计时基线记下来。

步骤：(1) VS2022 打开 `stitcher_cpp/build/360stitcher.sln`，切 `Release` 编译；(2) 运行 `360stitcher.exe --benchmark-stitch=deploy/simulation --provider=onnx`，确认九行 `[Provider]`（形如 `wired: orb=1500`）；(3) `python tools/nine_head_probe.py` 期望 `PASS (12/12)`；(4) 从日志里找出增益补偿段（luminance 计时行）与融合段（blend 计时行）的数字，同一基准**至少跑三遍**，记下稳定读数——这是你实验三、四 A/B 的 before。

验收证据：编译、九行 Provider、探针三段日志；三遍基线计时记录表。

常见坑：日志级别没开，计时行压根不打印——先确认日志配置；只跑一遍就当基线；忘了切 Release。

3.7.3　实验三（11 月 23 日）：gain_field 真训练交付，seam 头交设计

目标：交付 gain_field 头并用 A/B 说话；seam_uncertainty 头此时以“链路通 + 设计成文”为验收标准。

步骤：(1) gain_field：先 `python train.py --smoke` 验证链路，再 `--telemetry` 真数据训练，交付 `.onnx` + `.metrics.json` 到 `build/Release/models/`，重跑探针仍 `PASS (12/12)`；(2) A/B：`python tools/ab_compare.py before.json after.json`，与实验二记的基线对照；(3) seam_uncertainty：交一页设计文档（网络几层、为什么、损失函数、怎么验证），并用随机数数据跑 `python train.py --smoke` 证明全链路通。

验收证据：gain_field 训练日志、`metrics.json`、探针 PASS、A/B 结果；seam 设计文档与 `--smoke` 日志。

常见坑：A/B 文件路径写成相对路径而命令行不在对应目录，工具找不到文件；seam 头等数据闲着——设计文档与 `--smoke` 就是这几周的正经作业；`--smoke` 用完的随机数据目录留着别删，那是“链路通”的证据。

3.7.4　实验四（12 月 21 日）：八头齐，fusion 正式版压轴上场（合并联调）

目标：总体组放行后，用其余八个头的真实输出启动 fusion 正式训练并交付；本场同时是八组合并联调——真机标签接入、彩排版与正式版对账，给九头链路装上最后一块拼图。

步骤：(1) 放行前先做“上游验收”：确认第 1 组、第 2 组的头已交付且探针 `PASS (12/12)`，把确认记录写进证据包；(2) 总体组放行后，采集八头在岗时的新统计量作为训练数据；(3) `python train.py --smoke` 后 `--telemetry` 训练 fusion，交付并重跑探针；(4) A/B：用实验二基线与 fusion 上岗后的结果做对比，注意对比的是“八头在岗、fusion 换新”这一件事。

验收证据：上游确认记录、放行后的训练日志、`metrics.json`、探针 PASS、A/B 结果。

常见坑：等放行的几周是彩排期——彩排训练与上游验收清单都在这几周完成，放行当天就能开工；放行时间若推迟，与总体组确认新时间并留书面记录；把彩排版冒充正式版交付，或八头没齐就把旧系统 fusion 当新版交，均按无效处理。

第 4 组　六目拼接 · 数据与标签组

本章地图

讲什么：第 4 组任务书——全项目唯一“喂所有人”的组：遥测怎么变成三段标签、两条硬验收线（各 200 行、各 500 行）怎么达成、数据质量报告怎么写。**为什么值得读**：第 1–3 组的模型上限，从你们交出标签的那天起就定了——garbage in, garbage out，你们的手就是全项目的天花板。**学完你能**：独立跑通遥测校验与标签生成全流程，组织一次真机六路采集，写一份能让人信服的数据质量报告。

一句话概括你们组：别的组**训**模型，你们组**造**模型——九个参数头学得好不好，先看你们喂的标签对不对、够不够。八个小节点名最不起眼、分量最重的一环。

4.1　组档案

表 4.1　第 4 组档案

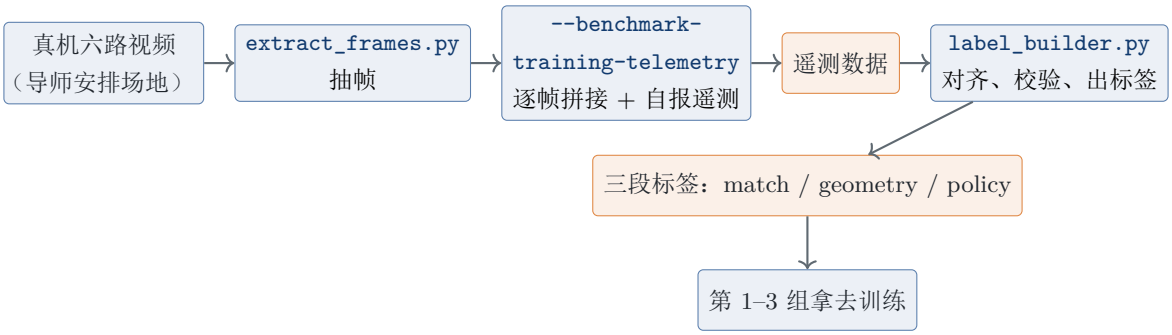
项目	六目拼接（项目一，360Stitcher实时全景系统，详见第 一 章）
组号与组名	第 4 组 • 数据与标签组
人数与投入	5 人；每周合计 6–8 小时（组内自行排班；实验三需随导师赴真机场地，另计）
负责的产出	match / geometry / policy 三段标签 + 数据质量报告（不训练模型，喂所有人训练）
上游依赖	开局任务就续：公开图对（UDIS-D 拼接对、COCO 单应图对）+ 系统自带 simulation 管线，第一周就能产仿真标签；真机场地录制由导师安排（实验三出行）
下游依赖	第 1–3 组第一周就吃你的公开仿真标签；真机标签实验四接入；第 5 组按数据量台账验收
工作目录	`stitcher_cpp/training_data/` 与 `stitcher_cpp/tools/`
硬件要求	本机无 GPU 亦可；本组不做模型训练，瓶颈在组织与细心

4.2　项目背景与你的位置

第 一 章展示过系统运行时的遥测：终端每帧滚出一行 `[Latency]` 日志，系统逐帧“自报家门”。这些自报的家门，加上匹配与几何的中间统计量，就是全项目最宝贵的原料——**把“系统跑的时候自己说了什么”变成“网络学的标准答案”**，就是你们组的全部工作。你在流水线里的位置：最上游。**match 段标签喂第 1 组的 mc 与 policy 头，geometry 段标签喂 geometry 头，policy 段标签喂策略训练**；第 5 组的数据量台账，记的就是你交的每一批。公开仿真标签让三个组第一周就能开训——真机标签决定的是精度上限，不再是开工日期；但标签的真实性依旧是生命线：交得虚一线，三个组的精度集体塌方。

4.3　核心任务

你的产品是三段标签，生产线是一条数据流：



三段标签各管一头：**match 段**记录特征匹配的统计量与结果，是 mc、policy 头的口粮；**geometry 段**记录几何统计量与参数，喂 geometry 头；**policy 段**记录策略相关字段。每行一个样本，**行数就是数据量，字段质量就是标签质量**。课程给两条硬验收线：实验三 **match / geometry 各 $\geq$ 200 行并过校验**，实验四扩量到**各 $\geq$ 500 行**外加一份数据质量报告。注意下图的遥测实录——那行 `[Latency]` 就是你们原料的模样，以后你看到它该像厨师看到食材。

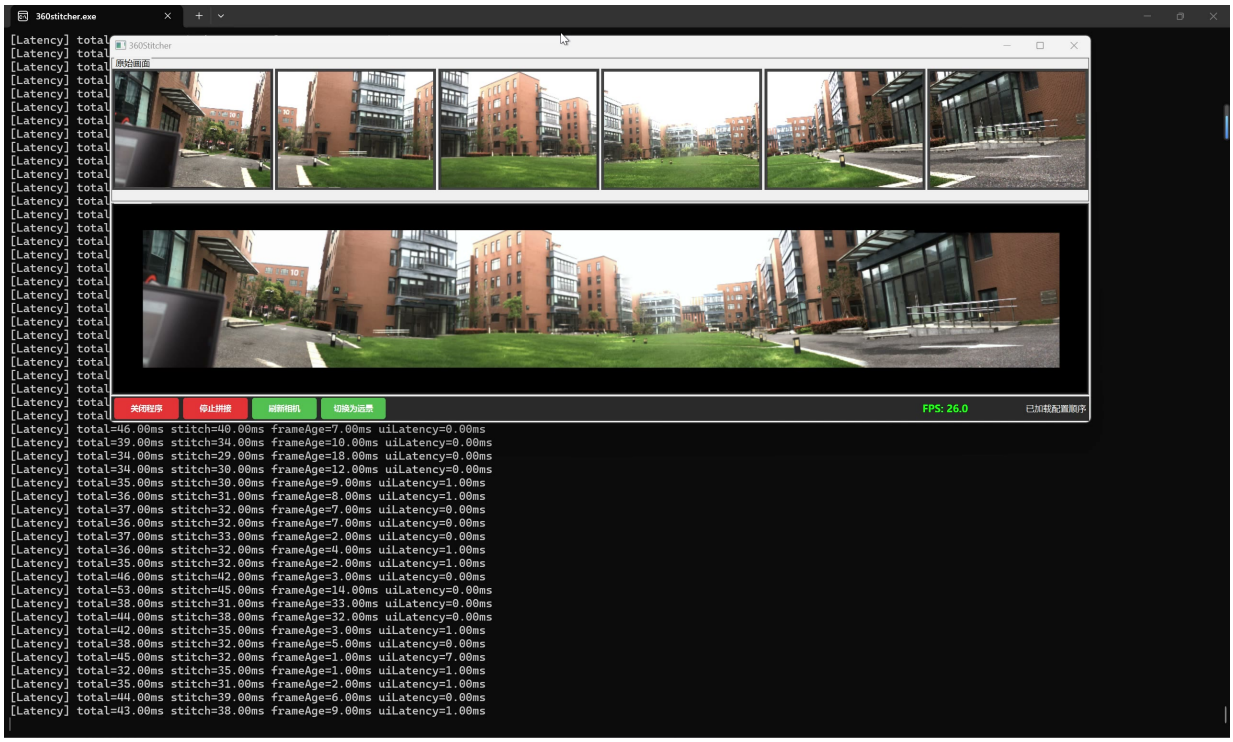


图 4.1　你们的原料长这样：系统运行遥测实录，终端每帧打印一行 [Latency] 日志（total / stitch / frameAge / uiLatency）——label_builder.py 的上游就是这股数据流

图片来源：360Stitcher 项目组真实运行截图（遥测终端实录）

项目连线：数据组与第一篇的哪几讲

第 6 讲“数据增强与训练技巧”里有半章在讲数据质量与分布——那正是你们的主战场：训练/验证怎么切、异常样本怎么处理、分布偏不偏，全是这一讲的方法论。第 3 讲的评估思想也站在你这边：校验器（`check_telemetry.py`）就是你给上游数据做的“验证集体检”。而“数据决定上限”这句话的出处，是第 2 讲就埋下的伏笔——反向传播只能从数据里学规律，数据里没有的，

它变不出来。

4.4　第 19 周周五提交清单

提交线：2027年 1 月 22 日（第 19 周周五）完全提交离场。你的组交齐下表六样。

表 4.2　第 4 组提交清单（2027 年 1 月 22 日）

交付物	格式	说明
三段标签数据集	CSV（`training_data/`）	match / geometry 各 ≥ 500 行且过校验；policy 段按任务要求
数据流图与脚本说明	一页图 + README	“遥测 → 标签”数据流图（实验一产物）与复现步骤
数据质量报告	PDF，4–8 页	字段分布、异常样本清单与处理依据、训练/验证划分建议
采集证据包	zip	真机视频清单、抽帧统计、遥测校验 `RESULT: PASS`、标签校验日志
组报告与 AI 声明	PDF 8–15 页 + 一页	按第 三 章模板；AI 使用声明逐条列清

4.5　AI 分工与人工必做

分工原则一句话：AI出草稿，人出签字——签字的人对数字负全责。

表 4.3　第 4 组 AI 分工与人工必做

AI可以自动做	人必须亲自做
解读 `label_builder.py` 三段标签的字段含义	核对 AI 的解释（逐字段对上代码才算数）
起草数据流图与 README、写画分布图的脚本	本机跑通校验、抽帧、遥测、标签全流程
起草质量报告的文字段落	亲眼抽检样本、亲手核对行数与校验结果
解释报错、复述遥测字段含义	现场采集组织；在证据包上签字担责

4.6　防糊弄指南与满分标准

防人工糊弄指南（第 4 组版）

(1) 行数凑数 = 掺假：复制已有行、同一帧连抽二十张凑行数，数据量台账一查同源重复就露馅，按如实条款该节点成绩作废。**(2)** 校验不过就手改 CSV——校验器 `check_telemetry.py` 就是为防这个而存在的，改数据不如改流程。**(3)** 现场录制不留场景与时间记录，数据不可追溯 = 白录。**(4)** 异常样本一删了之不留记录，质量报告里每一个被剔除的样本都要有名有姓有理由；编造数据 = 该节点成绩作废，全课程无弹性。

防 AI 糊弄指南（第 4 组版）

(1) AI 数行数会数错：行数、字段数、分布统计，一律以脚本输出为准，AI 口算不算。**(2) AI 画的“分布图”必须由真实 CSV 生成**——它编的直方图漂亮得毫无意义。**(3) AI 说“应该可以跑通”不等于跑通了**，校验器报 `RESULT: PASS` 才算。**(4)** 禁止把 AI 对字段含义的解释直接写

进报告——每个字段以 `label_builder.py` 代码为准。

成果满分 50 分，怎么拿每一分见下表。

表 4.4　第 4 组成果满分标准（50 分制）

分值	项目	怎么拿满
20	功能达成	实验三达成验收线：match / geometry 各 ≥ 200 行并过校验（10 分）；实验四扩量至各 ≥ 500 行且质量报告合格（10 分）
10	证据完整	采集证据包齐全可追溯:视频清单、抽帧统计、两道校验的 PASS 日志；缺一件扣 5 分
8	报告质量	组报告按模板 8–15 页；质量报告字段分布讲得清、异常处理有据
7	协作与台账	排班与现场分工有记录；向第 1–3 组交付有交接记录；组 5 数据量台账无缺失
5	加分	如：质量报告发现并追溯出遥测或标签管线的 bug 且被总体组采纳修复；给下游提供“按场景切片”的标签使用建议

4.7　四次实验安排

四个节点与全班同频，每场按“目标—步骤—验收证据—常见坑”执行。先看总览表，再逐场展开。

表 4.5　第 4 组四次实验总览

节点	日期	本组动作	核心验收证据
实验一	10月 12 日	agent 解读 `label_builder.py`，画数据流图	解读 + 核对记录 + 流程图
实验二	11 月 2 日	基准图走通“遥测 → 三段标签”全流程	`RESULT: PASS` + 标签文件
实验三	11 月 23 日	真机采集 → 抽帧 → 遥测 → 标签	各 ≥ 200 行并过校验
实验四	12 月 21 日	扩量 ≥ 500 行 + 数据质量报告	校验日志 + 质量报告

4.7.1　实验一（10 月 12 日）：读懂标签生产线，画出数据流图

目标：用大模型 agent 解读 `tools/label_builder.py` 中三段标签（match / geometry / policy）的数据来源与字段含义，画出“遥测 → 标签”数据流图——你以后的每一步操作，都建立在这张图上。

步骤：(1) 装好编程助手并切到 agent 模式；(2) 让 AI 逐段回答：三段标签各从哪股遥测来、每列字段什么含义、校验逻辑查什么；(3) 逐字段与代码对照核对，标“对 / 错 / 存疑”，至少纠正 AI 一处；(4) 画“遥测 → 标签”数据流图，工具不限（手绘拍照也行），交图。

验收证据：一页“AI 解读 + 人工核对”记录 + 一张数据流图。

常见坑：AI 对字段含义张口就来——让它引用代码行号；三段标签的字段有同名不同义的情况，AI 最爱混着讲；图只画了主流程没画校验环节——校验是生产线的质检站，必须上图。

4.7.2　实验二（11 月 2 日）：先在基准图上跑通全流程

目标：真机采集之前，先用 `deploy/simulation` 的五张基准图把“遥测 → 三段标签”全流程跑通；再批量喂入 UDIS-D 公开拼接对，产出公开仿真标签交给第 1–3 组——他们的开工口粮就在你这场实验里。

步骤：(1)遥测体检：`python scripts/check_telemetry.py` 后接你的遥测目录参数（如 `training_data/simulation_run1`），期望输出 `RESULT: PASS`；(2) 命令行运行 `360stitcher.exe --benchmark-training-telemetry`（部署流程见第 1 组实验二：VS2022 切 `Release` 编译 `360stitcher.sln`），对五张基准图逐帧拼接并记录遥测；(3) 运行 `label_builder.py` 生成三段标签，跑一遍标签校验，确认行数与字段齐全。

验收证据：遥测校验 `RESULT: PASS` 日志、基准图跑遥测的完整日志、三段标签文件与校验输出。

常见坑：遥测目录路径写错，校验器查了个空目录也“不报错”——先看它统计了多少帧再下结论；标签生成后没跑校验就往下传——校验 5 秒钟，能省下游一晚上；五张基准图验证流程；随后批量喂 UDIS-D 图对（至少 50 对），产出公开仿真标签并过校验，match/geometry 各不少于 200 行——这是第一份交付物，第 1–3 组第一周开训就靠它。真机标签的行数指标留给实验三。

4.7.3　实验三（11 月 23 日）：真机场地采集，交出第一条验收线

目标：随导师到真机场地录制六路视频，走完“抽帧 → 遥测 → 标签”全流程，达成 **match / geometry 各 ≥ 200 行并过校验**。

步骤：(1) 随导师赴真机场地录制六路视频，逐段记录场景与时间；(2) 回校后 `python tools/extract_frames.py` 抽帧，统计抽帧数量；(3) 运行 `--benchmark-training-telemetry` 对抽出的帧逐帧拼接、记录遥测；(4) `python scripts/check_telemetry.py` 体检遥测，期望 `RESULT: PASS`；(5) `label_builder.py` 生成三段标签并过校验，核对行数达标。

验收证据：视频清单（场景/时长）、抽帧统计、遥测校验 PASS、三段标签行数与校验通过日志。

常见坑：现场忘了记每段视频的场景与时间——标签要能追溯，回头对不上就是白录；六路相机没同步，时间戳对不齐，遥测对不上帧，标签全废；抽帧太密全是相似帧——行数够了但“有效多样性”不够，校验会卡。

4.7.4　实验四（12 月 21 日）：扩量与质检，交出最终数据家底

目标：真机标签扩量到**各 ≥ 500 行**并全部过校验，交数据质量报告；本场同时是八组合并联调：真机标签接入第 1–3 组复训、与公开仿真标签做对比报告（公开版 vs 真机版：指标差异与原因）。

步骤：(1) 补充采集或补充抽帧，扩量到验收线以上，全程留证；(2) 逐字段做分布统计，画分布图；(3) 清点异常样本（缺失字段、离谱值、同源重复），逐条记录处理方式与依据；(4) 写数据质量报告：字段分布、异常样本处理、训练/验证划分建议；(5) 向第 1–3 组交接，并在组 5 数据量台账登记。

验收证据：真机标签 500 行以上的校验通过日志、字段分布图、异常样本处理清单、质量报告 PDF、台账登记；另附公开仿真标签与真机标签的对比说明。

常见坑：为凑行数复制样本——台账查同源重复一查一个准；异常样本静默删除不留记录——质量报告必须有名有姓有理由；分布图让 AI “随手画一张”——每一张图都要能从真实 CSV 一条命令重新生成。

第 5 组　六目拼接 · 测试集与评测组

本章地图

讲什么：第 5 组任务书——测试集怎么用哈希“冻结”、基线指标台账怎么记、A/B 与回归评测怎么跑、组间评分怎么汇成离场验收会那张打分表。**为什么值得读**：八个组里唯一“给所有人打分”的组——双台账是全项目“节点验收与成绩评定唯一依据”，而台账在你手里。**学完你能**：亲手冻结一份不可抵赖的测试集，把两条技术路线都部署验收一遍，独立完成一次全绿回归，主持一场有数据支撑的组间评分。

一句话概括你们组：别的组**造**东西，你们组**验**东西——测什么、怎么测、分数怎么算，全项目听你们组的。这叫**数据生产与质量验收分离**：第 4 组生产数据，你们验收数据；各组交付模型，你们验收模型。裁判不下场踢球，但哨子必须过硬。

5.1　组档案

表 5.1　第 5 组档案

项目	六目拼接（项目一，360Stitcher实时全景系统，详见第 一 章）
组号与组名	第 5 组 • 测试集与评测组
人数与投入	5 人；每周合计 6–8 小时（组内自行排班；交付季评测集中，建议交付周加开一次组会）
负责的产出	测试集（冻结版）+ 哈希清单、基线与数据量双台账、A/B 与回归评测报告、组间评分汇总表
上游依赖	第 1–3 组与第 7 组：每一次交付都进你的台账；第 4 组：公开仿真场景（开局即有）+ 录制场景（实验四增补）是你的测试素材库
下游依赖	全体组：A/B 统一用你冻结的测试集；第 三 章离场验收会：你的组间评分汇总表是核心材料
工作目录	`stitcher_cpp/tools/`（`ab_compare.py`、`run_regression.ps1`）
硬件要求	**两条路线都要能部署**：路线一 `360stitcher.exe` 与路线二 `360stitcher_e2e.exe` 都要能在你本机跑起来；无 GPU 亦可

代码基座两个目录都在你机器上：`stitcher_cpp`（VS2022 打开 `360stitcher.sln`，九头路线）与 `e2e_gpu`（端到端路线，自含程序与训练）。别的组可以只守一亩三分地，你必须两边都熟——两条路线的成绩，都从你手里过。

5.2　项目背景与你的位置

第 一 章讲过两条路线的赛马：九头学习路线用九个小参数头逐个接管经验参数，端到端路线用一个网络直接吃图像对、吐对齐变换。一个学期下来，谁进步了、该得多少分——不靠嗓门大，只靠**同一把尺子**：同一份冻结测试集、同一组指标、同一本台账。你在流水线里的位置是**终点站兼裁判席**：

九头交付跑九头探针、端到端交付跑引擎探针，指标进你的台账；两轮回归全绿由你执行；离场验收会那张组间评分汇总表，由你从台账里一笔一笔汇出来。下图就是你们天天打交道的被评对象——六路画面进、一条全景出，你的每一个数，都在给这条全景带的稳、快、好记账。

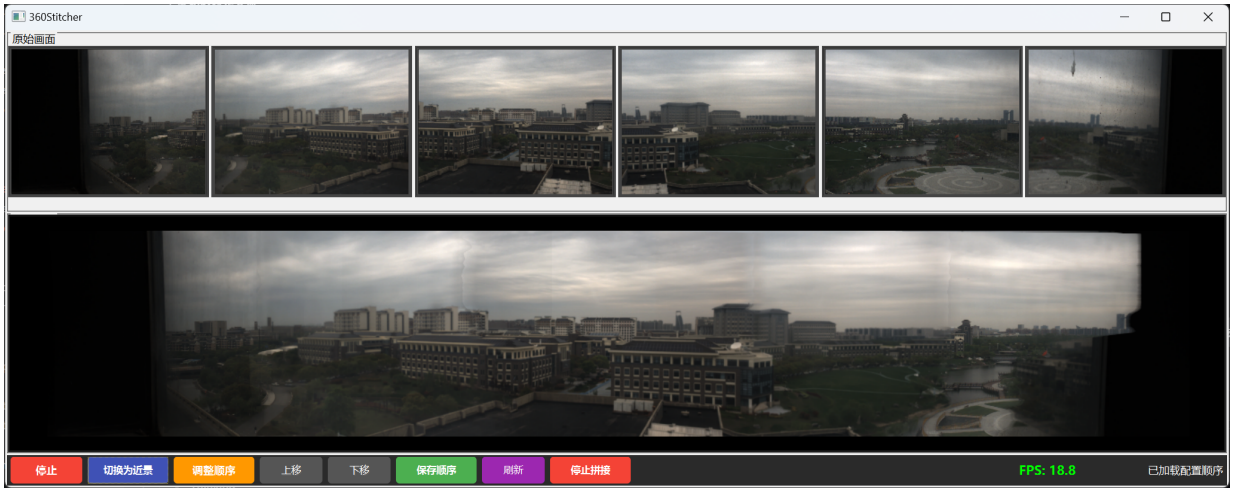


图 5.1　你们的被评对象长这样：360Stitcher 六路原始画面（上排）拼成一条 360° 全景（下排）——测试集里的每张图、台账里的每个数，最终都是在给这条全景带的稳、快、好记账

图片来源：360Stitcher 项目组真实运行截图

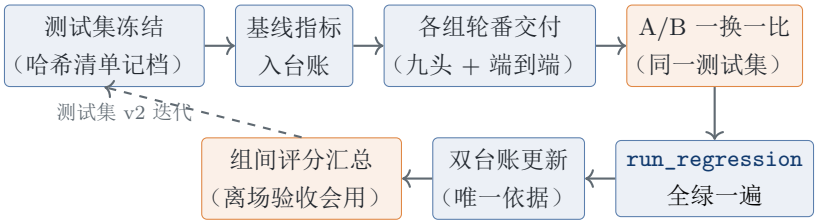
5.3　核心任务

四项职责一张表，随后说人话。

表 5.2　第 5 组四项职责：做什么、交什么、哪门课的血缘

职责	做什么	产出	血缘
测试集构建与冻结	选图（基准五图 + 录制场景抽帧）、生成 SHA-256 哈希清单、按难易分桶	测试集 v1 / v2 + 哈希清单	第 3 讲
基线指标台账	两条路线的初始化 / 稳态 / P95 指标、各组数据量、探针状态逐条入账	双台账	第 3 讲
A/B 与回归评测	`ab_compare.py` 一换一比；`run_regression.ps1` 全绿一遍	A/B 报告、回归日志	第 6 讲
组间评分汇总	从台账汇成评分表，交叉复核后交离场验收会	评分汇总表	全项目

说人话版本：**测试集构建**是出考卷——考卷发下去就不能换题，每张图的“指纹”当场记档；**基线指标台账**是记账——各组每次交付的指标、数据量、探针状态逐行入账（数据量台账记第 4 组的产出，基线指标台账记各组的模型成绩），这本账是“节点验收与成绩评定唯一依据”；**A/B 与回归评测**是验货——A/B 是“只换一个模型、其余全不动”的控制变量对比，回归是“改完之后老功能还在不在”的例行体检；**组间评分汇总**是判卷——从台账汇总，不凭印象。闭环一圈一圈转，见下图。



裁判的两件兵器：哈希与分桶

SHA-256 哈希：给文件内容算一个 256 位的“指纹”，改一个字节就面目全非——冻结测试集 = 给每张图记指纹，谁换了图一查便知。PowerShell 一条命令：`Get-FileHash 文件名 -Algorithm SHA256`。**难度分桶**：测试样本按“易 / 中 / 难”分桶再报成绩——只报总平均会掩盖“人人都会”与“全军覆没”的差别；分桶之后，谁在难桶上掉分一目了然。

项目连线：裁判的每件兵器都来自第一篇

冻结测试集是第 3 讲“验证集”的工程化——训练组在验证集上防过拟合，你在冻结测试集上防“评测漂移”，方法论同一套；A/B 是控制变量实验，第 6 讲“一次只动一个因子”的对照思路直接搬用；台账是第 3 讲评估表格的加长版——把“准确率”换成“稳态毫秒”与“重投影误差”。你不需要发明新方法，只需把第一篇的方法用到全项目尺度。

5.4　第 19 周周五提交清单

提交线：2027年 1 月 22 日（第 19 周周五）完全提交离场。你的组交齐下表五样——离场验收会上，评委手里翻的就是前三样。

表 5.3　第 5 组提交清单（2027 年 1 月 22 日）

交付物	格式	说明
双台账终版	xlsx / CSV	数据量台账 + 基线指标台账，各组每次交付一条，可回溯到原始日志
A/B 报告归档	zip	全部 before/after JSON 与对比表，按日期与组号编目
组间评分汇总表	PDF + xlsx	离场验收会用；每一个分数都能回溯到台账条目
测试集与哈希清单	目录 + 清单文件	v1 冻结版 + v2 困难样本版；清单含每张图的 SHA-256 与分桶标注
组报告与 AI 声明	PDF 8–15 页 + 一页	按第 三 章模板；AI 使用声明逐条列清

5.5　AI 分工与人工必做

分工原则一句话：AI出草稿，人出签字——签字的人对数字负全责。

表 5.4　第 5 组 AI 分工与人工必做

AI可以自动做	人必须亲自做
解读 ab_compare.py 与 run_regression.ps1，起草评测指标说明书	本机部署两条路线，亲手点亮两个探针
写哈希清单与分桶统计的脚本草稿	亲手冻结测试集，抽三张复核哈希一致
起草 A/B 报告与评分表的表格框架	逐条核对 AI 结论（对着代码与日志才算数）
解释报错、给排查建议	台账录入与评分复核；在账本上签字担责

求助纪律：组内 → 助教（研究生）→ 导师，求助必须附完整命令与完整输出。课程成绩建议构成：实验一不计分，实验二至四各 10 分共 30 分，平时作业 20 分（十二讲章后习题十二选二），项目成果 50 分，总评满分 100 分（教师可调）；无论怎么调，每次节点验收都以双台账为唯一依据——包括你们自己组。

5.6　防糊弄指南与满分标准

防人工糊弄指南（第 5 组版）

(1) 改台账一个数 = 改全项目成绩：台账是“节点验收与成绩评定唯一依据”，笔下每个数字都要能与原始日志对上；**虚构或美化指标 = 该节点成绩作废**，裁判组作假加倍严肃。**(2) 测试集不冻结就开测**：今天这五张、明天那五张，A/B 结论全作废——哈希清单是防“测试集漂移”的唯一手段，先冻结、后评测。**(3) 评分凭印象**：每个分数必须能回溯到一条台账记录。**(4) 回归没全绿就签字**：run_regression.ps1 全绿一遍才算数，红一项就如实记红并写明原因。

防 AI 糊弄指南（第 5 组版）

(1) AI 背的“标准评测指标”不算数：本项目测什么，以 ab_compare.py 的指标表与引擎日志为准。**(2) AI 口算的哈希一律不信**：哈希、均值、P95 必须由你本机命令生成并复核。**(3) AI 说“回归应该能全绿”不等于全绿**：只认你终端里的完整输出与退出码。**(4)** AI 起草的评分汇总表，每一格都要人对着台账核一遍再交——AI 录错一格，五个组的成绩跟着错。

成果满分 50 分，怎么拿每一分见下表。

表 5.5　第 5 组成果满分标准（50 分制）

分值	项目	怎么拿满
20	功能达成	实验二两条路线部署验收双双通过（8 分）；实验三测试集 v1 冻结 + 基线入账 + 回归全绿（7 分）；实验四台账 v2 全覆盖 + 困难样本测试集 v2（5 分）
10	证据完整	四次实验证据包齐全：哈希清单、部署与探针日志、全绿回归日志、A/B 归档；缺一件扣 5 分
8	报告质量	组报告按模板 8–15 页；评测指标说明书讲得清，评分表可回溯、有交叉复核记录
7	协作与台账	排班有记录；各组交付登记无缺失、无迟到；评分汇总按时产出并用于离场验收会
5	加分	如：设计“易 / 中 / 难”分桶并给出各组在难桶上的差距分析；为回归脚本补充新检查项且被采纳

5.7　四次实验安排

四个节点与全班同频：实验一“大模型 agent 模式配置及应用”（2026 年 10 月 12 日）、实验二“匹配合适程序并执行本地部署”（11 月 2 日）、实验三“深度网络的模型训练实践”（11 月 23 日）、实验四“模型优化训练实践”（12 月 21 日）。每场按“目标—步骤—验收证据—常见坑”执行：步骤照抄就能跑，证据一栏就是成绩。

表 5.6　第 5 组四次实验总览

节点	日期	本组动作	核心验收证据
实验一	10月 12 日	agent 解读两个评测工具，写指标说明书	说明书 + 核对记录一页
实验二	11 月 2 日	两条路线部署 + 双探针点亮	`PASS (12/12)` 与 `PASS (5/5)`
实验三	11 月 23 日	测试集 v1 冻结、基线入账、回归全绿	哈希清单 + 台账 + 全绿日志
实验四	12 月 21 日	台账 v2 + 困难样本测试集 v2	台账 v2 + v2 哈希清单

5.7.1　实验一（10 月 12 日）：让 agent 替你读评测工具，你来写说明书

目标：用大模型 agent 解读 `tools/ab_compare.py` 对比哪些指标、`tools/run_regression.ps1` 回归哪些环节，产出全组第一份武器：一页“评测指标说明书”。

步骤：(1) 装好编程助手并切到 agent 模式；(2) 让 AI 回答 `ab_compare.py` 三问：读的是哪两份文件、逐项列出对比的指标（初始化耗时、稳态均值、稳态 P95、输出宽高等）与各自“越高越好还是越低越好”、为什么强调“一次只换一个模型”；(3) 让 AI 拆解 `run_regression.ps1` 的环节链——配置、编译、单元测试、基准拼接、遥测——并回答“每一环失败会发生什么”；(4) 逐条与代码对照核对并至少纠正一处，成文一页“评测指标说明书”：每项指标的中文名、方向、进台账的格式。

验收证据：一页评测指标说明书 + “AI 解读 + 人工核对”记录。

常见坑：AI 把指标张冠李戴——让它逐项对代码里的指标列表报行号；把初始化耗时混进每帧耗时——加载只发生一次，稳态才是每帧的账；说明书只写指标名不写方向，两周后连自己都看不懂“这个数大了是好是坏”。

5.7.2　实验二（11 月 2 日）：两条路线都部署，两种探针都点亮

目标：裁判先当一次被评者——两条路线都在你本机部署并通过探针。以后各组交付，你验收的手感就来自今天。

步骤：(1)路线一：VS2022 打开 `stitcher_cpp/build/360stitcher.sln`，切 `Release` 整体生成；命令行进入 `build/Release` 运行 `360stitcher.exe --benchmark-stitch=deploy/simulation --provider=onnx`，日志应出现九行 `[Provider]` 加载记录；再运行 `python tools/nine_head_probe.py`，期望 `PASS (12/12)`。(2) 路线二：进入 `e2e_gpu`，运行 `bin/360stitcher_e2e.exe --benchmark-stitch=bin/simulation --engine=ropstitch`，日志应出现 `[Engine] loaded` 与 `engine active`；再运行 `python training/engine_probe_test.py`，期望 `PASS (5/5)`。

验收证据：两段部署日志 + 两个探针的完整输出（12/12 与 5/5），四样齐全。

常见坑：路线一不在 `build/Release` 目录下运行，程序找不到基准图；路线二的引擎探针会真实替换模型做实验——跑探针时别开着 `360stitcher_e2e.exe`，文件占用会让“交付还原”失败；两个探

针一个 12/12 一个 5/5，别背混——裁判先把自己手里的验收线背熟。

5.7.3　实验三（11 月 23 日）：冻结测试集 v1，基线入账，回归全绿

目标：“数据生产与质量验收分离”落地日——考卷 v1 提前冻结：基准五图 + 第 4 组公开仿真场景（UDIS-D 采样，开局即有）先行定版，录制场景到手后作为 v1.1 增补；基线记账、全项目第一轮回归跑绿。

步骤：(1) 选定测试集 v1：`deploy/simulation` 基准五图 + 首批录制场景抽帧，逐张生成 SHA-256 哈希清单并归档；(2) 在冻结测试集上跑两条路线基准，把初始化耗时、稳态均值、稳态 P95 记入台账基线栏；(3) 运行 `powershell -File tools/run_regression.ps1` 全绿一遍，留完整日志；(4) 台账登记本节点各组状态一览，向全班公示。

验收证据：测试集 v1 哈希清单、基线指标台账条目、`run_regression.ps1` 全绿日志，三样缺一不可。

常见坑：哈希没写，测试集就会漂移——两周后有人换了图你自己都发现不了，A/B 前后不可比；`run_regression.ps1` 的权限问题是新手第一坑——PowerShell 执行策略拦脚本是常态，按报错调整策略或用 `powershell -File` 方式运行，别管理员权限硬怼；把原始视频直接当测试集——先抽帧、写清单、再冻结。

5.7.4　实验四（12 月 21 日）：台账 v2 与困难样本，裁判的终场哨

目标：九头三组（第 1–3 组）与端到端组（第 7 组）全部交付完毕，台账升到 v2；再从录制素材里挑出“容易翻车”的样本，出第二版考卷。

步骤：(1) 逐行核对台账 v2：每个交付组每次交付的指标、A/B 结论、探针状态，查漏补缺；(2) 从录制场景人工挑困难样本（弱纹理、运动模糊、重叠区小、光照突变），按易 / 中 / 难分桶标注，冻结为测试集 v2 并写哈希清单；(3) 通知各组在 v2 上补测“难桶”成绩，入台账；(4) 汇总组间评分表初稿，组内交叉复核后定稿。

验收证据：台账 v2 全表、测试集 v2 哈希清单、各组难桶补测记录、评分汇总表定稿。

常见坑：困难样本只挑最难的、不挑有代表性的——分桶的意义在拉开区分度，全是地狱难度谁都零分；评分先斩后奏——先定评分规则并公示，再录分数；台账 v2 与 A/B 归档对不上号——每条指标都要能翻出对应的 before/after JSON。

第 6 组　六目拼接 · 端到端数据组

本章地图

讲什么：第 6 组任务书——端到端路线吃什么数据、32+8 对合成数据怎么一键生成、首批真实数据从组 4 的录制里怎么切出来、`build_dataset.py` 与 `benchmark.py` 两个工具怎么交付。**为什么值得读**：端到端模型的上限，从你交出数据集那天起就定了——第 7 组训得再努力，吃的全是你做的饭。**学完你能**：跑通“合成数据演练 → 真实数据实跑”的切换，写一份数据集卡片，交付能对任意 onnx 引擎算重投影误差与耗时的 benchmark 工具。

一句话概括你们组：九头路线吃“统计量 + 标签”（第 4 组把遥测变成 CSV 行），端到端路线吃**原始图像对 + 对齐变换标签**——后一碗饭归你做。合数据先“干跑”，真数据再“实跑”，切换的操作手就是你。

6.1　组档案

表 6.1　第 6 组档案

项目	六目拼接（项目一，360Stitcher实时全景系统，详见第 一 章）
组号与组名	第 6 组 • 端到端数据组
人数与投入	5 人；每周合计 6–8 小时（组内自行排班，建议每周固定 2 次）
负责的产出	`build_dataset.py`（一键训练/验证清单）、`benchmark.py`（重投影误差与耗时 → `benchmark_report.json`）、合成 + 真实两代数据集与数据集卡片
上游依赖	开局即可构建：`make_demo_data.py` 合成对 + UDIS-D 公开拼接对（实验三的主粮）；组 4 组真机录制抽帧实验四接入
下游依赖	第 7 组吃你的 train/val 清单去训练（公开版 v1 实验三交付）；第 8 组吃你的 `benchmark.py` 测耗时与精度
工作目录	`e2e_gpu/training/group10_e2e_data`（数据装载位 `e2e_gpu/training/datasets/`）
硬件要求	本机无 GPU 亦可：造数据与算指标不吃 GPU，CPU 足够

目录名里的 `group10` 是源工程里端到端数据组的原始编号，与本课程“第 6 组”无关。两个提醒先立在此：其一，`datasets/` 的目录格式**写死**（按批次建夹，`left` 与 `right` 里**同名文件是一对**），以 `datasets/README.md` 为准；其二，读写图一律走 `io_cn.py` 的 `imread_cn`/`imwrite_cn`——OpenCV 原生函数遇到中文路径会静默失败，前人踩过的坑。

6.2　项目背景与你的位置

第 一 章讲过两条路线的赛马：九头路线是“系统老手带九个小学徒”，端到端路线是“一个学徒吃整对图像”——一次吃一对 512×512 的相邻相机画面，直接吐出把右图对齐到左图的 3×3 变

换矩阵。六目每采一帧可切 5 对相邻对（cam1+2、cam2+3、……、cam5+6），你的数据量单位是“对数”。你在流水线里的位置：端到端路线的**最上游**——合数据阶段把演练流程跑顺，真数据阶段从第 4 组的录制素材里切出训练对；“干跑到实跑”那一脚油门，由你踩。

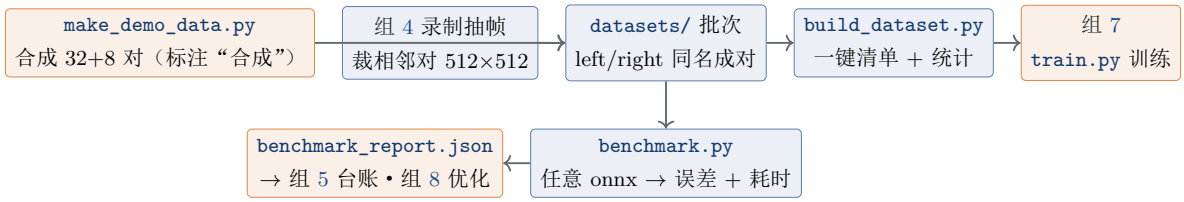
6.3　核心任务

两件工具、两代数据，一张表说清。

表 6.2　第 6 组核心交付：工具两件、数据两代

交付	要做到什么	谁在用
`build_dataset.py`	扫描 `datasets/` 全部批次，一键生成训练/验证清单与统计（对数、场景、划分比例）	第 7 组
`benchmark.py`	对任意引擎 onnx：读测试图对，逐对推理，输出重投影误差与耗时，写 `benchmark_report.json`	第 8 组、组 5
合成数据批次	`make_demo_data.py` 生成 32+8 对（train 32 对、val 8 对，含已知平移真值），全程标注“合成”	全流程演练
真实数据集 v1 → v2	首批 ≥ 50 对（来自组 4 录制抽帧的相邻对裁剪），实验四扩量并更新卡片	第 7 组

端到端数据的“标签”值得多嚼两口：合成对造的时候平移量是自己定的，真值天然已知；真实对没有现成答案，用传统拼接管线算出的变换当**伪真值**，再人工抽检兜底——与第 4 组“用系统遥测当标签”是同一种思想。模型学得好不好，用**重投影误差**来量（见下方知识盒）。整条流水线如下图：上游两个水龙头（合成、真实），下游两个用户（训练、评测）。



重投影误差：对齐准不准的量尺

把 right 图按模型预测的变换“拉”回 left 图的坐标系，公共内容应完全重合；逐对对应点算偏移再取平均，就是**重投影误差**（单位：像素），越小对齐越准。好比印刷的“套印误差”：红蓝两色没套准，边缘就发虚——误差就是那道虚边的宽度。第 8 组判断“量化后精度损没损”、你们判断“模型逐版有没有进步”，用的都是这把尺。

零基础扫盲：数据集卡片四要素

发布数据集要附一页“卡片”，四要素缺一不可：**来源**——谁、什么设备、什么场景，哪些是合成；**划分**——train/val怎么切、按什么原则（按场景切，不按帧切，防止“同一场景一半在训练一半在考试”）；**统计**——对数、场景数、分辨率、重叠率分布；**许可**——素材能用到什么范围（本课程项目内部使用）。

项目连线：你的数据喂的是哪种网络

端到端模型的输入是 512×512 图像对——第 5 讲卷积怎么“看”图，它就在看什么；第 4 讲“数据自己会说话”的思想，在这里变成“让图像对直接喂出变换”。扩量想加旋转、亮度扰动，第 6 讲的数据增强九宫格就是菜单。而“合成干跑、真实实跑”两步走，是机器人与自动驾驶领域的标准做法：仿真里先把流程磨顺，再上真数据——你这一学期走的正是这条路。

下图是“对齐变换”的直观样子：同一个平面在两张相邻视角照片里的四角，被一个 3×3 矩阵一一对应地搬过去——你的每对训练样本，学的就是这么一个搬运关系。

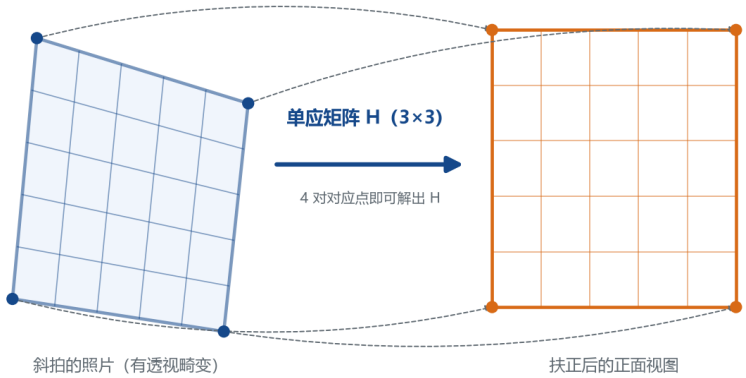


图 6.1　单应变换直观图：平面 A 的四角经一个 3×3 变换映射到平面 B——端到端模型每对输入要学的“搬运关系”，也是 `benchmark.py` 用重投影误差去量的对象

6.4　第 19 周周五提交清单

提交线：2027年 1 月 22 日（第 19 周周五）完全提交离场。你的组交齐下表五样。

表 6.3　第 6 组提交清单（2027 年 1 月 22 日）

交付物	格式	说明
`build_dataset.py` + `benchmark.py`	py	一键清单生成；对任意 onnx 输出 `benchmark_report.json`，README 写明命令与期望输出
端到端数据集 v2	datasets 目录 + 清单	合成 32+8 对（标注“合成”）+ 真实 ≥ 150 对、≥ 3 场景
数据集卡片	一页	来源 / 划分 / 统计 / 许可四要素，v1 与 v2 各一版或合并说明
benchmark 报告	json + PDF 2–4 页	`ropstitch.onnx` v1/v2 的重投影误差与耗时，均值与 P95 并列
组报告与 AI 声明	PDF 8–15 页 + 一页	按第 三 章模板；AI 使用声明逐条列清

6.5　AI 分工与人工必做

分工原则一句话：AI出草稿，人出签字——签字的人对数字负全责。

表 6.4　第 6 组 AI 分工与人工必做

AI可以自动做	人必须亲自做
解读 `datasets/README.md` 契约与 `make_demo_data.py` 逻辑	本机跑通合成、推理自测、清单、benchmark 全部命令
起草裁对脚本、统计图与数据集卡片的框架	亲手抽检真实对：同名成对、重叠区没被裁丢
起草两路线数据差异说明的表格框架	抽 10 对可视化逐对过目；在卡片与报告上签字担责
解释报错、给排查建议	与组 7/8 的交接当面确认，不靠转述

求助纪律：组内 → 助教（研究生）→ 导师，求助必须附完整命令与完整输出。课程成绩建议构成：实验一不计分，实验二至四各 10 分共 30 分，平时作业 20 分（十二讲章后习题十二选二），项目成果 50 分，总评满分 100 分（教师可调）；每次节点验收以组 5 维护的双台账为唯一依据——你的数据量，就是台账里记的那一行。

6.6　防糊弄指南与满分标准

防人工糊弄指南（第 6 组版）

(1) 合成数据忘了标“合成”：`demo` 批次是演练用的，混进真实数据台账就是数据事故——批次名、卡片、清单三处同标。**(2) 录屏当相机序列：**数据契约明文禁止——录屏里只有内嵌的 6 路小画面是真实相机画面。**(3) 对数凑数：**同一对图复制改名、同场景连抽五十张近似帧，台账查同源重复一查一个准；**虚构数据集统计 = 该节点成绩作废。****(4) 考卷泄题：**`benchmark.py` 只准用 val 批次，训练集混进评测集，一切指标作废。

防 AI 糊弄指南（第 6 组版）

(1) AI 数对数会数错：对数、场景数、划分统计，一律以 `build_dataset.py` 的输出为准。**(2) AI 写的裁对脚本必须本机验证：**中文路径会不会静默失败、中心裁方会不会把重叠区裁丢，跑过才知道。**(3) AI 说“benchmark 应该能出数”不等于出数：**`benchmark_report.json` 落盘且字段齐全才算。**(4)** AI 起草的数据集卡片，来源与统计两栏逐条对真实目录核过再交。

成果满分 50 分，怎么拿每一分见下表。

表 6.5　第 6 组成果满分标准（50 分制）

分值	项目	怎么拿满
20	功能达成	实验二合成 32+8 对全流程跑通（5 分）；实验三真实 ≥ 50 对 + 数据集卡片合格（8 分）；实验四扩量 ≥ 150 对且 `benchmark.py` 对 `ropstitch.onnx` 出数（7 分）
10	证据完整	四次实验证据包齐全：生成日志、推理自测可视化、清单统计、`benchmark_report.json`；缺一件扣 5 分
8	报告质量	组报告按模板 8–15 页；卡片四要素齐全，benchmark 报告均值与 P95 并列、结论有数据
7	协作与台账	向组 7/8 交接有记录；数据量按时入组 5 双台账；求助记录规范
5	加分	如：用数据增强扩量并用 A/B 证明不伤指标；产出“困难样本建议清单”交组 5 进难桶

6.7　四次实验安排

四个节点与全班同频：实验一“大模型 agent 模式配置及应用”（2026 年 10 月 12 日）、实验二“匹配合适程序并执行本地部署”（11 月 2 日）、实验三“深度网络的模型训练实践”（11 月 23 日）、实验四“模型优化训练实践”（12 月 21 日）。每场按“目标—步骤—验收证据—常见坑”执行。

表 6.6　第 6 组四次实验总览

节点	日期	本组动作	核心验收证据
实验一	10月 12 日	agent 解读数据需求，写两路线差异说明	差异说明 + 核对记录一页
实验二	11 月 2 日	合成数据全流程：生成 + 推理自测	32+8 对 + `infer_overlay.jpg`
实验三	11 月 23 日	首批真实数据 ≥ 50 对 + 卡片	清单统计 + 抽检可视化
实验四	12 月 21 日	扩量 + `benchmark.py` 初版	`benchmark_report.json`

6.7.1　实验一（10 月 12 日）：读懂端到端的数据餐单，写两路线差异说明

目标：用大模型 agent 解读端到端路线的训练数据需求——相邻相机对的图像对 + 对齐变换标签——并与九头路线的数据做对照，产出一页差异说明。

步骤：(1) 装好编程助手并切到 agent 模式；(2) 让 AI 读 `datasets/README.md` 与 `make_demo_data.py`，回答三问：一个训练样本长什么样（left/right 同名成对、512×512）、标签是什么（合成对的已知平移真值 / 真实对的伪真值变换）、训练与验证怎么分；(3) 让 AI 对照第 4 组的遥测标签，列“样本形态 / 标签来源 / 数据量单位 / 质量瓶颈”四行对照表；(4) 逐条与文件核对并至少纠正一处，成文一页。

验收证据：一页“两路线数据差异说明”+ AI 解读与人工核对记录。

常见坑：AI 说“端到端不需要标签”——合成对的平移真值是造数据时写死的，真实对也要伪真值标签，“不要标签”是它背错了课文；AI 把 left/right 讲成“左相机右相机”——口径是**对内前一路 / 后一路**；对照表写“数据不一样”级别的空话——每个格子都要落到具体文件与字段。

6.7.2　实验二（11 月 2 日）：合成数据全流程，32+8 对先干跑一遍

目标：真数据到位前，用合成对把“生成 → 训练口径 → 推理自测”整条流程跑顺——流程里的坑今天踩完，实验三就只剩数据问题。

步骤：(1)进入 `e2e_gpu/training`，`pip install -r requirements-cpu.txt`；(2) 运行 `python make_demo_data.py`，产出 `datasets/train/demo`（32 对）与 `datasets/val/demo`（8 对），含已知平移真值；(3) 运行 `python infer_test.py`（缺省用 val 第一对），到 `outputs/` 查看 `infer_overlay.jpg` 的叠加可视化，确认对齐合理；(4) 加餐：`python make_real_dataset.py` 生成 8 对真实测试批次（只放 val，评测用、不进训练）。

验收证据：32+8 对生成日志与目录清单、`infer_overlay.jpg`、真实测试批次的说明一句（来源与用途），全进证据包。

常见坑：清单或脚本路径含中文导致脚本崩——读写图一律用 `io_cn.py` 的 `imread_cn`/`imwrite_cn`；合成批次忘了标“合成”——批次名、清单、卡片三处同标；`infer_test.py` 报形状错误——模型输入是五维 `[1,2,3,512,512]`，脚本注释里有前车之鉴。

6.7.3　实验三（11 月 23 日）：首批公开数据集（UDIS-D），50 对起步

目标：从 UDIS-D 公开拼接对里切出首批训练数据——**≥ 50 对、覆盖 ≥ 2 个场景**——并发布数据集卡片（如实标注“公开数据版”）；真机录制抽帧实验四联调时增补为真实版。

步骤：(1) 取组 4 组录制视频的抽帧，六路同帧按相邻对切分（cam1+2、……、cam5+6，一帧切 5 对）；(2) 中心裁方 + 缩放至 512×512（对齐引擎输入口径），先算重叠区再下剪刀；(3) 按 `datasets/train/批名/left|right` 放置，**同名文件成对**；(4) 运行 `build_dataset.py` 出清单与统计，核对对数达标；(5) 写数据集卡片，抽 10 对出可视化逐对过目，交接组 7 。

验收证据：≥ 50 对清单与统计、数据集卡片一页、10 对抽检可视化、交接记录。

常见坑：left/right 同名不成对——文件名对不上一个字符，`train.py` 读出来就是张冠李戴的错配对；中心裁方把重叠区裁没了——重叠是标签的载体，裁丢等于白切；只在同一场景抽——50 对全长一个样，多样性不够，训练一上真场景就露馅。

6.7.4　实验四（12 月 21 日）：数据扩量与 benchmark 初版，双线收官

目标：真实数据集扩量到**≥ 150对、≥ 3 个场景**并更新卡片；交付 `benchmark.py` 初版——对当前 `ropstitch.onnx` 输出重投影误差与耗时。

步骤：(1) 扩量：补切批次或补场景（用第 6 讲的增强手段时，增强对单独标注来源），更新清单与卡片至 v2；(2) 写 `benchmark.py`：输入测试图对目录 + 任意 onnx 路径，逐对推理、算重投影误差与耗时，写 `benchmark_report.json`（含逐对明细与汇总均值/P95）；(3) 对当前 `ropstitch.onnx` 跑一遍出报告；(4) 结论登记组 5 台账，交接组 8 作优化基线。

验收证据：扩量统计与 v2 卡片、`benchmark_report.json`、`benchmark.py` 的 README（命令与期望输出）。

常见坑：benchmark 只报均值不报分布——P95 与最差对也要有，优化组的提升空间全藏在尾部；评测批次混入训练批次——考卷泄题，指标再好也作废；onnx 输入尺寸写死 512——从模型读输入形状或做成参数，下一版模型一换就崩。

第 7 组　六目拼接 · 端到端训练组

本章地图

讲什么：第 7 组任务书——端到端模型的结构与损失怎么读、CPU 冒烟怎么做、全书唯一一次 200 epoch 真训练怎么在 RTX 4080 上完成、v1/v2 两版怎么逐版交付。**为什么值得读**：你是唯一把“深度网络的模型训练实践”四个字全部走到实处的组——真 GPU、真 200 epoch、真交付上系统。**学完你能**：独立跑通“训练 → 导出 onnx → 一键交付 → 探针验收”全链路，管好自己的 GPU 时段，用 A/B 数据证明每一版都在进步。

一句话概括你们组：第 6 组把饭做好，你们组**吃饭长个子**——把端到端模型从随机初始化一路训到能上岗，交付给系统替换旧大脑。全书唯一需要 GPU 的组，唯一的 200 epoch，都在你手里。

7.1　组档案

表 7.1　第 7 组档案

项目	六目拼接（项目一，360Stitcher实时全景系统，详见第 一 章）
组号与组名	第 7 组 • 端到端训练组
人数与投入	5 人；每周合计 6–8 小时（组内自行排班；实验三按 GPU 时段另排）
负责的产出	`ropstitch.onnx` v1/v2 + `ropstitch.metrics.json`、两次真训练完整日志、A/B 对比报告
上游依赖	开局即可训练：组 6 组的合成对与 UDIS-D 公开数据集（实验三的主粮）；真机数据实验四续训；组 5 冻结测试集是 A/B 考场
下游依赖	第 8 组拿你的 onnx 做推理优化；每次交付入组 5 双台账
工作目录	`e2e_gpu/training/`（本组任务夹 `group11_e2e_train`；`train.py`、`export_onnx.py`、`deliver_model.bat` 在 `training/` 根目录）
硬件要求	**RTX 4080 机器**（训练用）；交付与验证走 CPU 推理，回本机即可

目录名里的 `group11` 是源工程里端到端训练组的原始编号，与本课程“第 7 组”无关。先立一条**特别纪律**：4080 机器只归第 7 组（第 8 组可申请时段）与导师使用——用前预约登记，用时守时段，训练完立刻把 `exports/` 与日志拷回本机。模型交付的形状契约别背错：输入 `[B,2,3,512,512]`（一对 512×512 图）、输出 `[B,3,3]`（一个变换矩阵）、opset 13——`export_onnx.py` 会按契约自检。

7.2　项目背景与你的位置

第 一 章讲过端到端路线的野心：不再让九个小参数头逐个学经验参数，而是一个网络直接看一对相邻画面、吐出把右图对齐到左图的 3×3 变换；六目每帧拆 5 个相邻对逐对推理，链式累积后由渲染管线合成全景。你在流水线里的位置：端到端路线的**发动机**——第 6 组的数据喂进来，你把模型训出来、导成 onnx、经 `deliver_model.bat` 交付进 `bin/models/`，再由引擎探针证明“系统真的在用你

的模型”。你的每一个版本都会被组 5 的台账与 A/B 记录在案：**重投影误差逐版不升**，是你给自己的军令状。

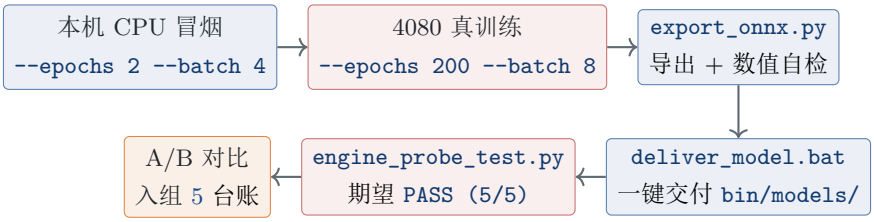
7.3　核心任务

两个版本、一条流水线，一张表加一张图说清。

表 7.2　第 7 组核心交付：两个版本一台流水线

版本	要做到什么	数据
冒烟版	`python train.py --epochs 2 --batch 4` 本机 CPU 跑通全链路，验证代码、数据、交付、探针四处无堵点	合成 32+8 对
v1	4080 上 `--epochs 200 --batch 8` 真训练，导出交付，A/B 证明优于基线（公开数据版）	组 6 UDIS-D 公开数据集 ≥ 50 对
v2	真机数据微调（从 v1 续训、调小学习率；合并联调收官），A/B 证明**逐版不升**，不升则如如实回退 v1	组 6 真机数据（联调日接入）

整条流水线如下图：本机冒烟排除低级错误，4080 出真模型，导出自检守契约，一键交付换大脑，探针验收防绕过，A/B 入账定成绩——六步串成一圈，每转一圈出一个新版本。



训练三旋钮与一个保险：epoch / batch / 学习率 + 断点续训

epoch：全部训练数据过一遍网络算一个 epoch；**batch**：一次一起送进网络的样本数；这两个数加**学习率**（每步挪多大），就是第 3 讲的三大超参数：epoch 太少没学饱、太多会过拟合，batch 与学习率互相牵制。200 epoch 是长跑，中途断电、超时都可能有——`training/checkpoints/` 里的 `.pth` 就是保险：从上次存档续训，不用从头再来。每次训练的超参数与最终 `val_loss` 记成表格，这是你实验报告的骨架。

网络骨干带着**残差连接**——第 7 讲“让信息走捷径”的短路设计（下图），梯度可经捷径直通浅层，深网络才训得动；损失函数是**重投影误差**：用预测变换把 right 拉回 left，逐像素比对偏移。结构细节以 `train.py` 代码为准，实验一你要亲手把它读出来。

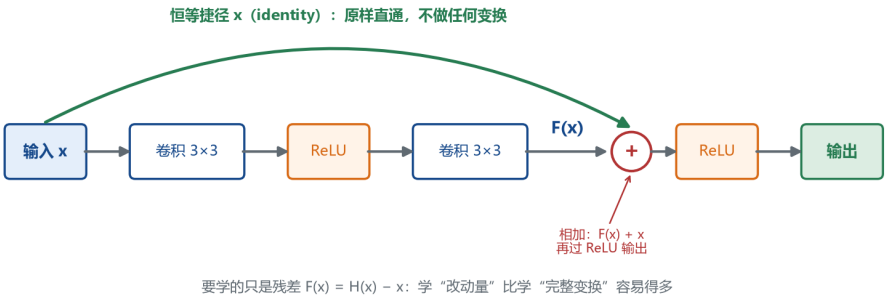


图 7.1　残差块：输入经两层卷积得到 $F(x)$，再与恒等捷径相加——你模型骨干里的“信息捷径”，第 7 讲的知识直接上岗

项目连线：200 epoch 把第一篇串成一条线

`train.py` 里的训练循环，就是第 6 讲那张“取 batch → 前向 → 算 loss → 反向 → 更新”的流程环原样运转；loss 降不动、会过拟合，按第 3 讲的曲线方法论盯着验证集说话；骨干里的残差连接出自第 7 讲；导出 onnx 后与 PyTorch 输出的数值一致性自检（误差 $< 10^{-3}$），是第 0 讲“浮点数不完全相等”的实战回响。一门课的知识在这一章拧成一股绳。

7.4　第 19 周周五提交清单

提交线：2027年 1 月 22 日（第 19 周周五）完全提交离场。你的组交齐下表五样。

表 7.3　第 7 组提交清单（2027 年 1 月 22 日）

交付物	格式	说明
`ropstitch.onnx` v2 + `ropstitch.metrics.json`	onnx + json	经 `deliver_model.bat` 交付、`bin/models/` 在位；交付后探针仍 `PASS (5/5)`
训练证据包	zip	冒烟、v1（200 epoch）、v2（微调）三段完整日志，含运行者与起止时间；超参数记录表
A/B 对比报告	PDF，2–4 页	基线 / v1 / v2 同测试集重投影误差对比，“逐版不升”或如实回退的结论
数据流图与训练记录	一页图 + 表	实验一“图像对 → 变换估计”数据流图 + 各版本超参数与 `val_loss` 台账
组报告与 AI 声明	PDF 8–15 页 + 一页	按第 三 章模板；AI 使用声明逐条列清

7.5　AI 分工与人工必做

分工原则一句话：AI出草稿，人出签字——签字的人对数字负全责。

表 7.4　第 7 组 AI 分工与人工必做

AI可以自动做	人必须亲自做
解读 `train.py` 结构与损失函数，起草数据流图	本机跑通冒烟、导出、交付、探针全部命令
解读报错、给续训与调参建议	4080 上亲手启动训练、盯 loss 曲线、守时段
画 loss 曲线、整理超参数表格的脚本	亲手核对 `metrics.json` 每个字段与日志对账
起草 A/B 报告文字段落	在训练证据包上签字，对“本人运行”担责

求助纪律：组内 → 助教（研究生）→ 导师，求助附完整命令与完整输出。课程成绩建议构成：实验一不计分，实验二至四各 10 分共 30 分，平时作业 20 分（十二讲章后习题十二选二），项目成果 50 分，总评满分 100 分（教师可调）；每次节点验收以组 5 双台账为唯一依据——你的 `metrics.json`，就是台账里记的那一行。

7.6　防糊弄指南与满分标准

防人工糊弄指南（第 7 组版）

(1) 铁律原文，一字不改：交付后程序输出的相机位置必须随模型内容变化（探针原理），训练指标必须来自本人运行——探针会放大模型权重再跑一遍，输出不变就说明系统没用你的模型；日志没有你的账号与时间，就不算你训的。**(2)** 冒烟指标当真训练成绩报——`epochs_done: 2` 的 `metrics.json` 是流程验证，不是模型成绩。**(3)** v2 不如 v1 就悄悄只交 v1——如实条款：回退 v1 并写明原因同样是合格结论，**选择性上报按虚构处理**。**(4)** GPU 时段超时霸机、替组 8 代跑占用——时段登记白纸黑字，违规记录进台账。

防 AI 糊弄指南（第 7 组版）

(1) AI 报的 loss 与 `val_loss` 一律不信：只有你终端里滚出来的那份是真的，AI 现编的曲线一张都不许进报告。**(2) AI 讲网络结构爱“脑补”：**几层卷积、哪里有残差、损失怎么算，以 `train.py` 代码为准，让它逐行引用行号。**(3) AI 说“这个超参数肯定更好”不等于更好：**调参结论只认同测试集 A/B 数据。**(4)** AI 生成的训练曲线必须能从真实日志一条命令重画——画不出原始日志的那条，就是装饰品。

成果满分 50 分，怎么拿每一分见下表。

表 7.5　第 7 组成果满分标准（50 分制）

分值	项目	怎么拿满
20	功能达成	实验二 CPU 冒烟全链路 + 探针 `PASS (5/5)`（5 分）；实验三 4080 真训练交付 v1 且 A/B 优于基线（10 分）；实验四 v2 微调完成并给出对比结论或如实回退（5 分）
10	证据完整	三段训练日志、`metrics.json`、交付后探针 PASS、A/B 报告，缺一件扣 5 分
8	报告质量	组报告按模板 8–15 页；训练曲线、超参数表、A/B 表齐全，结论有数据支撑
7	协作与台账	GPU 时段预约与使用登记无违规；交付按时入组 5 台账；与组 6/8 交接有记录
5	加分	如：学习率 / batch 小实验（2–3 组对比曲线有结论）；对损失函数做改进且 A/B 有证据

7.7　四次实验安排

四个节点与全班同频：实验一“大模型 agent 模式配置及应用”（2026 年 10 月 12 日）、实验二“匹配合适程序并执行本地部署”（11 月 2 日）、实验三“深度网络的模型训练实践”（11 月 23 日）、实验四“模型优化训练实践”（12 月 21 日）。每场按“目标—步骤—验收证据—常见坑”执行。

表 7.6　第 7 组四次实验总览

节点	日期	本组动作	核心验收证据
实验一	10月 12 日	agent 解读结构与损失，画数据流图	解读 + 核对记录 + 数据流图
实验二	11 月 2 日	本机 CPU 冒烟全链路 + 探针	`PASS (5/5)` + 三段日志
实验三	11 月 23 日	4080 真训练 200 epoch，交付 v1 并 A/B	训练日志 + `metrics.json` + A/B
实验四	12 月 21 日	全量真实数据微调 v2，逐版对比	`metrics.json` + v1/v2 对比表

7.7.1　实验一（10 月 12 日）：读懂你将要训练的东西，画出数据流

目标：用大模型 agent 解读端到端模型的网络结构与损失函数（重投影误差），画出“图像对 → 变换估计”的数据流图——先知道要训什么，再谈怎么训。

步骤：(1) 装好编程助手并切到 agent 模式；(2) 让 AI 读 `training/train.py`，回答三问：网络骨架长什么样（几层、卷积核多大、残差连接在哪）、输入输出什么形状、损失函数怎么定义（重投影误差如何逐像素算）；(3) 让 AI 画出“图像对 → 变换估计”数据流：左右图 → 骨干提特征 → 融合 → 回归 3×3 矩阵 → 重投影误差；(4) 逐条与代码对照核对，标“对 / 错 / 存疑”，至少纠正一处。

验收证据：一页“AI 解读 + 人工核对”记录 + 一张数据流图（手绘拍照亦可）。

常见坑：AI 把它讲成“分类网络”——输出是 9 个数的变换矩阵，不是类别得分；损失函数名张口就来——让它引用 `train.py` 行号，代码为准；数据流图漏掉预处理（缩放 512、归一化）——预处理不对，后面全错。

7.7.2　实验二（11 月 2 日）：本机 CPU 冒烟，三十分钟走通全链路

目标：不上 GPU，先把“训练 → 导出 → 交付 → 探针”四个环节在本机 CPU 上全走通——冒烟 5 分钟，救你半天 GPU 时段。

步骤：(1) 进入 `e2e_gpu/training`，`pip install -r requirements-cpu.txt`；(2) `python train.py --epochs 2 --batch 4`——CPU 几分钟，产出 `checkpoints/ropstitch.pth`；(3) `python export_onnx.py`——导出 `exports/ropstitch.onnx`，数值自检误差应远小于 10^{-3}；(4) 双击 `deliver_model.bat`——正本拷入 `bin/models/`，见 `[OK]`；(5) `python engine_probe_test.py`，期望 `RESULT: PASS (5/5)`。

验收证据：冒烟训练日志、导出自检输出、交付 `[OK]` 提示、探针 `PASS (5/5)`，四样齐全（`epochs_done: 2` 会如实写进 `metrics.json`——它就是冒烟的痕迹）。

常见坑：跳过冒烟直接上 4080——环境没装对、数据格式不对，半天时段烧在低级错误上；本机误装 `requirements-gpu.txt`——没有 CUDA 立刻报错，CPU 冒烟装 CPU 版；双击 `deliver_model.bat` 时 `360stitcher_e2e.exe` 还开着——文件占用拷贝失败，关程序再交付。

7.7.3　实验三（11 月 23 日）：4080 真训练，全书唯一的 200 epoch

目标：在 RTX 4080 上完成 `--epochs 200 --batch 8` 真训练，导出交付 v1，用 A/B 证明它优于人工参数基线。

步骤：(1) 按预约时段上 4080（纪律：只归第 7 组、第 8 组可申请时段、导师；用前登记、用后拷回）；(2) 整个 `e2e_gpu` 拷到 4080，`pip install -r requirements-gpu.txt`，`python check_gpu.py` 确

认识别到 GPU；(3) 组 6 的 UDIS-D 公开数据集放入 `datasets/`（按数据集卡片核对），先 `--epochs 2` 跑一遍确认无堵点，随后 `python train.py --epochs 200 --batch 8`——每半小时看一眼 loss，发散立刻停；(4) 训完 `python export_onnx.py`，把 `exports/` 与完整日志拷回本机；(5) 双击 `deliver_model.bat` 交付 v1，重跑探针须 `PASS (5/5)`，再在组 5 冻结测试集上 A/B：基线 vs v1 的重投影误差。

验收证据：200 epoch完整训练日志（含运行者与起止时间）、`ropstitch.metrics.json`、交付后探针 `PASS (5/5)`、基线 vs v1 的 A/B 对比。

常见坑：时段超时——先测单 epoch 耗时再决定本次时段训多少，`checkpoints/` 断点续训是保险不是摆设；中断后从头重训——续训从存档走，从头重来等于白烧电；loss 发散不肯停——学习率太大就早停排查，跑完 200 个 epoch 的废模型更浪费；日志残缺——没有完整日志，训练等于没发生过。

7.7.4　实验四（12 月 21 日）：全量真实数据微调，v2 收官

目标：真机数据到位（八组合并联调），从公开版 v1 续训微调出真实场景版 v2，用同测试集 A/B 回答“真实数据换掉公开数据，模型变好了吗”。

步骤：(1) 从 `checkpoints/` 的 v1 存档续训，`--epochs 50 --batch 8` 量级、学习率调小（微调不是推倒重来）；(2) 训完导出 v2，交付、重跑探针须 `PASS (5/5)`；(3) 同一冻结测试集 A/B：基线 / v1 / v2 三列重投影误差对比表，写不少于 100 字结论；(4) v2 若不如 v1，如实记录并回退交付 v1——“数据多了不一定赢”本身就是有价值的结论；(5) 交付与结论登记组 5 台账，交接组 8 做优化。

验收证据：v2 训练日志与 `ropstitch.metrics.json`、交付后探针 `PASS (5/5)`、三列 A/B 对比表与结论段。

常见坑：微调学习率不调小——大学习率把 v1 学到的冲掉，越微调越差；只挑好看的版本报——三列表里少一列都算选择性上报；新数据没过卡片就开训——先核对组 6 的 v2 卡片与清单，喂错数据训出的模型没有讨论价值。

第 8 组　六目拼接 · 端到端优化组

本章地图

讲什么：第 8 组任务书——一帧 800 ms 的预算怎么分、逐帧计时基线怎么测、组 7 组的模型怎么复现对账、fp32/fp16/int8 三档怎么比、5 对合并一次推理怎么设计。**为什么值得读**：实时系统的“实时”两个字由你负责——单次引擎调用压不进 300 ms，前面所有精度进步都上不了生产线。**学完你能**：独立测出一台机器的推理耗时账本，做一张“速度与精度权衡表”，掌握量化与批处理两条加速正道。

一句话概括你们组：第 7 组把模型训得准，你们组把它跑得**快**——目标只有一条：单次引擎调用 $\leq$ 300 ms（每帧总预算 800 ms），且优化前后全景输出**视觉无差异**。“又快又一样”，四个字，一学期。

8.1　组档案

表 8.1　第 8 组档案

项目	六目拼接（项目一，360Stitcher实时全景系统，详见第 一 章）
组号与组名	第 8 组 • 端到端优化组
人数与投入	5 人；每周合计 6–8 小时（组内自行排班；fp16 档测试需向第 7 组申请 GPU 时段）
负责的产出	逐帧计时基线、性能预算书（含复盘）、fp32/fp16/int8 三档“速度与精度权衡表”、batch 推理方案、（达标才交）量化版 onnx
上游依赖	第 7 组：v1（公开数据版，实验三即交付）/v2（真机版，实验四联调）是你优化的对象；第 6 组：`benchmark.py` 是你的测量仪
下游依赖	量化版（若达标）经 `deliver_model.bat` 交付进系统；结论入组 5 双台账
工作目录	`e2e_gpu/training/group12_e2e_optim`
硬件要求	本机 CPU 为主战场（交付版引擎走 CPU 推理）；GPU 时段向第 7 组申请

目录名里的 `group12` 是源工程里引擎优化组的原始编号，与本课程“第 8 组”无关。口径先对齐：六目每帧把画面拆成 5 个相邻对，现状是逐对调 5 次引擎；**单次引擎调用 $\leq$ 300 ms** 是硬指标，**5 对合并成一次 batch 推理**是达成路径之一。加速有个铁门槛：**不能牺牲人眼可见的质量**——任何优化都要过“视觉无差异”这一关，所以精度测量和速度测量在你这里同等重要。

8.2　项目背景与你的位置

第 一 章给过一句话：实时全景系统每一帧从六路画面进来到全景上屏，总共只有 800 毫秒——采集预处理、引擎推理、渲染缝合、上屏显示，四段各分一块蛋糕。第 7 组的模型再准，单次推理要是慢吞吞，全景就得掉帧，用户看着就是幻灯片。你在流水线里的位置：端到端路线的**最后一公里**——先给整条链路做“耗时体检”（哪段慢、慢多少），再对大头（骨干网络推理）下刀：量化压缩、

批处理合并，刀刀要有数据佐证。第 5 组的台账会记下你的每一项优化：提速多少毫秒、精度劣化千分之几。

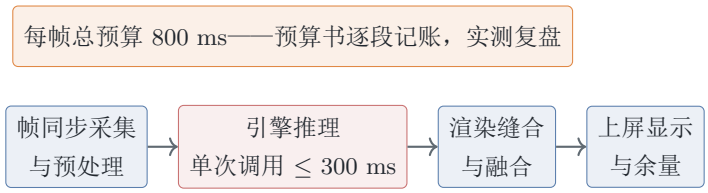
8.3　核心任务

一张预算书、一张基线表、一张权衡表、一份 batch 方案，四件套说清。

表 8.2　第 8 组核心交付：给整条链路做体检，再给推理下刀

交付	要做到什么	什么时候
性能预算书	800 ms总预算的分配表（预处理 / 引擎 ≤ 300 ms / 渲染 / 余量），每项写明测量方法；离场前复盘	实验一立项、离场复盘
逐帧计时基线	引擎模式跑 `--benchmark-stitch`，收集 `[BenchmarkFrame]` 行，汇总均值与 P95	实验二
速度与精度权衡表	fp32 / fp16 / int8 三档：耗时（均值 / P95）× 重投影误差，同一冻结测试集、同一输入	实验四（招牌）
batch 推理方案 + 量化交付	5 对合并一次调用的方案设计；int8 劣化 < 10% 才交付量化版	实验四

一帧 800 ms 花在哪，先看下面的预算条：引擎推理是必须死磕的一段，其余三段也要有人盯着——预算书就是把每一段变成“有名有数有测法”的账。



量化三档与校准集：把模型“压缩”但不压垮

网络权重默认用 **fp32**（32 位浮点）存储与计算；**fp16** 减半，带宽立省；**int8** 用 8 位整数近似浮点，提速潜力最大、风险也最大。代价靠**校准集**控制：拿一小撮有代表性的样本跑一遍，统计各层取值范围，再决定每个数怎么映射到 8 位——校准集要用 val 批次的代表性样本，不能抓训练集凑数。另一条路是 **batch 推理**：5 个相邻对拼成 `[5,2,3,512,512]` 一次调用，省 4 次调用开销——GPU 爱吃“一大口”，不爱吃“五小口”。

前沿速递：量化是这两年的主战场

2024–2026 年大模型落地的关键词就是量化：手机上的 7B 级模型普遍以 4–8 位量化运行，端侧芯片把 fp16/int8 算力当核心卖点，推理框架把“选对硬件档位”做成显式选项。你的三档对比表，与工业界上线模型前的评测流程同构——做过一遍，以后聊“模型部署”就不虚。

项目连线：第 8 讲的实战全对应

第 8 讲“轻量化与分割”里有一条完整的方法链：模型大小与精度的取舍（下图）、压缩网络设计的思路、以及“快到能上手机/上板子”的工程诉求——你的三档权衡表就是把那一讲的方法论用在真模型上。batch 推理的思想在第 3 讲就埋了伏笔：向量化与成批计算总是快于逐个计算。而“视觉无差异”的验收观，来自第 5 讲——指标只是代理，最终要回到图像本身。

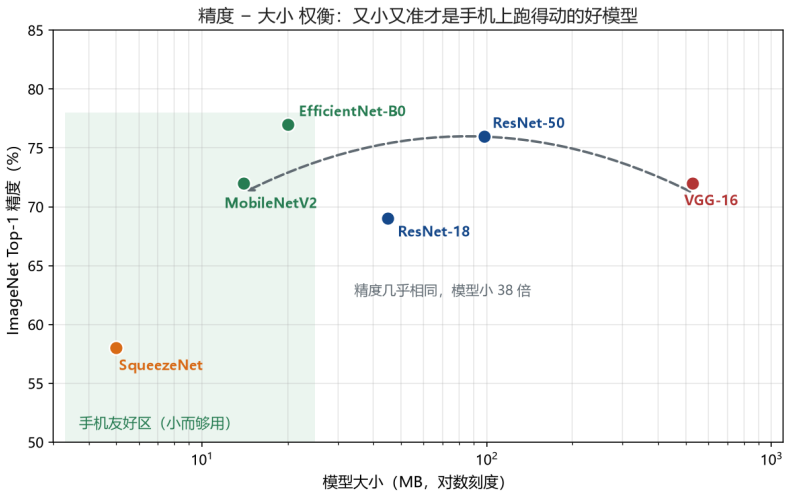


图 8.1　模型大小与精度的取舍：又大又准与又小又快之间永远要做选择——你的三档权衡表，就是给 `ropstitch.onnx` 在这张图上找点

8.4　第 19 周周五提交清单

提交线：2027年 1 月 22 日（第 19 周周五）完全提交离场。你的组交齐下表五样。

表 8.3　第 8 组提交清单（2027 年 1 月 22 日）

交付物	格式	说明
速度与精度权衡表	PDF + xlsx	招牌交付：三档 × 耗时（均值 / P95）× 重投影误差，注明测试集哈希与硬件环境
性能预算书（含复盘）	一页	实验一的预算分配 vs 离场实测复盘，差异要有解释
batch 推理方案设计	文档 2–4 页	5 对合并一次调用的方案、预期收益与验证计划
量化版 onnx（达标才交）	onnx + json	int8 劣化 < 10% 才交付；走 `deliver_model.bat`；探针结果如实附上
组报告与 AI 声明	PDF 8–15 页 + 一页	按第 三 章模板；AI 使用声明逐条列清

8.5　AI 分工与人工必做

分工原则一句话：AI出草稿，人出签字——签字的人对数字负全责。

表 8.4　第 8 组 AI 分工与人工必做

AI可以自动做	人必须亲自做
解读引擎侧源码的耗时构成，起草预算书框架	本机跑通引擎模式 benchmark 与逐帧计时采集
写三档汇总表与画耗时对比图的脚本	亲手复现组 7 v1 并与 `metrics.json` 对账
起草 batch 方案与量化步骤的文字	亲手执行量化与校准，核对劣化是否 < 10%
解释报错、给排查建议	在权衡表与交付物上签字，对每个毫秒担责

求助纪律：组内 → 助教（研究生）→ 导师，求助附完整命令与完整输出。课程成绩建议构成：实验一不计分，实验二至四各 10 分共 30 分，平时作业 20 分（十二讲章后习题十二选二），项目成

果 50 分，总评满分 100 分（教师可调）；每次节点验收以组 5 双台账为唯一依据——你的每毫秒提速，都要能在台账里查到。

8.6　防糊弄指南与满分标准

防人工糊弄指南（第 8 组版）

(1) 换输入比速度 = 换考卷：三档对比必须同一冻结测试集、同一输入、同一机器同一状态，换任何一个条件，表就是废纸。**(2)** 只报耗时隐去精度（或反之）——权衡表的全部价值在“两列都在”；**选择性上报按虚构处理，虚构数据 = 该节点成绩作废**。**(3)** 量化版不达标硬交——劣化 $< 10\%$ 是硬线，越线就如实写“不交付”，交一个糊的全景比慢 50 ms 罪过大得多。**(4)** 交付不走 `deliver_model.bat` 手动拷文件——交付动作只有一条正道，绕道就走在了探针的枪口上。

防 AI 糊弄指南（第 8 组版）

(1) AI 背的耗时数字一律不信：哪段占多少毫秒，只认你机器上 `[BenchmarkFrame]` 日志算出来的数，它背的博客不算。**(2) AI 说“fp16/int8 一定更快”不等于更快**：CPU 上 fp16 可能跑不了甚至更慢——一切以实测为准，实测不了就注明环境限制。**(3) AI 算的均值与 P95 要复算**：用脚本从原始日志重算一遍再进表。**(4)** AI 起草的量化步骤先在小样本上验证过再推广——量化坏起来是“全军覆没”，不是“略有损耗”。

成果满分 50 分，怎么拿每一分见下表。

表 8.5　第 8 组成果满分标准（50 分制）

分值	项目	怎么拿满
20	功能达成	实验二逐帧计时基线完整（5 分）；实验三 v1 复现对账通过（5 分）；实验四三档权衡表 + batch 方案合格（10 分）
10	证据完整	四次实验证据包齐全：预算书、逐帧日志、复现对账记录、三档原始数据；缺一件扣 5 分
8	报告质量	组报告按模板 8–15 页；权衡表讲得清、结论有数据、预算复盘诚实
7	协作与台账	GPU 时段申请与使用有记录；与组 7/6 交接顺畅；结论按时入组 5 台账
5	加分	如：batch 方案做出原型并给出实测收益；单次引擎调用实测压进 300 ms；发现并反馈引擎管线的耗时 bug

8.7　四次实验安排

四个节点与全班同频：实验一“大模型 agent 模式配置及应用”（2026 年 10 月 12 日）、实验二“匹配合适程序并执行本地部署”（11 月 2 日）、实验三“深度网络的模型训练实践”（11 月 23 日）、实验四“模型优化训练实践”（12 月 21 日）。每场按“目标—步骤—验收证据—常见坑”执行。

表 8.6　第 8 组四次实验总览

节点	日期	本组动作	核心验收证据
实验一	10月 12 日	agent 解读耗时构成，写性能预算书	预算书 + 核对记录一页
实验二	11 月 2 日	引擎模式 benchmark，逐帧计时基线	`[BenchmarkFrame]` 日志 + 基线表
实验三	11 月 23 日	复现组 7 v1，指标对账	复现报告 + 探针 `PASS (5/5)`
实验四	12 月 21 日	三档权衡表 + batch 方案 ± 量化交付	权衡表 +（可选）量化版 onnx

8.7.1　实验一（10 月 12 日）：拆解一帧的时间都花在哪，写性能预算书

目标：用大模型 agent 解读引擎一次调用的耗时构成——预处理、骨干网络、后处理各占多少——写一页“性能预算书”，给 800 ms 立规矩。

步骤：(1) 装好编程助手并切到 agent 模式；(2) 让 AI 读引擎侧源码（`e2e_gpu/cpp/src/` 的引擎通道），回答三问：一次调用前后各做了什么（缩放 512、归一化等预处理）、骨干网络前向在算什么、后处理（5 对变换链式累积与渲染）做了什么；(3) 让 AI 给出三段的先验占比估计，并追问“依据是什么”；(4) 逐条与代码对照核对，标“对 / 错 / 存疑”，至少纠正一处；(5) 写预算书：800 ms 的分配表（预处理 / 引擎 ≤ 300 ms / 渲染 / 显示余量），每项配一列“怎么测”。

验收证据：一页性能预算书 + “AI 解读 + 人工核对”记录。

常见坑：AI 把“模型推理”当成唯一耗时——预处理与后处理常常合计占到三成上下，漏记它们预算就是错的；预算书没有“怎么测”一列——不可测量的预算只是愿望清单；AI 拿通用模型的经验数字套本项目——引擎是自己家的，一切以本机实测为准，先验只当假设。

8.7.2　实验二（11 月 2 日）：引擎模式开起来，逐帧计时基线立起来

目标：跑通引擎模式 benchmark，采集逐帧计时，建立你组的第一本“耗时账”。

步骤：(1)部署路线二（部署与探针步骤同组 5 组实验二）：`bin/360stitcher_e2e.exe --benchmark-stitch=bin/simulation --engine=ropstitch`，确认日志出现 `[Engine] loaded` 与 `engine active`；(2) 让 benchmark 跑满 30 帧以上，逐帧收集 `[BenchmarkFrame]` 行（每帧各段耗时）；(3) 用脚本汇总均值与 P95，形成基线表（分：模型加载 / 预处理 / 引擎推理 / 后处理渲染）；(4) 与第 6 组的 `benchmark.py` 交叉验证一次，两本账对得上才算数。

验收证据：完整 `[BenchmarkFrame]` 日志、逐帧计时基线表（均值 + P95）、交叉验证记录。

常见坑：把模型加载（init）算进每帧耗时——加载只发生一次，逐帧账里要单列；后台浏览器、杀毒扫描让计时抖动——关干净、多跑几轮取稳定段；基线没固定输入就开跑——基线表必须注明测试图目录与帧数，下学期回看才知道测的是什么。

8.7.3　实验三（11 月 23 日）：复现第 7 组的公开版 v1，指标对得上账

目标：把第 7 组交付的 v1（公开数据版，实验三即交付——不用等真机数据）在你机器上完整复现一遍——指标可复现、数值对得上账，优化才有出发点。

步骤：(1) 确认 v1 在位：`bin/models/ropstitch.onnx` + `ropstitch.metrics.json`；(2) 跑 `python training/engine_probe_test.py`，期望 `RESULT: PASS (5/5)`——先证明系统用的就是这份模型；(3) 同一

冻结测试集复跑重投影误差与逐帧耗时，与 `metrics.json` 及组 7 的 A/B 数据对账，偏差应在 10^{-3} 量级以内；(4) 写一页复现报告：命令、输出、与原记录的对照表，交组 5 入台账。

验收证据：探针 `PASS (5/5)`、复现对照表（你测的 vs `metrics.json`）、复现报告一页。

常见坑：对不上账先怪上一组——先查自己：测试集哈希对不对、输入尺寸与归一化一致不一致、机器状态相同不相同；中途动过 `bin/models/` 忘了还原——下次探针 FAIL 溯源半天才找到你头上；只复现精度不复现耗时——你的主战场是时间，耗时不复现等于白来。

8.7.4　实验四（12 月 21 日）：三档权衡表与 batch 方案，招牌交付收官

目标：完成 fp32/fp16/int8 三档“速度与精度权衡表”与 5 对合并 batch 推理方案；若 int8 精度达标（重投影误差劣化 $< 10\%$），交付量化版 onnx；本场同时是八组合并联调：对真机版 v2 补做三档对比，达标即交量化版，结论入台账。

步骤：(1) 三档准备：fp32 直接用 v1；fp16 需 GPU 或支持 fp16 的执行环境——向第 7 组申请时段，CPU 不适用就如实注明；int8 量化：校准集从第 6 组 val 批次选代表性样本（别抓训练集凑数）；(2) 同一冻结测试集、同一输入跑三档，记录耗时（均值 / P95）与重投影误差，成“速度与精度权衡表”；(3) 写 batch 方案：5 对拼成 `[5,2,3,512,512]` 一次调用的接口改动、预期收益与验证计划；(4) 达标判定：int8 劣化 $< 10\% \rightarrow$ 走 `deliver_model.bat` 交付量化版并重跑探针、如实记录；不达标 $\rightarrow$ 如实写结论不硬交；(5) 预算书复盘 + 全部结论入组 5 台账。

验收证据：三档权衡表（含原始数据）、batch 方案文档、量化交付与探针记录（或“不达标不交付”的结论说明）、预算书复盘。

常见坑：fp16 硬在 CPU 上跑——CPU 执行环境对 fp16 支持有限，跑不了是正常现象，注明条件去 GPU 上测；比较时忘了固定输入——换图等于换考卷，表作废；int8 校准集用了训练集——校准要有代表性，还要与训练数据拉开距离；量化版交付后探针表现与浮点版不同——如实记录现象，与助教确认判据，别为了绿灯改数据。

第 9 组　水下双目 · 数据组

本章地图

讲什么：第 9 组任务书——“视频体检 → 抽帧去冗 → 双目预处理 → 数据集组装”四步赛道怎么走，四个实验节点各做到什么程度，第 19 周周五交哪几样。**为什么值得读**：模型的天花板是数据踩出来的。四条并行赛道里你守的是“粮草”——训练组、部署组、评测组吃的每一对图，都从你手里过。**学完你能**：独立跑通体检与抽帧去冗全流程，说得清数据泄漏为什么必须按场景切分才防得住，用 2–3 种配比实验为全队选出更好的一口粮。

一句话概括你们组：整条水下双目流水线的“厨房”。别人讨论模型、算力、指标，你们负责把原料做成能下锅的粮食——本项目送给你们的一句话：**你们喂给模型的第一口粮，来自你们手中**。

9.1　组档案

表 9.1　第 9 组档案

项目	水下双目（项目二，中海油 HYSY123 FPSO 锚链 ROV 检查视频的双目立体感知，详见第 二 章）
组号与组名	第 9 组 • 数据组
人数与投入	5 人；每周合计 6–8 小时（组内自行排班，建议每周固定 2 次）
赛道	视频体检 → 抽帧去冗 → 双目预处理 → 数据集组装
上游依赖	开局资产包（500 对锚链图对、UWStereo 子集、零样本基线）；真机新视频由助教协调（实验四接入）
下游依赖	第 10 –12 组共享你的数据集与划分（实验四联调对账）
工作目录	`tasks/u09_data/`（主线 `code/` 只读，见防糊弄指南）
硬件要求	体检与抽帧用普通 CPU 电脑即可；实验三、实验四的小微调需向第 10 组预约 4080 时段

先把你们组的家底交个底：**开局资产包**已经发到组里——500 对锚链校正立体图对（`stereo_dataset`，无标签）、UWStereo 公开子集、零样本基线成绩，外加一版用公开数据初步微调的基线模型 v0。前期最小实现已经跑通，你的工作不再是“等米下锅”，而是把这套粮食管出质量：扩量、提质、调配比。UWStereo / FLSea / SQUID 等公开数据集用于扩量与对照；实验四联调时，助教协调的新录视频接入同一条流水线。另外把求助通道抄在工位上：组内互助 → 研究助理 → 项目负责人，逐级上升；求助必须附完整命令与完整输出，只发一句“跑不通”等于没发。

9.2　项目背景与你的位置

第 二 章讲过来龙去脉：中海油 HYSY123 FPSO 的系泊锚链靠 ROV 定期下水检查，检查视频要变成锚链的三维信息，靠双目立体匹配；技术方案是 FoundationStereo 当教师模型、给海量帧打伪标签、学生模型向它学习，资产包里的 500 对自建立体图对开局就在你手里，UWStereo 公开数据用于扩量对照。问一句：这 500 对、以及将来更多的训练图对，从哪来？——从你们手里来。你在流

水线最上游：体检决定哪些视频值得吃，抽帧去冗决定哪些帧入池，预处理决定图对是否对齐，配比直接决定下游微调的成色上限。教师打标、过滤、学生训练的闭环见下图——闭环入口的“数据”两个字，写的就是你们组。

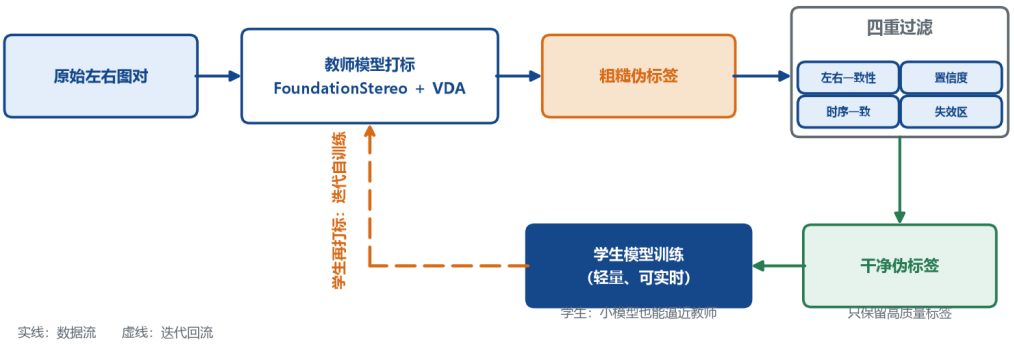


图 9.1　教师–学生伪标签闭环：教师打标、过滤、学生训练、再打标——闭环入口的数据供应正是第 9 组的交付物

9.3　核心任务

四步赛道一张表看明白，随后逐段说人话。

表 9.2　第 9 组四步赛道：干什么与产出什么

阶段	干什么	产出
视频体检	逐段视频出“体检单”：按场景切割，统计模糊度与低照度占比，排出抽帧优先级	体检报告
抽帧去冗	场景内均匀抽帧，近重复帧剔重，记录每帧溯源信息	候选帧池
双目预处理	左右目对齐成校正立体图对（第 二 章的双目校正）	立体图对
数据集组装	按场景/日期级切分训练集与验证集，写数据卡	数据集 v0.1

逐段展开。**视频体检**是给每段 ROV 视频先做一次体检：脚本自动把视频按场景切割（镜头切换、内容突变处下刀），再统计每个场景的模糊度与低照度帧占比——浑浊水里又糊又黑的片段，抽了也是浪费模型胃口，体检单就是“哪段值得吃”的排序依据。**抽帧去冗**：相邻视频帧几乎一模一样，全抽等于让模型反复吃同一道菜；正确做法是场景内均匀抽帧，再用近重复剔除（感知哈希一类指标）去掉“双胞胎帧”，并记录每一帧来自哪段视频的哪一秒——没有溯源信息，评测组将来就没法按场景分桶。**双目预处理**把左右目画面校正成行对齐的立体图对，工程细节第 二 章讲过（下图是一对真实产出的左目），你们的职责是让它批量化、可复现。**数据集组装**是最后一步，也是最容易翻车的一步——翻车的名字叫数据泄漏，见下面的知识盒。

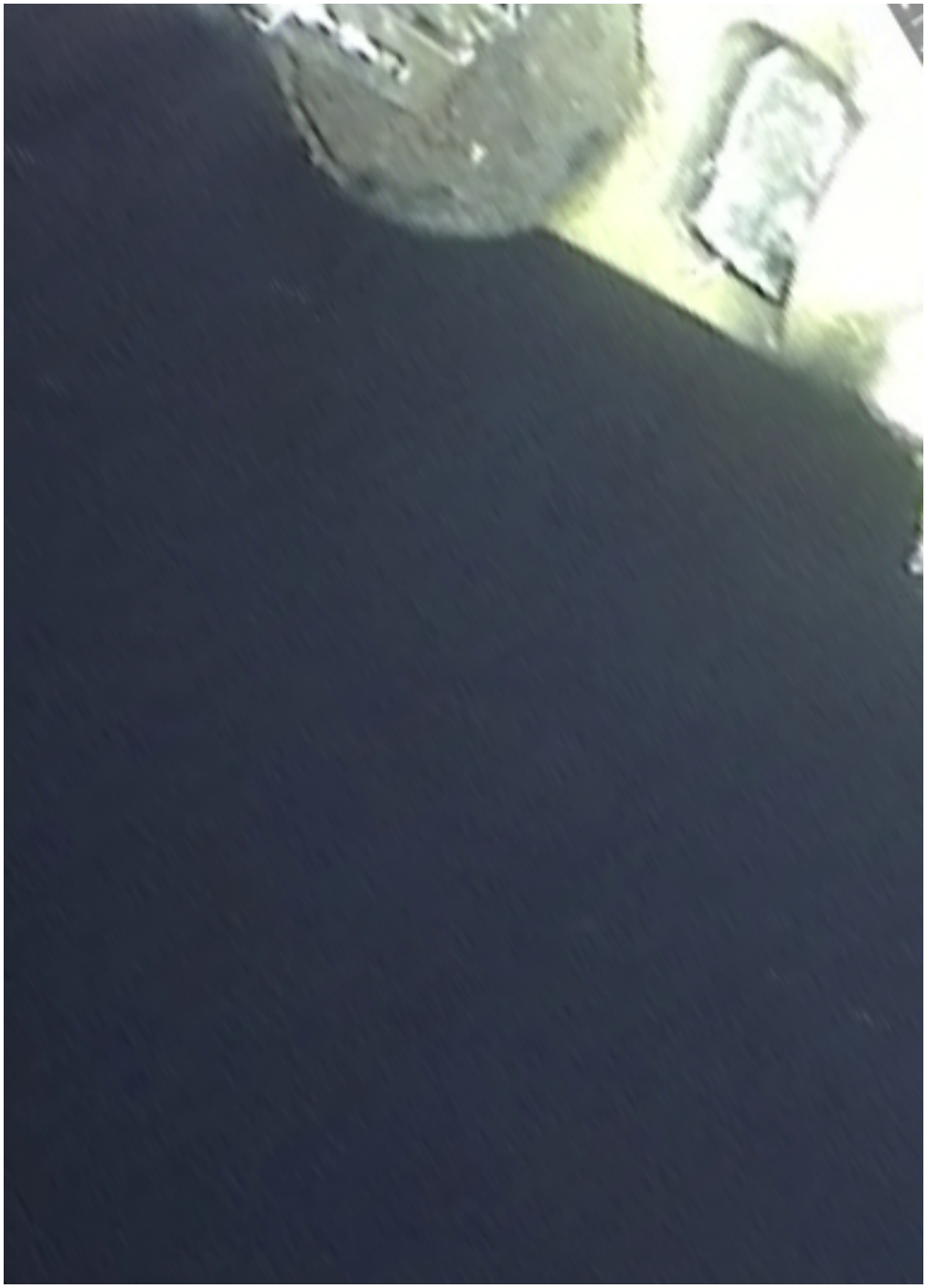


图 9.2　校正立体图对 • 左目（锚链链环清晰、行对齐）：数据组装就是把成百上千对这样的图对管起来

图片来源：本项目真实数据（E:\锚链视频\stereo_dataset）

数据泄漏（data leakage）：验证成绩虚高的头号元凶

一段锚链视频里相邻两帧几乎一样。如果按“帧”随机切分训练集与验证集，同一场景的“双胞胎帧”一边一张——模型在验证集上等于闭卷考试看了答案，验证 EPE（端点误差，越小越好，下一节细讲）好看得不得了，换一段没见过的视频就原形毕露。防法只有一条：**按场景/日期级切分**——同一天、同一场景的所有帧，只能进训练集或验证集中的一边。切分清单本身也是交付物，第 11 组微调、第 12 组评测都按它执行。

项目连线：数据组的活与第一篇哪几讲有关

第 5 讲告诉你，图像在卷积网络眼里只是一摞数字矩阵——你抽的每一帧最终都变成它，所以“模糊度”“低照度”这些体检指标，本质是在给这摞数字的质量打分；第 6 讲讲训练循环与数据增强，你们的去冗与配比本质是“配餐”：同一个模型，粮的好坏直接写进 loss 曲线。实验三你们还要亲手跑一次小微调，把这条因果链亲眼看一遍——数据组也要懂训练。

9.4　第 19 周周五提交清单

提交线：2027年 1 月 22 日（第 19 周周五）完全提交离场。你的组交齐下表各件，每样都与第 三 章的总清单模板对应；水下四个组还要共同填写《四组交接总表》，本表最后一行就是总表里你组的“交接内容”栏。

表 9.3　第 9 组提交清单（2027 年 1 月 22 日）

交付物	格式	说明
体检配置与体检报告	配置 + PDF	主线 `code/tools/video_check.py` 的运行配置；样例视频的体检报告（场景切割 + 模糊度/低照度统计 + 抽帧优先级排序）
水下立体图对数据集 v0.1	目录 + 清单	资产包 500 对的体检与重组子集 + UWStereo 扩量子集；附数据卡（数量/来源/配比）与训练/验证划分清单（场景/日期级）
数据配比报告	PDF，2–4 页	2–3 种配比的小微调验证 EPE 对比、推荐配比与理由
小组项目报告	PDF，8–15 页	按第 三 章模板：摘要/背景/方法/实验/结论/分工声明
证据包	zip	四次实验各一包，命令 + 日志 + 产物三件套，日志为终端完整输出
AI 使用声明	一页	哪些事 AI 做了、人做了什么核验，逐条列清
交接内容	《四组交接总表》一行	数据集 v0.1 与划分清单交第 10/11/12 组；体检报告与配比结论交第 11 组定训练范围

9.5　AI 分工与人工必做

分工原则一句话：AI出草稿，人出签字——数据集里每一个数字，签字的人负全责。

表 9.4　第 9 组 AI 分工与人工必做

AI可以自动做	人必须亲自做
解读体检脚本参数、生成运行命令草稿	**抽 30 帧人工看图核对体检指标排序是否合理（AI 不可代看）**
起草抽帧、去冗、重命名脚本草稿	在本机与 4080 上实跑每一条命令
把体检数字画成图表，起草数据卡与报告文字	逐条核对 AI 结论（在脚本里找到对应行才算数）
建议数据配比方案与过滤阈值候选	亲眼确认划分清单无泄漏（同场景只进一边）
解释报错信息、给排查建议	在证据包与数据卡上签字，对数字担责

9.6　防糊弄指南与满分标准

防人工糊弄指南（第 9 组版）

先背全项目通用的纪律红线：**(1) 原始视频/帧不上云盘、不进在线 AI（保密!）**——体检统计数字可以写进提示词，视频与帧本体绝对不行；**(2) 主线 `code/` 只读不改**，你们的一切产出放进 `tasks/u09_data/`；**(3) 不直接联系甲方**，数据需求一律走项目负责人。本组再加三条：**(4) 抽帧只报数量不报去冗率 = 没做去冗**——“双胞胎帧”糊弄不了训练曲线；**(5) 划分清单混场景 = 数据泄漏**，验证 EPE 再好看也不算数，一经查实该节点成绩作废；**(6) 代跑必须留运行者姓名与时间**，谁跑的谁负责。

防 AI 糊弄指南（第 9 组版）

(1) AI 产出必须实跑验证——第四条纪律红线：它声称“脚本写好了、应该能跑”，只有你终端里跑出来的那份算数。**(2) AI 没看过你的视频**——它对“这段视频浑不浑”的任何回答都是编的，看图的事只能你来，这正是 30 帧人工核对 AI 不可代看的原因。**(3) AI 给的阈值（切割阈值、去重阈值）是“常见值”不是“你们数据的合理值”**——必须用体检数字调出来。**(4) 禁止把 AI 的回答直接粘进数据卡与实验记录**——记录里只许出现你亲手跑出来的内容。

按课程建议方案，实验一不计分，实验二至四各 10 分共 30 分，平时作业 20 分（十二讲章后习题十二选二）；期末成果满分 50 分（教师可调）。成果怎么拿满，看下面这张 50 分制细则。

表 9.5　第 9 组成果满分标准（50 分制）

分值	项目	怎么拿满
20	功能达成	实验一体检报告 + 30 帧人工核对记录（5 分）；实验二抽帧去冗全流程 + 防泄漏划分清单（5 分）；实验三资产包数据小微调、亲眼看 loss 下降（5 分）；实验四 2–3 种配比对比 + 推荐配比（5 分）
10	证据完整	四次实验证据包三件套齐全、可复现；缺一件扣 5 分
8	报告质量	组报告按模板 8–15 页，数据卡清晰、配比结论有 EPE 数据支撑
7	协作与台账	排班执行有记录；4080 时段预约与用量记录无缺失；求助记录规范
5	加分	如：自研更聪明的去冗指标并用数字证明有效；为浑浊/低照度场景设计专门过滤规则并被训练组采用；把体检做成一键复跑的工具

9.7　四次实验安排

四个节点与全班同频（实验一 2026 年 10 月 12 日、实验二 11 月 2 日、实验三 11 月 23 日、实验四 12 月 21 日），但每场都有本组的命令级动作。先看总览表，再逐场展开；每场按“目标—步骤—验收证据—常见坑”执行：步骤照抄就能跑，证据一栏就是成绩。总则：实验一至三是离线自研——资产包开局就发到每组手里，谁也不用等谁；实验四是合并联调——四条赛道的成果在同一台机器、同一套考卷上会师。

表 9.6　第 9 组四次实验总览

节点	日期	本组动作	核心验收证据
实验一	10月 12 日	配 agent，实跑视频体检，人工核对 30 帧	体检报告 + 核对记录
实验二	11 月 2 日	工具链装通，抽帧去冗全流程 + 防泄漏切分	去冗日志 + 划分清单
实验三	11 月 23 日	干跑数据集小微调 1–2 个 epoch	训练日志 + loss 曲线
实验四	12 月 21 日	2–3 种配比对比，选出更优配比	EPE 对比表 + 推荐结论

9.7.1　实验一（10 月 12 日）：给视频开体检单，你亲自看 30 帧

目标：配好大模型编程助手（GLM类命令行 agent），实跑“视频体检脚本”，产出第一份体检报告，并由人工核对排序是否合理。

步骤：(1) 按课堂演示装好编程助手并切换到 agent 模式；(2) 让 agent 解读主线 `code/tools/video_check.py` 的参数——场景切割阈值怎么设、模糊度与低照度怎么判——并把它的解释逐条与源码对照，标注“对/错/存疑”；(3) 在 `tasks/u09_data/` 下写运行配置，对一段样例视频实跑：`python code/tools/video_check.py --input demo.mp4 --out tasks/u09_data/check/`，得到按“最值得抽帧”排序的体检报告；(4) **抽 30 帧人工看图**，核对排序是否合理——糊的、黑的是不是真排到了后面（AI 不可代看，这半小时没有人能替你）。

验收证据：agent 解读 + 人工核对记录一页（含 AI 原话、你的核对结论、至少一处纠正）、体检报告、30 帧看图核对记录。

常见坑：**ffmpeg 装了但 PATH 没配**——命令行敲 `ffmpeg -version` 报“不是内部或外部命令”，脚本起不来，把 ffmpeg 的 bin 目录加进系统 PATH 再重开终端即可；agent 替你“想象”体检结果——它没看过视频，数字必须来自你的终端；只核对前 5 帧就签字——排序靠后的片段才是抽查重点。

9.7.2　实验二（11 月 2 日）：抽帧去冗一条龙，切分防住泄漏

目标：装通 ffmpeg / PySceneDetect / OpenCV 工具链，跑通“场景切割 → 段内抽帧 → 近重复去冗 → 按场景/日期切分”全流程，组装干跑数据集 v0.1 的雏形。

步骤：(1) 装三件套并逐个验证：`ffmpeg -version`、`scenedetect version`、`python -c "import cv2"`，三条都有版本号输出才算装通；(2) 用 PySceneDetect 切场景：`scenedetect -i demo.mp4 detect-content list-scenes`；(3) 场景内均匀抽帧，再用感知哈希做近重复剔除（阈值先抄默认值，再用体检数字调）；(4) 按场景/日期级切分训练集与验证集——同一天同一场景只进一边；(5) 生成划分清单与数据卡，每帧记录溯源信息（哪段视频、哪一秒、哪个场景）。

验收证据：工具链三条版本号输出；抽帧与去冗日志（抽了多少、去掉多少、留了多少）；训练/验证划分清单与数据卡。

常见坑：**抽出来的重复图没去冗**——目录看着热闹，全是双胞胎帧，模型等于反复吃一道菜；按帧随机切分导致数据泄漏——切分粒度是场景/日期，不是帧；光顾着抽帧忘了记溯源信息——评测组没法按场景分桶，返工的是你们自己。

9.7.3　实验三（11 月 23 日）：数据组也要懂训练——亲眼看 loss 下降

目标：用资产包数据（500对采样与 UWStereo 混合），在 4080 上跑一次小规模微调（1–2 个 epoch 即可，FoundationStereo 或第 10 组降级链可跑的版本），亲眼看 loss 下降——理解“数据怎么

喂、模型怎么吃”。

步骤：(1)向第 10 组预约 4080 时段，按其部署手册激活环境；(2) 用第 11 组同款的启动命令，把数据路径换成你自己的数据集，跑 1–2 个 epoch 的小微调；(3) 记下初始 loss 与每个 epoch 结束的 loss，画成曲线；(4) 换一版你改过的数据子集（如去掉低照度帧）再跑一次，对比两条 loss 曲线——这就是“数据质量 → 训练效果”的最小实证。

验收证据：两段训练日志（含每个 epoch 的 loss 数字）、两条 loss 曲线、数据子集差异说明。

常见坑：数据路径写错，dataloader 空转却“看起来在训”——先确认日志里读到了多少对图；epoch 贪多超出 1–2 的范围，占用别人的 4080 时段——时段是全队的；只看最后一步 loss 不看走势——一条降、一条平还是要能说清原因。

9.7.4　实验四（12 月 21 日）：换配比、调阈值，用验证 EPE 选出更好的一口粮

目标：调整数据配比与过滤阈值，对比 2–3 种配比下小微调的验证 EPE，找出更优配比，形成“配比—效果”结论交给全队——本场同时是四组合并联调：新配比数据集接入第 11 组重训、与第 12 组统一口径总评测，成果进《四组交接总表》。

步骤：(1) 设计 2–3 种配比（如：全量 / 去低照度 / 清澈场景加权），验证集固定为同一份；(2) 每种配比各跑一次小微调，训练参数与实验三完全一致，只动数据；(3) 用第 12 组的一键评测脚本（统一口径）测每种配比下的验证 EPE；(4) 填对比表：配比、帧数、验证 EPE、训练耗时，写不少于 100 字的结论并给出推荐配比。

验收证据：配比设计说明、各配比训练日志、EPE 对比表、推荐结论。

常见坑：换配比时顺手改了学习率或 epoch——变量没控住，对比无效；验证集不固定，三个配比三份验证集，数字没法比；只比 EPE 不记耗时——推荐配比说不清代价，训练组没法排时段。

第 10 组　水下双目 · 部署组

本章地图

讲什么：第 10 组任务书——“环境部署 → 零样本冒烟 → 微调护航 → 性能优化”四步怎么走：FoundationStereo 怎么在一块真实的 GPU 上跑起来、跑得快，每个实验留什么证据。**为什么值得读**：教师模型是十亿级参数的立体大模型，它装不装得上、跑不跑得动，决定整个伪标签方案成不成立。全队的 4080，归你管。**学完你能**：独立完成 GPU 环境部署与故障降级，用 EPE/耗时/显存三项数字完成模型“选定”，给出有数据支撑的“精度–速度”权衡建议。

一句话概括你们组：全队的“发动机舱”。你们的任务书标题可以换成一句话：**让大模型在一块真实的 GPU 上跑起来、跑得快**——跑不起来，后面的训练评测全是纸上谈兵；跑不快，全队的粮食预算就要砍半。

10.1　组档案

表 10.1　第 10 组档案

项目	水下双目（项目二，中海油 HYSY123 FPSO 锚链 ROV 检查视频的双目立体感知，详见第 二 章）
组号与组名	第 10 组 • 部署组
人数与投入	5 人；每周合计 6–8 小时（组内自行排班，建议每周固定 2 次）
赛道	环境部署 → 零样本冒烟 → 微调护航 → 性能优化
上游依赖	开局资产包：FoundationStereo 权重与可跑环境（最小实现已完成——你是它的压测员与优化师，不是从零部署者）
下游依赖	第 11 组（微调环境与闸门时段）、第 12 组（推理与评测链路）；实验四联调统一用你验证过的环境
工作目录	`tasks/u10_deploy/`（主线 `code/` 只读，见防糊弄指南）
硬件要求	RTX 4080 工作站一台（32GB，全队共用，你组排班值守）；其余成员用 CPU 电脑做脚本与文档

先说清楚你们最值钱的资产：那块 4080 是全队共用的发动机——第 9 组微调、第 11 组训练、第 12 组批量推理都要来你这约时段。你们组的隐性职责是**时段台账**：谁、哪天、用了多久、跑了什么，一条不缺。台账记全，排班才不吵架，故障才能溯源。求助通道照旧：组内互助 → 研究助理 → 项目负责人，逐级上升，求助必须附完整命令与完整输出——GPU 报错千奇百怪，截半张图等于让医生看半张化验单。

10.2　项目背景与你的位置

第 二 章讲过方案：ROV 检查视频靠双目立体匹配变成锚链三维信息，路线是 FoundationStereo 当教师打伪标签、学生模型学习，；前期最小实现已把这条路走通——500 对数据、初步微调基线 v0 都在开局资产包里。你这一关的任务：把“能跑”带到“跑得快、跑得稳”。这套方案落地的第一关

就在你们组——教师模型参数量大，conda 环境、CUDA 版本、flash-attn 编译，任何一环卡住，全项目停摆。你在“发动机舱”：第 11 组的训练、第 12 组的评测、第 9 组的微调实践，全部跑在你验证过的环境里。你交付的不是模型，而是“别人开机即用”的环境、手册与三项数字。本组的特色纪律是：**基础模型的“选定”用实跑数字完成；失败则按降级链处置并书面报告**——降级链是 vitl（大杯）→ vits（中杯）→ IGEV++（另一款开源立体匹配模型），每一级都要留下“卡在哪、换了什么、数字多少”的记录。

10.3　核心任务

四项任务一张表，随后说人话。

表 10.2　第 10 组四项任务：干什么与产出什么

任务	干什么	产出
环境部署	按清单装通 conda / CUDA / flash-attn，跑通 `run_demo.py`	部署手册
零样本冒烟	UWStereo 50对零样本推理，EPE/耗时/显存三项都记	冒烟报告
微调护航	给第 9/11 组提供训练环境与时段，记录显存曲线与吞吐	显存/吞吐报告
性能优化	批量/分辨率/FP16 对比 + ONNX 导出初探，给出“精度–速度”权衡建议	权衡建议

先补一个词：**零样本**（zero-shot）——不做任何微调，拿预训练权重直接推理。零样本冒烟的意义是给全队一个“出厂底分”：教师模型没经过水下微调时到底几斤几两，第 11 组将来“微调提升多少”，全靠这个数当分母。逐项展开：**环境部署**是体力活加侦探活，照清单装，装不上按降级链换；**零样本冒烟**要三项数字一个不少——EPE（精度）、单对耗时（速度）、显存峰值（能不能装下）；**微调护航**是你们对全队的服务义务，训练时段的显存曲线与吞吐（对/分钟）由你们记录并公告；**性能优化**是学期末的收官戏，三个旋钮扫一遍，再摸一摸 ONNX 导出这条路通不通。模型大小与精度的取舍直觉见下图——你们实验四干的就是在真实模型上量这条曲线。

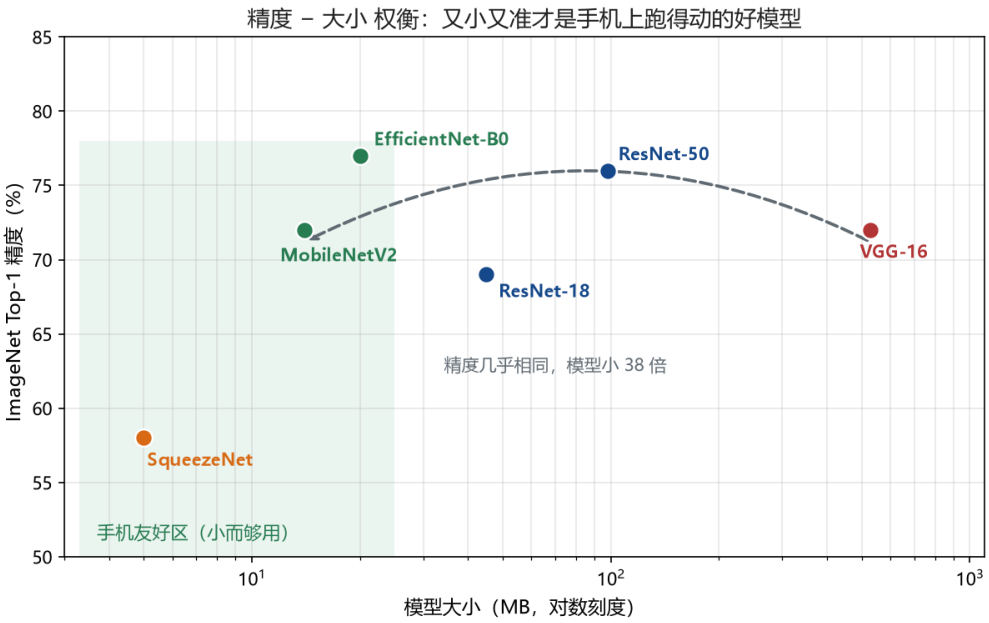


图 10.1　模型大小与精度的取舍散点（示意）：更大通常更准但更慢更吃显存——部署组的日常就是在真实权重上量出这条权衡

显存去哪了：权重、激活、梯度三口锅

模型参数本身（权重）占一口锅；推理时 layer 之间的中间结果（激活）占第二口；训练时还要额外存梯度与优化器状态，第三口最大——所以**微调远比推理吃显存**。可控的三个旋钮：**批量大小**（batch size，一次一起送几张图，越大越吃显存）、**分辨率**（图小一半，显存与耗时大幅下降）、**精度**（FP16 半精度，数字从 32 位存成 16 位，省显存提速，但可能掉精度）。实验四扫的就是这三个旋钮。

项目连线：部署组的活与第一篇哪几讲有关

第 8 讲是你们的方法论老家：轻量化与部署——模型压缩、深度可分离卷积、ONNX 格式、“精度换速度”的整套取舍，都在那一讲打过样。你们学期末的权衡建议，就是那一讲思想在十亿级立体大模型上的实弹演习。

10.4　第 19 周周五提交清单

提交线：2027年 1 月 22 日（第 19 周周五）完全提交离场。你的组交齐下表各件，每样都与第 三 章的总清单模板对应；《四组交接总表》由水下四组共填，本表最后一行就是总表里你组的“交接内容”栏。

表 10.3　第 10 组提交清单（2027 年 1 月 22 日）

交付物	格式	说明
部署手册	PDF/Markdown	从裸机到 `run_demo.py` 的每一条命令与每一个坑，别人照做必须能复现
冒烟报告	PDF，2–4 页	UWStereo 50 对零样本 EPE/耗时/显存三项数字；如走降级链，附完整处置记录
性能优化与权衡建议	PDF，2–4 页	批量/分辨率/FP16 三组对比表 + ONNX 导出初探测速 + 推荐设置
小组项目报告	PDF，8–15 页	按第 三 章模板：摘要/背景/方法/实验/结论/分工声明
证据包	zip	四次实验各一包，命令 + 日志 + 产物三件套，日志为终端完整输出
AI 使用声明	一页	哪些事 AI 做了、人做了什么核验，逐条列清
交接内容	《四组交接总表》一行	部署手册与环境清单交第 9/11/12 组；显存曲线与吞吐数字交全队排时段用

10.5　AI 分工与人工必做

分工原则一句话：AI出草稿，人出签字——4080 上跑出的每个数字，签字的人负全责。

表 10.4　第 10 组 AI 分工与人工必做

AI可以自动做	人必须亲自做
生成“4080 环境配置清单”草稿	在 4080 上实跑每一条命令
解读依赖冲突、CUDA 与 flash-attn 编译报错	**逐条对照官方仓库核实清单（以仓库为准，不以 AI 为准）**
写显存采样与测速脚本草稿	记全 EPE/耗时/显存三项数字，缺一不可
建议降级链处置方案、起草手册与报告	降级链处置与推荐设置逐项签字
解释报错信息、给排查建议	在证据包与台账上签字，对数字担责

10.6　防糊弄指南与满分标准

防人工糊弄指南（第 10 组版）

先背全项目通用的纪律红线：**(1) 原始视频/帧不上云盘、不进在线 AI（保密!）**——公开数据集不涉密，甲方 ROV 视频帧绝对涉密，测试输入只用公开数据；**(2) 主线 `code/` 只读不改**，你们的一切产出放进 `tasks/u10_deploy/`；**(3) 不直接联系甲方**。本组再加三条：**(4) 冒烟跑通 1 对就当 50 对交差 = 没做**——50 对的均值才有资格当基线；**(5) 显存只记平均值不记峰值 = 没记**——爆显存永远发生在峰值；**(6) 代跑与值守必须留姓名与时间**，时段台账一条不缺。

防 AI 糊弄指南（第 10 组版）

(1) AI 产出必须实跑验证——第四条纪律红线：它说“这样能装上”，只有你 4080 上的终端说了算。**(2) AI 报的版本号十有八九过时或虚构**——CUDA、PyTorch、flash-attn 的版本对应关系，以官方仓库与 `conda list` 输出为准，它背的是去年的黄历。**(3) AI 说“跳过 flash-attn 没**

关系”不算结论——跳过可以，但必须实跑确认对速度与显存的影响并写进降级链记录。**(4) AI 报的耗时/显存/EPE 必须与你的脚本和 `nvidia-smi` 输出对上账**，对不上以你的终端为准。

按课程建议方案，实验一不计分，实验二至四各 10 分共 30 分，平时作业 20 分（十二讲章后习题十二选二）；期末成果满分 50 分（教师可调）。成果怎么拿满，看下面这张 50 分制细则。

表 10.5　第 10 组成果满分标准（50 分制）

分值	项目	怎么拿满
20	功能达成	实验一配置清单经逐条核实（3 分）；实验二 `run_demo.py` 跑通 + 50 对冒烟三项数字齐全（10 分）；实验三微调护航显存曲线与吞吐（2 分）；实验四三组对比 + 权衡建议（5 分）
10	证据完整	四次实验证据包三件套齐全、可复现；缺一件扣 5 分
8	报告质量	组报告按模板 8–15 页，部署手册可复现、权衡建议有数据支撑
7	协作与台账	4080 时段台账无缺失；值守与代跑记录规范；求助记录规范
5	加分	如：ONNX 导出真跑通并给出可用性结论；写一键环境校验脚本被三组采用；FP16 精度损失的定量分析

10.7　四次实验安排

四个节点与全班同频（实验一 2026 年 10 月 12 日、实验二 11 月 2 日、实验三 11 月 23 日、实验四 12 月 21 日），每场都有本组的命令级动作。先看总览表，再逐场展开；每场按“目标—步骤—验收证据—常见坑”执行：步骤照抄就能跑，证据一栏就是成绩。总则：实验一至三是离线自研——资产包开局就发到每组手里，谁也不用等谁；实验四是合并联调——四条赛道的成果在同一台机器、同一套考卷上会师。

表 10.6　第 10 组四次实验总览

节点	日期	本组动作	核心验收证据
实验一	10月 12 日	agent 生成环境清单，逐条对照官方仓库核实	核实版清单
实验二	11 月 2 日	4080 部署 + `run_demo.py` + 50 对冒烟	三项数字表 + 处置记录
实验三	11 月 23 日	部署环境跑通小微调，记显存曲线与吞吐	显存曲线 + 吞吐数字
实验四	12 月 21 日	批量/分辨率/FP16 对比 + ONNX 初探	对比表 + 权衡建议

10.7.1　实验一（10 月 12 日）：让 agent 开清单，你逐条当审计

目标：用 agent 生成“4080 环境配置清单”草稿（conda / CUDA / flash-attn），并逐条对照官方仓库核实——清单是你们整个学期的施工图，错一条，实验二还你一夜。

步骤：(1) 让 agent 读 FoundationStereo 官方仓库说明，产出清单草稿：Python 版本、PyTorch 与 CUDA 版本、`flash-attn` 版本、其余依赖；(2) 逐条到官方仓库核对（README、环境文件、`requirements.txt`），标注“对/错/存疑”——**以仓库为准，不以 AI 为准**；(3) 每一条后面补一行“怎么验证装好

了”，如 `python -c "import torch; print(torch.cuda.is_available())"` 应输出 `True`；(4) 定稿存 `tasks/u10_deploy/env_checklist.md`。

验收证据：清单草稿、逐条核对标记版、带验证命令的终版清单，三样齐全。

常见坑：AI编造不存在的版本组合——它把“常见搭配”当“官方要求”；官方仓库更新后 AI 的记忆停在旧版本——核对日期写进清单；清单只有“装什么”没有“怎么验”——验收时按缺一半算。

10.7.2　实验二（11 月 2 日）：装上 4080，跑通第一对图（本组主力）

目标：按清单复现部署（资产包里已有可跑环境——你的价值是独立复现与压测），跑通 `run_demo.py`；UWStereo 50 对零样本冒烟，EPE/耗时/显存三项都记；受阻按降级链（vitl → vits → IGEV++）验证并记录。

步骤：(1) 按清单装环境，`conda list` 全量存档；(2) 下载权重并校验完整性，先跑官方样例 `python run_demo.py`，看到视差图输出才算部署成功；(3) 跑 UWStereo 50 对零样本推理，逐对记 EPE、单对耗时、显存峰值，汇总成三项数字表；(4) 受阻按降级链处置：vitl 装不上或跑不动 → vits → IGEV++，每级记录“卡在哪、换了什么、三项数字多少”；(5) 中途显存爆就降分辨率重试，同样记录。

验收证据：`run_demo.py` 成功日志；50 对三项数字表；降级链处置记录（如有）。

常见坑：**flash-attn 装不上**——十有八九是版本与 CUDA 不匹配，别死磕一整晚，按降级链走并记录，降级不是失败，无记录才是失败；**显存爆**——先降分辨率，再降批量；权重文件没下全，报错长得像“模型坏了”，先校验文件完整性再怀疑人生；驱动与 CUDA 版本不匹配——`nvidia-smi` 先看驱动支持的 CUDA 上限。

10.7.3　实验三（11 月 23 日）：微调护航——你的环境要能“载训练”

目标：在自己部署的环境里跑通小规模微调（与第 11 组同款配置、1–2 个 epoch），记录显存占用曲线与吞吐，给全队的训练排班拿出数字依据。

步骤：(1) 与第 11 组对齐启动命令与配置快照；(2) 用第 9 组的资产包数据（500 对采样）跑 1–2 个 epoch 微调冒烟；(3) 开后台采样记录显存曲线（按秒采 `nvidia-smi`）与吞吐（对/分钟）；(4) 给出估算：一块 4080 一周能吃多少对数据——这个数字直接决定全队的训练计划。

验收证据：微调成功日志、标了峰值的显存曲线、吞吐数字与排班建议。

常见坑：只记显存平均值不记峰值——爆显存永远发生在峰值时刻；训练时有人同时在上面临时推理，数字互相污染——训练时段 4080 独占，写进值守制度；曲线没标时间轴，事后没法与训练日志对账。

10.7.4　实验四（12 月 21 日）：三组旋钮 + ONNX 初探，给出权衡建议

目标：推理性能优化——批量大小/分辨率/精度（FP16）对比，ONNX导出初探与测速，给出有数字撑腰的“精度–速度”权衡建议——本场同时是四组合并联调：优化结论与第 11 组重训、第 12 组总评测对账，成果进《四组交接总表》。

步骤：(1) 固定同一批 20 对测试图与同一套评测口径（请第 12 组把关）；(2) 扫三组旋钮：批量 1/2/4、分辨率 1.0x/0.5x、FP32/FP16，每组记 EPE、单对耗时、显存峰值；(3) ONNX 导出初探：能导出则测速并记录不支持的算子，导不出则写清卡在哪一步、报错原文是什么；(4) 写一页权衡建议：推荐一组“够用且快”的设置，每个推荐后面都跟着数字。

验收证据：三组对比表（EPE + 耗时 + 显存）、ONNX 导出记录、权衡建议一页。

常见坑：FP16 只记提速不记精度损失——快了 30%、差了 3 个像素，白快；改了分辨率却拿不同口径的 EPE 比——对比无效；ONNX 导出失败就悄悄跳过——初探的价值恰恰在“为什么导不出”的记录里。

第 11 组　水下双目 · 训练组

本章地图

讲什么：第 11 组任务书——“零样本基线 → UWStereo 微调复现 → 训练策略与超参优化”三步赛道怎么走，四次实验各做到什么程度，第 19 周周五交哪几样。**为什么值得读**：你们是全项目唯一“改模型权重”的组。微调成不成立，直接决定伪标签闭环转不转得起来；而这一切的前提是先有零样本基线。**学完你能**：把训练配置读成人话，亲手让一个立体大模型在你的数据上 loss 下降，用“一次只动一个旋钮”的方法做超参对比实验。

一句话概括你们组：全项目的主战部队。项目送给你们一句话：**亲手把一个立体大模型“教会”一点水下本领**——它本来是岸上练出来的好眼力，你们负责把它掰得看得清浑水里的锚链。

11.1　组档案

表 11.1　第 11 组档案

项目	水下双目（项目二，中海油 HYSY123 FPSO 锚链 ROV 检查视频的双目立体感知，详见第 二 章）
组号与组名	第 11 组 • 训练组
人数与投入	5 人；每周合计 6–8 小时（组内自行排班，建议每周固定 2 次）
赛道	零样本基线 → UWStereo 微调复现 → 训练策略与超参优化
上游依赖	开局资产包：500 对图对、UWStereo 子集、初步模型 v0 与零样本基线（你的对照组开局就有）；第 10 组的环境与 4080 时段
下游依赖	第 12 组评测你的权重；实验四联调时接受第 9 组新配比数据续训
工作目录	`tasks/u11_train/`（主线 `code/` 只读；改超参只改自己目录下的配置副本）
硬件要求	**需 4080 时段**（向第 10 组预约，时段与用量记入台账）；日常分析与画图用 CPU 电脑即可

先记住你们组最重要的一条战场纪律：**先基线，后微调**。没有零样本基线，一切“提升了多少”都无法证明——这句在实验三还会重申，因为它是最常见的翻车点。另一条是公共资源纪律：训练闸门与 4080 时段（见防糊弄指南），管住的是全队的发动机。求助通道照旧：组内互助 → 研究助理 → 项目负责人，逐级上升；求助必须附完整命令与完整输出——训练报错往往在几千行日志的最后一行。

11.2　项目背景与你的位置

第 二 章讲过：锚链 ROV 检查视频靠双目立体匹配变成三维信息，方案是 FoundationStereo 当教师打伪标签、学生模型学习；500 对自建立体图对已在开局资产包里，前期还用公开数据初步微调出基线模型 v0。FoundationStereo 零样本已经很强，但水下浑浊、低照度仍是它的软肋——这正是你们的战场。第 9 组喂你数据，第 10 组给你机房与时段，第 12 组给你打分：你们站在流水线正中央，

全项目唯一有权改模型权重的组。你们的产出很朴素：一版权重、一组曲线、一张“零样本 vs 微调”的对比表——但对全队来说，这就是主战成果本身。

11.3　核心任务

三步赛道一张表，随后说人话。

表 11.2　第 11 组三步赛道：干什么与产出什么

阶段	干什么	产出
零样本基线	预训练权重直接在验证集上推理打分，立起“赛前成绩”	基线 EPE 数字
微调复现	用 UWStereo 数据短程微调，复现“零样本 → 微调”对照	微调权重 v1 + 曲线
超参优化	学习率/主干 lr 比例/数据增强中选 1–2 项做 2–3 组对比	对比表 + 结论

训练的全部旋钮都集中在 `configs/train_uw_fresh.yaml` 一个文件里：**lr**（学习率，每一步沿梯度方向挪多大步子）、**bs**（batch size，一次一起送进网络多少对图）、**iters**（总迭代步数）与 **epoch**（全量数据完整过几遍）。实验一的任务，就是让 agent 逐项解释它们，你核对后写成“配置说明书”——训练之前先读懂自己要拧的每一个钮。训练本身就是一个环（下图）：取一批数据、前向、算 loss、反向、更新，转一圈进一步；你们这学期的工作就是盯住这个环，让它在 4080 上转得稳、转得值。

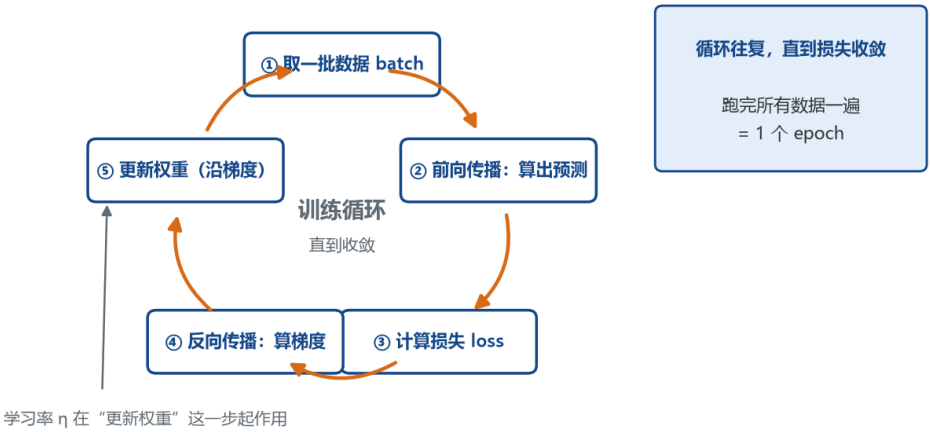


图 11.1　训练循环流程环：取 batch → 前向 → 算 loss → 反向 → 更新——实验二你要亲手让它转起来

超参数：训练开始前就要拍板定好的数

与权重（训练中学出来的数）相对，**超参数**是训练前你定下的数。学习率太大，loss 震荡甚至起飞；太小，磨蹭不前（见下图三条曲线）。batch size 影响显存占用与训练稳定性；epoch 太多会过拟合——训练 loss 还在降、验证指标反而变差，第 3 讲的双曲线就是它。改超参的正确姿势：**一次只动一个，其余全部不动**，否则说不清谁起了作用。

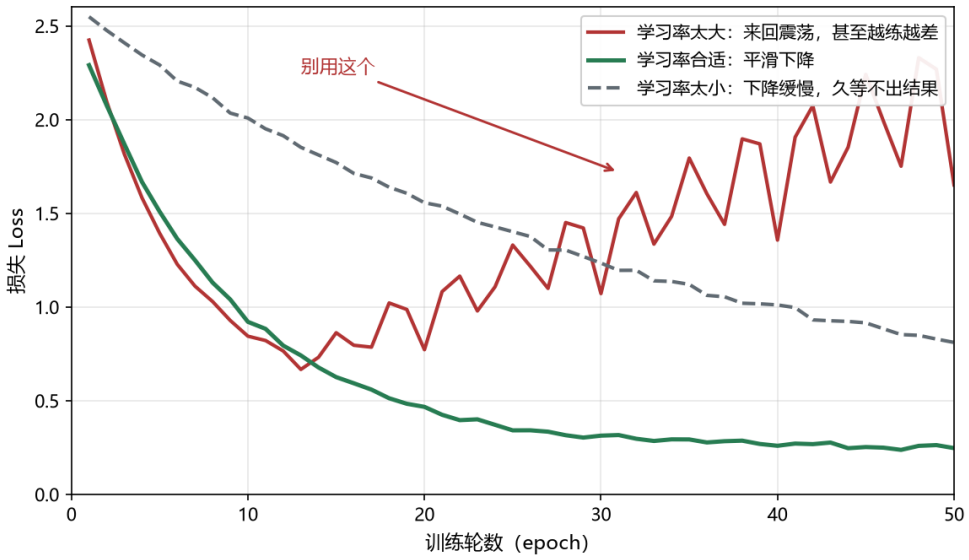


图 11.2　三条学习率下的 loss 曲线：太大震荡上行、合适平滑下降、太小磨蹭不前——这是实验四对比实验的直觉底图

项目连线：训练组的活与第一篇哪几讲有关

方法论底座是第 3 讲——loss、学习率、训练/验证双曲线、过拟合与早停，全部在那讲打过样，你们现在要在真实大模型上重演一遍。格局来源是第 7 讲——“预训练大模型 + 领域微调”正是那一讲迁移学习思想的当代放大版：FoundationStereo 在海量数据上学会了“看立体”，你只用几千对水下数据把它的眼光“掰”向浑浊水域。

11.4　第 19 周周五提交清单

提交线：2027年 1 月 22 日（第 19 周周五）完全提交离场。你的组交齐下表各件，每样都与第 三 章的总清单模板对应；《四组交接总表》由水下四组共填，本表最后一行就是总表里你组的“交接内容”栏。

表 11.3　第 11 组提交清单（2027 年 1 月 22 日）

交付物	格式	说明
微调权重 v1	权重文件 + 快照	经闸门批准的短程微调权重（不超过 5 个 epoch），附配置快照与随机种子
训练曲线包	PNG + CSV	loss 与验证 EPE 双曲线，零样本基线同图对照
对照报告	PDF，2–4 页	“零样本 vs 微调”验证 EPE 对比、结论与下一步建议
配置说明书	PDF/Markdown	`configs/train_uw_fresh.yaml` 逐项人话解释（实验一产出，随报告定稿）
小组项目报告	PDF，8–15 页	按第 三 章模板：摘要/背景/方法/实验/结论/分工声明
证据包	zip	四次实验各一包，命令 + 日志 + 产物三件套，日志为终端完整输出
AI 使用声明	一页	哪些事 AI 做了、人做了什么核验，逐条列清
交接内容	《四组交接总表》一行	权重 v1 与训练曲线交第 12 组系统评测；最优超参结论交第 9 组配比实验参考

11.5　AI 分工与人工必做

分工原则一句话：AI出草稿，人出签字——每一行训练日志、每一个验证 EPE，签字的人负全责。

表 11.4　第 11 组 AI 分工与人工必做

AI可以自动做	人必须亲自做
逐项解释配置文件与训练代码	在 4080 上实跑训练，全程盯 loss 曲线
写画曲线脚本与日志解析脚本	**提交闸门申请（epoch 数、启动时机）并等批准**
解读训练异常（loss 为 NaN、不下降）	先测零样本基线，再开微调
起草配置说明书、报告与闸门申请文字	核对 AI 解释与配置原文逐条一致
建议超参对比方案	在权重交付单与证据包上签字，对数字担责

11.6　防糊弄指南与满分标准

防人工糊弄指南（第 11 组版）

先背全项目通用的纪律红线：**(1) 原始数据/帧/视频不上云盘、不进在线 AI（保密!）**；**(2) 主线 `code/` 只读不改**——改超参只改 `tasks/u11_train/` 下的配置副本；**(3) 不直接联系甲方**。本组再加三条：**(4) 训练闸门（epoch 数、启动时机）必须经小助手批准，不得自行开长训**——闸门保护的是公共显卡时段，也是你自己的成绩；**(5) 没做零样本基线就开微调 = 对照失效**，该节点返工；**(6) 拿训练 loss 冒充验证 EPE= 造假**，一经查实该节点成绩作废。

防 AI 糊弄指南（第 11 组版）

(1) AI 产出必须实跑验证——第四条纪律红线：它写的启动脚本、给的配置改法，只有你跑出来的日志说了算。**(2) AI 爱报“最佳超参”**——那是别人数据集上的经验，你的数据要自己扫，抄来的最优往往是你的最差。**(3) AI 说“loss 降了就是训好了”不可信**——不看验证集的回答一律存疑，训练 loss 降、验证 EPE 升是常见的假繁荣。**(4) AI 贴的命令在你的环境里可能路径对不上、版本对不上**——以你的 `conda list` 与目录结构为准。

按课程建议方案，实验一不计分，实验二至四各 10 分共 30 分，平时作业 20 分（十二讲章后习题十二选二）；期末成果满分 50 分（教师可调）。成果怎么拿满，看下面这张 50 分制细则。

表 11.5　第 11 组成果满分标准（50 分制）

分值	项目	怎么拿满
20	功能达成	实验一配置说明书 + 逐项核对（3 分）；实验二数据加载 + 前向/反向冒烟（5 分）；实验三零样本基线 + 闸门内短程微调 + 对照曲线（10 分）；实验四 1–2 项超参 2–3 组对比（2 分）
10	证据完整	四次实验证据包三件套齐全、可复现；缺一件扣 5 分
8	报告质量	组报告按模板 8–15 页，对照结论有基线数据支撑、曲线图齐全
7	协作与台账	闸门申请与批准记录齐全；4080 时段台账无缺失；求助记录规范
5	加分	如：数据增强消融实验；按浑浊度分层训练子集并给出 EPE 对比；冻结主干只调局部的对照

存档点思维：checkpoint 是你的“读档”

游戏打 boss 前先存档，训练同理：每个 epoch 存一次 checkpoint（权重存档点）。断电、爆显存、超参试错，都能从最近的存档读档重来，而不是从零再训。实验三要求逐 epoch 存档，就是给你们上的这份保险。

11.7　四次实验安排

四个节点与全班同频（实验一 2026 年 10 月 12 日、实验二 11 月 2 日、实验三 11 月 23 日、实验四 12 月 21 日），每场都有本组的命令级动作。先看总览表，再逐场展开；每场按“目标—步骤—验收证据—常见坑”执行：步骤照抄就能跑，证据一栏就是成绩。总则：实验一至三是离线自研——资产包开局就发到每组手里，谁也不用等谁；实验四是合并联调——四条赛道的成果在同一台机器、同一套考卷上会师。

表 11.6　第 11 组四次实验总览

节点	日期	本组动作	核心验收证据
实验一	10月 12 日	agent 逐项解释训练配置，人工核对成“配置说明书”	说明书 + 核对记录
实验二	11 月 2 日	一个 batch 过模型，前向/反向冒烟	形状打印 + loss + 显存快照
实验三	11 月 23 日	零样本基线 + 闸门内短程微调对照	基线数字 + 批准记录 + 曲线
实验四	12 月 21 日	选 1–2 项超参做 2–3 组对比	配置快照 + EPE 对比表

11.7.1　实验一（10 月 12 日）：把 yaml 读成人话

目标：用 agent 逐项解释 `configs/train_uw_fresh.yaml`（lr/bs/iters/epoch），人工核对后写成“配置说明书”——训练前先读懂每一个要拧的钮。

步骤：(1)把配置文件拷贝到 `tasks/u11_train/`（主线只读，改也只改副本）；(2) 让 agent 逐项解释：每个键控制什么、默认值多少、往大往小各会怎样；(3) 逐项与代码核对——在主线代码里找到读取该键的那一行，行号写进说明书，才算核对通过；(4) 给每项配一句人话（如 lr 是“步子迈多大”，bs 是“一口吃几对图”）；(5) 汇成 1–2 页说明书。

验收证据：配置说明书 + 逐项核对记录（含代码行号与“对/错/存疑”标注）。

常见坑：AI 把 iters 和 epoch 混为一谈——一个数的是“总共走多少步”，一个数的是“全量数据走几遍”，两者关系要自己在说明书里算清；只解释不核对——AI 的解释看着都对，一对代码就露馅；说明书抄成名词字典，没有“动了会怎样”的判断。

11.7.2　实验二（11 月 2 日）：让一个 batch 走完全程

目标：跑通数据加载（UWStereo一个 batch 过模型）+ 前向/反向冒烟——确认“数据能进、loss 能算、梯度能传”，为实验三扫清路障。

步骤：(1) 在第 10 组环境里激活 conda 环境；(2) 数据加载冒烟：取一个 batch，打印左目/右目/视差真值的形状与数值范围；(3) 前向一步：输出视差形状与初始 loss 记进日志；(4) 反向一步：优化器 `step()` 成功、loss 可反传、显存占用正常（`nvidia-smi` 拍快照）；(5) 全程输出存档。

验收证据：batch 形状打印、初始 loss 数字、反向与优化器更新成功日志、显存快照。

常见坑：数据路径大小写或后缀不对，dataloader 静默空转——“在训了”其实是空转，先确认日志里读到了多少对图；数据里有坏样本（全黑图、尺寸异常）导致 loss 为 NaN——找第 9 组要体检报告提前排查；忘了 `model.train()`，跑通的是推理不是训练。

11.7.3　实验三（11 月 23 日）：先立基线，再开微调（本组主力）

目标：复现“零样本 → UWStereo 微调”对照：短程微调（不超过 5 个 epoch，经小助手闸门批准）+ 零样本基线，对比验证 EPE，画出 loss/EPE 曲线。

步骤：(1) **先测零样本基线**：资产包里的预训练权重在第 9 组验证集上的验证 EPE，资产包还带了一版前期初步微调的基线模型 v0，一并测了当参照——这是对照的“赛前成绩”，没有它后面一切提升无从谈起；(2) 填闸门申请：epoch 数（不超过 5）、启动时段、配置快照，**经小助手批准后方可开训**；(3) 训练中盯 loss 曲线，每个 epoch 存一次 checkpoint；(4) 训完在同一验证集上测验证 EPE，与基线同表对比；(5) 画 loss 与验证 EPE 双曲线，标出最优 checkpoint。

验收证据：基线 EPE 数字、闸门批准记录、完整训练日志、基线/微调对比表、双曲线图。

常见坑：**没做零样本基线就开微调**——“提升 0.4”从此无人能证，流程上先基线后微调，顺序不能反；**epoch 超闸门**——多训的每一分钟都是借全队的，超闸门按违纪处理；验证集混入训练场景——开工前拿第 9 组的划分清单逐条核对；中途断连没存 checkpoint，只能从头再来。

11.7.4　实验四（12 月 21 日）：一次只动一个旋钮

目标：在学习率、主干 lr 比例、数据增强中选 1–2 项，做 2–3 组对比实验，回答“哪个设置更适合水下数据”——本场同时是四组合并联调：采纳第 9 组新配比数据续训、与第 12 组统一口径总评测，成果进《四组交接总表》。

步骤：(1) 选 1–2 个旋钮（如 lr 减半对加倍、开/关颜色抖动增强）；(2) 每组配置存快照并固定随机种子，**其余参数与实验三完全一致**；(3) 各组走同样的闸门流程跑短程微调，分别记验证 EPE；(4) 填对比表：配置差异、基线 EPE、各组 EPE、训练耗时，写不少于 100 字的结论——哪个好、好多少、你猜为什么。

验收证据：各组配置快照与训练日志、EPE对比表、结论段。

常见坑：一次动两个旋钮——差异说不清归谁，对比作废；没固定随机种子——随机波动比超参差异还大，白比；贪多开四组对比——一组都没跑完，踏实的两组结论比仓促的四组值钱。

第 12 组　水下双目 · 评测组

本章地图

讲什么：第 12 组任务书——“指标体系 → 误差分桶 → 误差可视化 → 阈值扫描”四步赛道怎么走：EPE/bad-N 指标怎么造准、怎么给全队出成绩单、怎么找模型的软肋。**为什么值得读**：训练组说“我微调好了”，数据组说“我的配比更优”，部署组说“FP16 更快”——谁说了算？你说了算，拿数字说话。**学完你能**：定义并亲手验证 EPE/bad-N/D1 指标，搭一条“推理 → 视差 → 指标”的一键评测链路，用分桶归因找出模型最差的那一块短板。

一句话概括你们组：全项目的裁判组。裁判的公信力只来自一样东西：**口径统一、可复现**——同一把尺子量所有人，量几次结果都一样。你们给出的基线数字表就是全队的成绩单，你们找出的最差桶就是下一轮改进的路标。

12.1　组档案

表 12.1　第 12 组档案

项目	水下双目（项目二，中海油 HYSY123 FPSO 锚链 ROV 检查视频的双目立体感知，详见第 二 章）
组号与组名	第 12 组 • 评测组
人数与投入	5 人；每周合计 6–8 小时（组内自行排班，建议每周固定 2 次）
赛道	EPE/bad-N 指标 → 误差分桶（浑浊度/距离）→ 误差可视化 → 阈值扫描
上游依赖	开局资产包：零样本基线与初步模型 v0（第一周就有两版可评）；第 10 组推理链路；第 9 组验证集与划分清单
下游依赖	全队：基线数字表是成绩单；阈值 τ 推荐值回灌伪标签过滤；实验四联调出四组总成绩单，最差桶归因交第 9/11 组
工作目录	`tasks/u12_eval/`（主线 `code/` 只读，见防糊弄指南）
硬件要求	批量推理需 4080 时段（向第 10 组预约）；指标计算与画图用 CPU 电脑即可

先说清楚你们与别的组的本质区别：别的组交付“东西”（数据集、环境、权重），你们交付“判断”。判断要服人，必须满足两条——**口径固定**（分辨率、子集、随机种子全部写进评测配置，交表时一并交出）与**可复现**（任何人按你的配置重跑，数字必须对得上）。求助通道照旧：组内互助 → 研究助理 → 项目负责人，逐级上升，求助必须附完整命令与完整输出。

12.2　项目背景与你的位置

第 二 章讲过：锚链 ROV 检查视频靠双目立体匹配变成三维信息，方案是 FoundationStereo 教师–学生伪标签闭环。注意“过滤”两个字的分量——教师打的标签不是全收，要按置信分过一道闸门，闸门的松紧（阈值 τ）就是你们的实验四要定的参数。你在流水线的终点，也是反馈的起点：向

终点方向，你给零样本与微调两版权重出成绩单，让第 11 组的劳动被量化；向起点方向，你的 τ 推荐值与最差桶归因回流给数据组和训练组，驱动下一轮改进。裁判失职的代价是全队迷路：指标造错了，后面所有人的“提升”都是幻觉。

12.3　核心任务

四步赛道一张表，随后说人话。

表 12.2　第 12 组四步赛道：干什么与产出什么

阶段	干什么	产出
指标体系	定义并实现 EPE、bad-1/2/4px、D1，手工验证 1 对样例	指标白皮书
全链路	跑通“推理 → 视差输出 → 指标计算”，串成一键命令	一键评测脚本
系统评测	微调权重 vs 零样本，每模型评不少于 50 对	基线数字表
扫描与归因	阈值 τ 扫描画保留率–精度曲线；按浑浊度/视差分桶找最差桶	扫描报告 + 归因

先把尺子造准（三个指标见下面的知识盒），再跑链路，再系统评测，最后干裁判最有价值的事——**指出模型差在哪**。为什么非要分桶？看下图：视差越大物体越近，远处的锚链视差小、误差天然大——如果只报一个总 EPE，近处的好成绩会把远处的灾难平均掉，**总分会撒谎，分桶才见真章**。按视差大小（近/中/远）与浑浊度（低照度、浑浊代理指标）分桶之后，“最差桶”三个字才有着落——实验四的人工归因就盯住它。

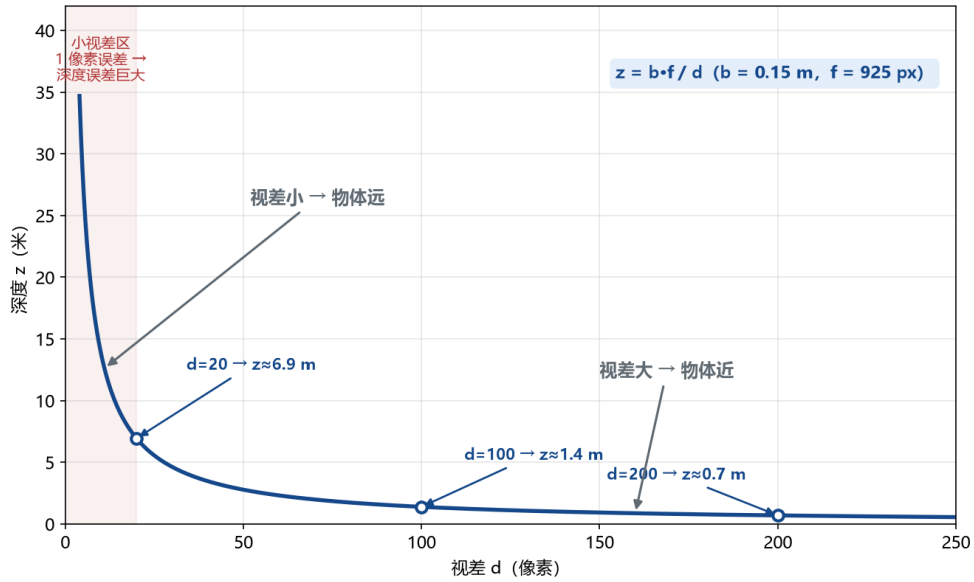


图 12.1　视差与深度成反比：视差越大物体越近——这正是评测要按视差分桶的道理，远处视差小、天然难，必须单独看

三把尺子把“差多少”说清楚

EPE（End-Point Error，端点误差）：所有像素上“预测视差与真值之差的绝对值”取平均，单位是像素，越小越好——这个方向性是全组成绩单的生死线，永远别写反。**bad-N**：误差超过 N 个像素的像素占比，bad-1/2/4px 就是 N 取 1、2、4，管的是“错得明显”的像素比例。**D1**：误差既超过 3 个像素、又超过真值 5% 的像素占比，专抓“错得离谱”的硬伤。四个数一起看：EPE 管平均，bad-N 与 D1 管最差的那部分像素。

项目连线：评测组的活与第一篇哪几讲有关

评估方法论出自第 3 讲——训练集/验证集的划分、泛化与过拟合、“指标方向性”这些评估的底层常识，那一讲全都打过样；你们是它在立体匹配任务上的执行人。EPE 算的是逐像素误差的平均，数学上就是那一讲的平均损失思想；“分桶看指标”则是它的分层版——平均会撒谎，分布不会。

12.4　第 19 周周五提交清单

提交线：2027年 1 月 22 日（第 19 周周五）完全提交离场。你的组交齐下表各件，每样都与第 三 章的总清单模板对应；《四组交接总表》由水下四组共填，本表最后一行就是总表里你组的“交接内容”栏。

表 12.3　第 12 组提交清单（2027 年 1 月 22 日）

交付物	格式	说明
指标白皮书	PDF，2–4 页	EPE/bad-1/2/4px/D1 的定义、公式、算法与出处；1 对样例的手工验算记录
基线数字表	CSV + 表格	零样本 vs 微调 v1，每模型评不少于 50 对：总分 + 分桶数字 + 评测配置
阈值扫描报告	PDF，2–4 页	阈值 τ 扫描的保留率–精度曲线、推荐 τ 与理由；最差桶归因结论
误差可视化集	图册	不少于 20 张误差可视化图，每张附人工归因“错在哪”
小组项目报告	PDF，8–15 页	按第 三 章模板：摘要/背景/方法/实验/结论/分工声明
证据包	zip	四次实验各一包，命令 + 日志 + 产物三件套，日志为终端完整输出
AI 使用声明	一页	哪些事 AI 做了、人做了什么核验，逐条列清
交接内容	《四组交接总表》一行	基线数字表交全队与教师；推荐 τ 交伪标签过滤环节；最差桶归因交第 9/11 组

12.5　AI 分工与人工必做

分工原则一句话：AI出草稿，人出签字——成绩单上每一个数字、每一句归因，签字的人负全责。

表 12.4　第 12 组 AI 分工与人工必做

AI可以自动做	人必须亲自做
起草指标白皮书的文字与公式排版	**看误差可视化图不少于 20 张并写“错在哪”的人工归因（AI 不可代看）**
写指标计算脚本与分桶统计脚本草稿	挑 1 对样例手工验算 EPE，与脚本数字对账
画曲线、热图与误差可视化图	固定随机种子，核对分桶结果可复现
解释异常数字（为何这一桶 EPE 特别高）	逐条核对 AI 公式与代码实现一致
建议阈值扫描的档位设计	在基线数字表上签字，对每个数字担责

12.6　防糊弄指南与满分标准

防人工糊弄指南（第 12 组版）

先背全项目通用的纪律红线：**(1) 原始数据/帧/视频不上云盘、不进在线 AI（保密！）**——评测用的甲方帧同样涉密，出门的只有统计数字；**(2) 主线 `code/` 只读不改**，评测脚本与产出都放 `tasks/u12_eval/`；**(3) 不直接联系甲方**。本组再加三条：**(4) 只报总 EPE 不报分桶 = 没评完**——总分会被简单样本撑好看，裁判不能只报总分；**(5) 中途更换评测口径（换分辨率、换子集）不声明 = 成绩单作废**；**(6) 代跑必须留运行者姓名与时间**，谁跑的谁负责。

防 AI 糊弄指南（第 12 组版）

(1) AI 产出必须实跑验证——第四条纪律红线：它给的指标实现、算出的任何数字，只有你脚本跑出来的那份算数。**(2) AI 写指标公式常有下标与符号幻觉**——逐条与你核对过出处的白皮书对账，公式的每个绝对值、每个平均范围都要有着落。**(3) AI“帮你看图”写的归因只能当候选答案**——必须你亲眼看图确认后才可采用，AI 不可代看是本组铁律。**(4) AI 报的指标数必须与你的脚本输出一致**，对不上以脚本为准、以你的终端为准。

按课程建议方案，实验一不计分，实验二至四各 10 分共 30 分，平时作业 20 分（十二讲章后习题十二选二）；期末成果满分 50 分（教师可调）。成果怎么拿满，看下面这张 50 分制细则。

表 12.5　第 12 组成果满分标准（50 分制）

分值	项目	怎么拿满
20	功能达成	实验一指标白皮书 + 手工验算对账成功（5 分）；实验二“推理 → 视差 → 指标”一键链路（5 分）；实验三每模型不少于 50 对的基线数字表（含分桶）（8 分）；实验四 τ 扫描曲线 + 最差桶归因（2 分）
10	证据完整	四次实验证据包三件套齐全、可复现；缺一件扣 5 分
8	报告质量	组报告按模板 8–15 页，评测口径交代清楚、结论有基线与分桶数据支撑
7	协作与台账	为三组提供统一评测口径并有记录；4080 时段台账无缺失；求助记录规范
5	加分	如：自动浑浊度分桶的量化指标被数据组采用；评测封装成一条命令被三组复用；提出并论证一个新指标

12.7　四次实验安排

四个节点与全班同频（实验一 2026 年 10 月 12 日、实验二 11 月 2 日、实验三 11 月 23 日、实验四 12 月 21 日），每场都有本组的命令级动作。先看总览表，再逐场展开；每场按“目标—步骤—验收证据—常见坑”执行：步骤照抄就能跑，证据一栏就是成绩。总则：实验一至三是离线自研——资产包开局就发到每组手里，谁也不用等谁；实验四是合并联调——四条赛道的成果在同一台机器、同一套考卷上会师。

表 12.6　第 12 组四次实验总览

节点	日期	本组动作	核心验收证据
实验一	10月 12 日	agent 写指标白皮书，实跑 10 对样例，手工验证 1 对	白皮书 + 验算记录
实验二	11 月 2 日	跑通“推理 → 视差 → 指标”全链路	一键脚本 + 批量日志
实验三	11 月 23 日	微调 vs 零样本系统评测，每模型不少于 50 对	基线数字表（总分 + 分桶）
实验四	12 月 21 日	阈值 τ 扫描 + 分桶归因	曲线 + 分桶表 + 20 张归因

12.7.1　实验一（10 月 12 日）：先把尺子造准

目标：用 agent 写“指标白皮书”（EPE/bad-1/2/4px/D1 的定义与算法），实跑 10 对样例算指标，并手工验证其中 1 对——尺子不准，后面全错。

步骤：(1) 让 agent 逐个指标写：定义、公式、取值范围、适用场景；(2) 逐条与主线代码及文献核对——公式里每个下标、每个绝对值、每个平均范围都要有出处；(3) 对 10 对样例实跑指标脚本，存全部输出；(4) 挑 1 对，把视差真值图缩小成 10×10，用计算器把 100 个像素的绝对误差加起来除以 100，与脚本算出的 EPE 对账；(5) 白皮书定稿，写明每条定义的出处。

验收证据：指标白皮书、10 对样例的指标输出、手工验算过程（可拍照存档）。

常见坑：**指标算错符号（EPE 越小越好）**——报告里把方向写反，整套结论全盘皆输；手工验算跳过——脚本错一个绝对值，往后所有数字都是错的，而没人知道；AI 把 D1 写成另一种变体定义——以你核对过的文献为准，白皮书里注明出处。

12.7.2　实验二（11 月 2 日）：一键打通“推理 → 视差 → 指标”

目标：跑通全链路：对 1 对图完成推理、保存视差、算出指标，再扩展成 10 对批量版，为实验三的大规模评测铺路。

步骤：(1) 在第 10 组环境里对 1 对 UWStereo 图跑推理，视差图存盘（float 格式）；(2) 把视差画成可视化色图，肉眼检查是否合理（锚链处视差应连续、不应有大片空洞）；(3) 对同一对算 EPE/bad-1/2/4px/D1，核对中间文件；(4) 把三步串成一条命令（脚本放 `tasks/u12_eval/`），一键完成“推理 → 保存 → 算分”；(5) 跑 10 对批量版，存完整日志。

验收证据：一键脚本、单对与批量日志、视差可视化样例。

常见坑：视差图存成 uint8 被截断——大于 255 的视差全变 255，指标全错，存 float 才稳；左右目路径接反——视差符号全反，EPE 大得离谱反而“提醒”了你；可视化用非线性配色，小误差看着触目惊心，干扰后面的人工归因。

12.7.3　实验三（11 月 23 日）：两版权重同场比武，每模型不少于 50 对

目标：对微调权重 vs 零样本权重做系统评测，每模型评不少于 50 对，建立全队基线数字表——这是你们给全队出的成绩单。

步骤：(1)从资产包取验证集划分与两版权重（零样本基线 + 前期初步模型 v0——开局即评，不必等人）；第 11 组联调新版本实验四补测；(2) 固定随机种子与评测口径，分辨率、对齐方式、子集名单全部写进评测配置并随表交出；(3) 每模型评不少于 50 对，记 EPE/bad-1/2/4px/D1 总分；(4) 按场景与视差范围分桶统计；(5) 汇成基线数字表，并给每版权重写一句“体检结论”。

验收证据：评测配置、两模型批量日志、基线数字表（总分 + 分桶 + 体检结论）。

常见坑：**分桶没固定随机种子**——每次分桶结果都不一样，两次报告数字对不上，裁判自己先打架；两模型用了不同分辨率——同场比武变成各跑各道，对比无效；50 对里混进训练集场景——拿第 9 组的划分清单逐条核对；只报平均不报分桶——总分会撒谎。

12.7.4　实验四（12 月 21 日）：扫阈值、找最差桶、当个好裁判

目标：伪标签过滤阈值 τ 扫描（保留率–精度曲线）；按浑浊度/视差大小分桶，找出最差桶并归因——这是你对伪标签闭环最直接的贡献；本场同时是四组合并联调：补测第 11 组新权重、统一口径出四组总成绩单，成果进《四组交接总表》。

步骤：(1) 对伪标签置信分扫描 τ：每档记保留率与精度（在有真值的子集上算 EPE）；(2) 画保留率–精度曲线，给出推荐 τ 与理由——太松，噪声标签混进训练集；太紧，粮食不够吃；(3) 按浑浊度代理指标与视差大小（近/中/远）分桶，找出 EPE 最差的桶；(4) **看不少于 20 张误差可视化图，人工写“错在哪”归因（AI 不可代看）**；(5) 归因结论与推荐 τ 交给第 9/11 组与伪标签过滤环节。

验收证据：τ 扫描表与曲线、分桶数字表、不少于 20 张的人工归因记录。

常见坑：归因写成“效果不好”四个字——合格的归因要具体到“远距离 + 低照度 + 细杆状结构”这一档；曲线只画不选——没有推荐 τ 等于没扫；老毛病再犯：符号方向与随机种子，本组两大惯犯，交表前组内互查一遍。

第 13 组　数据基座 · 数据结构化组

本章地图

讲什么：第 13 组任务书——把几份含表格与公式的工程文档 PDF，做成“机器可查”的结构化 JSON：表格抽成表、公式转成 TeX、每个元素登记锚点，外加一份敢说真话的质量报告。**为什么值得读**：真实系统 1,001 页拆出 484 张表、19,893 个锚点，靠的就是结构化流水线；你做的是它的教学版——规模缩小、方法论一份不打折，而且你的产物是下游问数组的口粮。**学完你能**：独立跑通“PDF 进、JSON 出”的小流水线；说得清锚点与“如实条款”为什么是命根子；亲手训一个“这块是表格、公式还是正文”的分类小模型。

一句话概括你们组：整条数据基座流水线的“拆迁办”。下游各组都在“新房”里干活——问数、查图谱、抓造假——而你们负责把 PDF 这栋老房子拆成登记造册的梁、柱、管线：表格是表格、公式是公式，每块碎片都带“户口”。拆迁不干净，下游全喝西北风。

13.1　组档案

表 13.1　第 13 组档案

项目	数据基座治理（项目三，工程资料结构化与本地数据查询系统，详见第 三 章）
组号与组名	第 13 组 • 数据结构化组
人数与投入	5 人；每周合计 6–8 小时（组内自行排班，建议每周固定 2 次）
负责的能力	B 类需求（文档结构恢复）的教学版——文档结构化小流水线
上游依赖	项目提供的样例包：3–5 份含表格与公式的工程文档 PDF 片段
下游依赖	第 14 章（第 14 组）：一句话问数 demo 吃你的结构化产物——中期前他们用演示数据兜底，你交付后无缝切换
工作目录	`tasks/d01_struct/`（样例 PDF 放 `data/`，产物放 `out/`，脚本放 `src/`）
硬件要求	普通笔记本 CPU 即可：解析库吃 CPU，实验三的小模型只有几百个参数，无 GPU 毫无压力

先解释一个词：**教学版**。真实系统的 B 类能力是工业级的——跨页表格合并、公式 TeX 源码、19,893 个锚点（第 三 章的数字）；你们做的是从它派生的缩小版：样例只有几份 PDF，流水线只跑几十页，但**每一个设计决策都和真实系统同构**——抽不出来怎么办？登记回退。抽出来没把握怎么办？如实标注。这套纪律练熟了，换到任何一个真实项目都能直接上手。另外把求助通道抄在工位上：组内互助 → 项目助教 → 导师，逐级上升；求助必须附完整命令与完整输出，只发一句“跑不通”等于没发。

13.2　项目背景与你的位置

第 三 章讲过来龙去脉：甲方的资料分两座山——126 份 CSV 共 3.1 亿行的运行数据是“洪水”，6 份共 1,001 个物理页的工程资料是“迷宫”。洪水难在量，迷宫难在结构：表格动辄跨页、公式不是图片就是扫描件、PDF 的逻辑页码和物理页还对不上。系统对这座迷宫下的第一步功夫，就是你们对口的 **B 类需求：文档结构恢复**——把 1,001 页还原成“表格是表格、公式是公式、每一段都能指回原文”的结构化数据，最终交出 484 张表、272 段公式源码、19,893 个锚点，并且大大方方登记了 39 处回退区域、138 张列宽证据不足的表、“人工确认计数 0”。

你在流水线的最上游。看下图，四条链里最左边那条“结构化”就是你们的工地：文档从你们手里变成带锚点的结构化数据，问数链才有 SQL 可查，门控链才有知识可抽，下游的每个数字才能“点回原件”。第 三 章缺口表上属于你们的两行——39 个局部回退区域、138 张列宽证据不足的表——就是这学期要正面强攻的城墙。

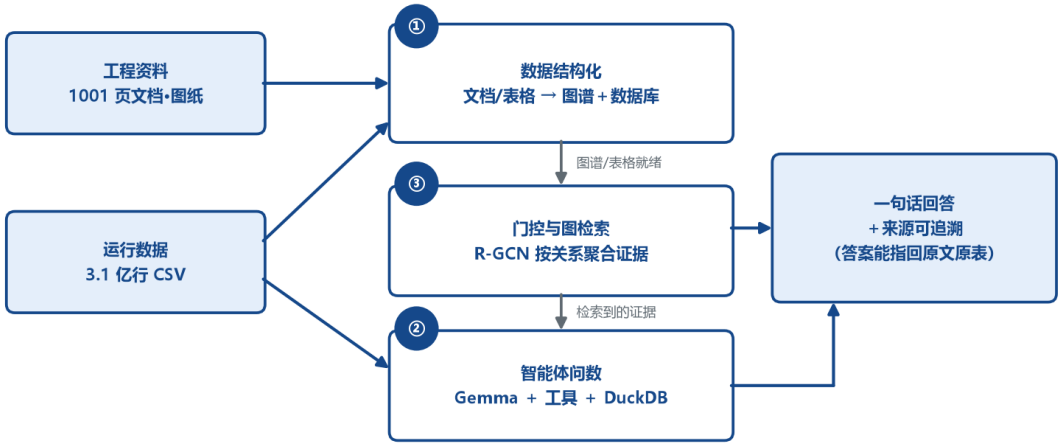


图 13.1　数据基座三大能力流水线上的第 13 组：你们站在最上游的“结构化”一环，把文档与表格变成带锚点的结构化数据——下游的智能体问数、门控与图检索、攻防，全都踩在你们打的地基上

13.3　核心任务

四件事一张表看明白，随后逐件说人话。

表 13.2　第 13 组四项任务：干什么与产出什么

任务	干什么	产出
表格抽取	用解析库把每页表格抽成行列，跨页表格登记“可拼接线索”，合并单元格关系尽量保留	表格 CSV/JSON
公式转 TeX	找出公式区域，转写成 LaTeX 片段（数学式子可复制、可检索的那套记号）	公式片段
锚点登记	每个元素登记“哪份文件、第几个物理页、页面哪个区域”的出处坐标	锚点表
质量报告	抽了多少表、多少公式、多少回退区域、多少存疑，**如实登记**，学真实系统的“如实条款”	质量报告 PDF

逐件展开。**表格抽取**是主力活：`pdfplumber` 靠文字坐标与线框认表，`PyMuPDF` 快而稳，`camelot` 对带线框的表更准——三选二装通即可（实验二细说）。抽完不是终点：跨页的表会被拆成两截，你至少要登记“本表在第 37 页结束、第 38 页有同列数的疑似延续”，能不能拼回来如实说明。**公式转 TeX**：样例里的公式先框出区域，简单式子可用规则或在线大模型转写，转写结果必须人工抽核——大模型最爱在上下标上幻觉。**锚点登记**是全组的地基，格式看下面这段产物示例。**质量报告**是本组的灵魂：真实系统“人工确认计数 0”都敢写，你们抽错几张、哪几页抽不出来，也要一张不落地说清楚。

```
{
  "element_id": "T-0037",
  "type": "table",
  "source_file": "sample_calc_book.pdf",
  "physical_page": 37,
  "bbox": [84.2, 120.5, 511.6, 702.9],
  "caption": "表 3-2 系泊缆索设计参数",
  "n_rows": 12, "n_cols": 5,
  "merged_cells": [[2, 0, 2, 1]],
  "cross_page_hint": "same_cols_continue_p38",
  "confidence": 0.86,
  "flag": "column_width_uncertain"
}
```

逐字段念一遍：`physical_page` 是物理页（不是目录里的逻辑页码，封面目录造成的错位在这里不起作用）；`bbox` 是页面坐标，四个数圈出这张表在页面上的位置——锚点的“户口地址”；`merged_cells` 记合并单元格；`confidence` 与 `flag` 就是“如实条款”的落点：机器没把握的，如实写出来，不许悄悄装没问题。下游看到 `flag` 就知道这张表用之前该查什么——**敢标存疑，别人才敢用**。

项目连线：拆房子的活与第一篇哪几讲有关

第 4 讲的“表示”思想：版面上的表格与公式，最终都要变成机器好算的格式——JSON 与向量一样，都是“表示”；第 6 讲的训练循环：实验三你训元素分类小模型时，取批、前向、算损失、反向、更新，一步不少；第 11 讲的注意力与在线模型 API：实验四的疑难表格二次抽取，本质是提示词工程 +few-shot，正是大模型 API 的标准用法。

13.4　第 19 周周五提交清单

提交线：2027年 1 月 22 日（第 19 周周五）完全提交离场。你的组交齐下表七样，每样都与第 三 章的总清单模板对应。

表 13.3　第 13 组提交清单（2027 年 1 月 22 日）

交付物	格式	说明
结构化流水线代码包	zip，含 README	环境说明 + 一键脚本，别人拿到能跑；三选二解析库的选型理由写进 README
结构化产物与锚点表	JSON/CSV	样例包全部 PDF 的产物：表格、公式片段、锚点表，字段按上一节示例齐全
质量报告	PDF，2–4 页	含如实条款：抽了多少表/公式、多少回退区域、多少存疑、人工核对计数，如实登记
分类小模型与曲线	模型文件 + 图	实验三的“表格/公式/正文”分类器、loss 曲线、混淆矩阵
小组项目报告	PDF，8–15 页	按第 三 章模板：摘要/背景/方法/实验/结论/分工声明
证据包	zip	四次实验各一包，命令 + 日志 + 产物三件套，日志为终端完整输出
AI 使用声明	一页	哪些事 AI 做了、人做了什么核验，逐条列清

13.5　AI 分工与人工必做

分工原则一句话：AI出草稿，人出签字——质量报告里每一个数字，签字的人负全责。

表 13.4　第 13 组 AI 分工与人工必做

AI可以自动做	人必须亲自做
写解析与批处理脚本草稿	**逐页核对 ≥20 页（AI 不可代看）**
生成标注建议与预标结果	**标注 ≥200 条终审**：AI 可预标，人逐条过目签字
解释报错信息、给排查建议	本机实跑每一条命令
画统计图、起草报告与质量报告文字	逐条核对 AI 结论（在代码里找到对应行才算数）
建议提示词与 few-shot 样例写法	在产物与质量报告上签字，对数字担责

13.6　防糊弄指南与满分标准

防人工糊弄指南（第 13 组版）

(1) 核对记录不许复制粘贴凑数——“逐页核对 20 页”就是 20 个不同的物理页，同一页抄十遍不行。**(2) “抽查合格”必须写清抽查了哪几页**——页码区间、核对结论、发现的问题，三样缺一不可；只写“抽查合格”四个字等于没查。**(3) 编造抽取数量 = 该节点成绩作废**——质量报告里的每个数字都要能与 JSON 产物对上账，对不上就是作废，没有商量。**(4) 代跑必须留运行者姓名与时间**，谁跑的谁负责。

防 AI 糊弄指南（第 13 组版）

(1) AI 说“抽取成功”不算数——成功不成功只看 JSON 内容：行数、列名、数值，打开文件逐项对。**(2) 大模型幻觉的表头必须人工纠错**——它会给缺列名的表“好心”补一个，补得头头是道，原文里根本没有。**(3) AI 报的页码与坐标要抽验**——它没真读过你的 PDF，位置信息十有八九是编的。**(4) 禁止把 AI 的回答直接粘进质量报告**——报告里只许出现你亲手核对过的内

容。

按课程建议方案，实验一不计分，实验二至四各 10 分共 30 分，平时作业 20 分（十二讲章后习题十二选二）；期末成果满分 50 分（教师可调）。成果怎么拿满，看下面这张 50 分制细则。

表 13.5　第 13 组成果满分标准（50 分制）

分值	项目	怎么拿满
15	流水线功能	实验二 10 页流水线跑通且产物字段齐全（8 分）；实验四疑难表格二次抽取并出对比报告（7 分）
10	质量报告与如实度	每个数字与产物对得上账；回退、存疑、人工核对计数如实登记，无一件虚报
8	小模型训练	标注 ≥200 条且终审留痕；loss 曲线齐全；混淆矩阵与宏平均有分析
7	报告质量	组报告按模板 8–15 页，方法讲得清、图表齐全、结论有数据支撑
5	协作与台账	排班执行有记录；标注分工与终审留痕；求助记录规范
5	加分	如：跨页表格自动合并逻辑并用数字证明有效；为扫描件接 OCR 兜底并对比效果；把流水线做成一键复跑工具

13.7　四次实验安排

四个节点与全班同频（实验一 2026 年 10 月 12 日、实验二 11 月 2 日、实验三 11 月 23 日、实验四 12 月 21 日），但每场都有本组的命令级动作。先看总览表，再逐场展开；每场按“目标—步骤—验收证据—常见坑”执行：步骤照抄就能跑，证据一栏就是成绩。

表 13.6　第 13 组四次实验总览

节点	日期	本组动作	核心验收证据
实验一	10月 12 日	agent 解读表格抽取原理，3 份样例各抽一页人工核对	解读 + 核对记录一页
实验二	11 月 2 日	解析环境三选二装通，10 页流水线跑通	版本输出 + 日志 +JSON
实验三	11 月 23 日	自造标注 ≥200 条，训元素分类小模型	训练日志 +loss 曲线
实验四	12 月 21 日	在线 API 疑难表格二次抽取，对比规则法	提示词 + 对比表 + 账单

13.7.1　实验一（10 月 12 日）：让大模型讲原理，你拿样例当裁判

目标：配好大模型编程助手的 agent 模式，让它讲清 `pdfplumber` 表格抽取的原理，你用 3 份样例 PDF 各抽一页，人工核对它说得对不对、抽得准不准。

步骤：(1) 按课堂演示装好编程助手并切换到 agent 模式；(2) 安装并验证：`pip install pdfplumber`，再跑 `python -c "import pdfplumber; print(pdfplumber.__version__)"`；(3) 让 agent 回答三问：`extract_tables` 是靠什么把一张表从页面里“认”出来的（线框？文字对齐？）、跨页表格为什么会被拆开、什么情况下会抽错；(4) 写一个十行小脚本对 3 份样例各抽一页，结果存 JSON；(5) 打开 PDF 逐格对

照，标注“对/错/漏”，并把 agent 的解释逐条与源码或官方文档对照，至少纠正它一处错误。

验收证据：一页“AI解读 + 人工核对”记录（含 AI 原话、你的核对结论、至少一处纠正），加 3 页抽取结果与逐格核对记录。

常见坑：扫描件让 pdfplumber 全军覆没——页面上没有文字层，`extract_tables` 返回空列表，先确认样例是文字版，扫描件要留到 OCR 兜底（可做加分项）；agent 会“脑补”原理——它讲的算法细节可能来自教科书而不是这个库，一切以源码为准；只数抽了几行不看列名对不对——列名错了，整张表都是错的。

13.7.2　实验二（11 月 2 日）：解析环境装通，10 页流水线跑通

目标：本地部署 Python 解析环境（PyMuPDF / pdfplumber / camelot 三选二装通），批量跑通 10 页的“抽取—锚点—登记”流水线，产出第一批结构化 JSON。

步骤：(1) 建独立虚拟环境，避免污染；(2) 三选二安装：`pip install pymupdf pdfplumber`，想上 camelot 再装 `pip install "camelot-py[cv]"`——它依赖 Ghostscript，装不上就果断放弃，三选二达标即可；(3) 写 `extract_pipeline.py`：输入一个 PDF，逐页跑“表格抽取 → 公式区域记录 → 锚点登记”，输出上一节格式的 JSON；(4) 对含 10 页的样例跑通全流程；(5) 打印统计：几张表、几段疑似公式、几条锚点、几个回退区域。

验收证据：两个库的版本号输出、10 页流水线的完整终端日志、JSON 产物路径。

常见坑：中文路径报错——部分库对非 ASCII 路径支持不佳，样例统一放英文相对路径（如 `data/sample1.pdf`）最省事；camelot 卡安装一整晚——三选二就是给你留的后路，别跟依赖较劲；跨页表格被拆成两张——先如实登记“可拼接线索”，不强行合并，自动合并是加分项不是门槛。

13.7.3　实验三（11 月 23 日）：自己造数据，训一个“认版面”的小模型

目标：用你自己造的标注数据（≥200条）训练一个“这块是表格/公式/正文”的三分类小模型，亲眼看 loss 下降——这是“深度网络的模型训练实践”在本组的落点。

步骤：(1) 造数据：把样例 PDF 切成块（如按行块），给每块打标签，写一页标注规范（什么算公式块、跨页怎么算）再动手；AI 可预标，人逐条终审，凑够 200 条且三类都有像样的样本；(2) 造特征：块高、字号、数字密度、线框数等十几维手工特征；(3) 训一个两层小网络（第 2 讲的反向传播就够了），训练/验证**按页切分**——同一页的块不许一边进训练一边进验证；(4) 记录每个 epoch 的 loss，画曲线；(5) 算验证集总体准确率之外，再算宏平均与混淆矩阵，看哪两类最容易混。

验收证据：标注集 + 标注规范一页、训练日志、loss 曲线、混淆矩阵与宏平均。

常见坑：类别不平衡——正文块远多于公式块，只看总体准确率会被“全判正文”骗过，宏平均才见真章；特征量纲不一忘了归一化，loss 降得莫名其妙地慢；两人标注打架——先写规范、抽 20 条交叉核对一致性，再批量标。

13.7.4　实验四（12 月 21 日）：在线大模型救疑难表格，规则法当对照组

目标：接入在线大模型 API，对规则法抽不动的疑难表格做“二次抽取”（提示词工程 +few-shot 样例），与规则法同题对比，输出对比报告与结论。

步骤：(1) 从实验二的产物里挑出抽错/抽不出的疑难表格 ≥5 张，登记“难在哪”（无线框？合并单元格？跨页？）；(2) 设计提示词：描述表格结构，配 1–2 个人工核对过的正例做 few-shot；(3) 调 API 抽取，结果逐格人工核对；(4) 填对比表：规则法 vs 在线增强法，各列“列名对/数值对/合并

关系对”的格数、耗时、花费；(5) 写不少于 100 字的结论：什么时候用规则、什么时候值得上在线 API。

验收证据：提示词全文、API调用记录（次数与花费账单）、对比表、结论段。

常见坑：API key 硬编码进代码包——用环境变量，交包前全文搜一遍 key；把整份文档发给在线 API——只发疑难页的文本或截图，能少发不多发；只报成功案例不报失败——在线法也有翻车的题，如实写进对比表，这份诚实正是你们组的看家本领。

第 14 组　数据基座 · 智能问答与图检索组

本章地图

讲什么：第 14 组任务书——两件事：搭一个“一句话问数”小 demo（本地小模型 + 演示库 + 白名单工具 +SQL 安全检查），再在一个 50–100 节点的小知识图谱上训两层 R-GCN 做链路预测。**为什么值得读**：真实系统 500 题回归考卷拿下 497 分的问数能力、缺口表上 G8/G9 两条（正式图谱未发布、R-GCN 未联合验收）都和你们对口——demo 虽小，“模型只提出 SQL，程序执行 SQL”的军规一条不少。**学完你能**：用 Ollama 跑通本地问数并录屏留证；说得清白名单与只读检查怎么防住 SQL 注入；独立完成一次小规模图神经网络训练并报出 AUC。

一句话概括你们组：数据基座流水线的“窗口柜员”兼“关系侦探”。柜员这半句：用户说人话，你们把人话变成 SQL，查出的每个数字都带出处；侦探这半句：把设备、部件、故障、文档连成一张网，让“A 和 B 什么关系”这种关键词检索答不了的题有据可查。

14.1　组档案

表 14.1　第 14 组档案

项目	数据基座治理（项目三，工程资料结构化与本地数据查询系统，详见第 三 章）
组号与组名	第 14 组 • 智能问答与图检索组
人数与投入	5 人；每周合计 6–8 小时（组内自行排班，建议每周固定 2 次）
负责的能力	A 类需求（一句话问数）与图检索的教学版——问数小 demo + R-GCN 链路预测小实验
上游依赖	第 13 章（第 13 组）的结构化产物；中期前用项目提供的演示数据兜底（四个查询模型的精简版 CSV，各几百行）
联动依赖	第 15 章（第 15 组）：其造假样本可进你们的测试题库（对抗联动，见核心任务）
工作目录	`tasks/d02_agent_kg/`（`demo/` 放问数服务，`kg/` 放图谱实验，`tests/` 放 30 题）
硬件要求	8GB 内存笔记本即可：1.5B–3B 的量化模型 CPU 能跑、有 GPU 更快；R-GCN 小图 CPU 可训

先解释“演示数据兜底”：第 13 组的结构化产物要到期中才交付，但你们的流水线一天都不能等——项目提供 `fpso_navigation` 等四个查询模型的精简版 CSV（各几百行，真数据的缩样）供你们先建库跑通；第 13 组产物到位后，同一套代码换个数据源就能切换。这套“先兜底、后切换”的打法，与真实系统“先沙箱、后生产”的节奏同源。求助通道照旧抄在工位上：组内互助 → 项目助教 → 导师，求助附完整命令与完整输出。

14.2　项目背景与你的位置

第 三 章讲过真实系统怎么答“2023 年 11 月 18 日 20:27 的尾吃水是多少”：大脑是本地部署的 Gemma 12B（上下文 8,192、温度 0、并发 1），手里只有 8 个白名单工具；它提出的每条 SQL 都要过 SQLGlot 检查、Wren 规划，最后由 DuckDB 执行——一句话军规：**模型选择工具，程序检查边界，数据库计算数值**。凭这套镣铐，500 题独立标准答案的回归考卷拿了 497 分。图检索那边，系统把知识连成图谱，用 R-GCN 按关系类型聚合邻居、给证据排序；但缺口表也如实写着：**G8**——正式知识图谱尚未发布；**G9**——R-GCN 尚未与业务数据联合验收。这两条就是你们的进场口。

你在流水线的中段，管两条链：**问数链**——人话进来、SQL起草、检查放行、数据库执行、答案带引用出去；**图检索链**——实体进来、沿关系扩邻居、模型排序、证据出去。看下图，真实系统把“理解问题 → 调用工具 → 核查依据 → 回答与来源”画成一个八步闭环，证据不足就回头重排步骤；你们的 demo 就是把这个闭环缩小装进笔记本——**环节一个不砍，规模全线下调**。

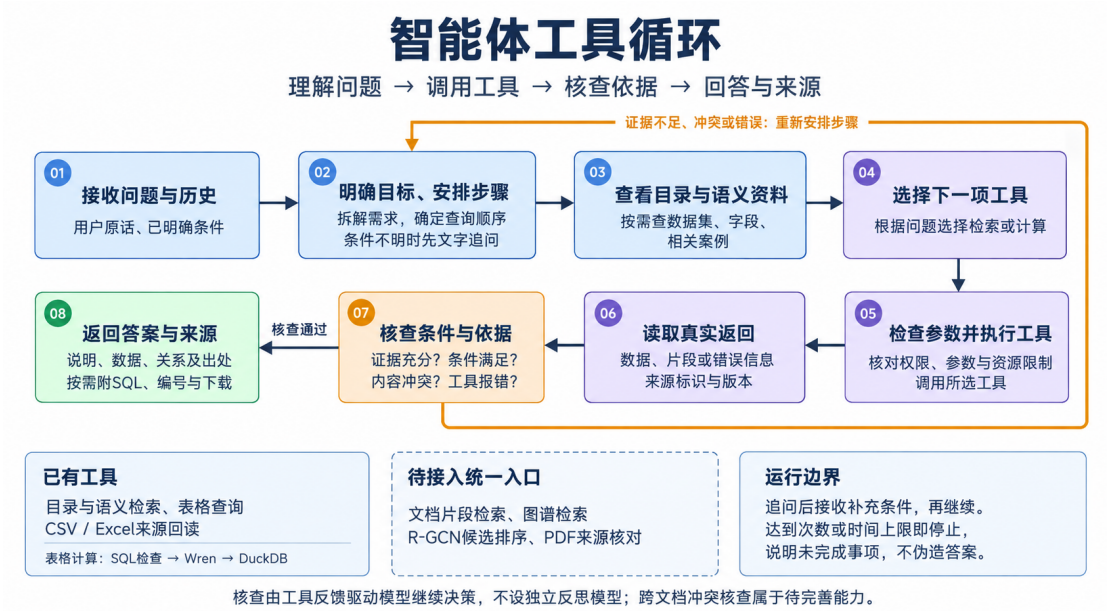


图 14.1　真实系统的智能体工具循环：理解问题 → 调用工具 → 核查依据 → 回答与来源的八步闭环，证据不足、冲突或报错就重新安排步骤；第 14 组的问数 demo 就是它的教学版——环节一个不砍，规模全线下调

图片来源：本项目《总体方案与验收计划》文档配图

14.3　核心任务

两块任务一张表看明白，随后逐块说人话。

表 14.2　第 14 组两块任务：干什么与产出什么

任务	干什么	产出
问数小 demo	本地小模型 + 演示库 +3 个白名单工具函数 +SQL 安全检查，一句话问数、答案带引用	可跑的 demo
R-GCN 实验	networkx 建 50–100 节点小图谱（设备-部件-故障-文档四类实体），训两层 R-GCN 做链路预测	notebook+ 指标

问数小 demo 四件套。第一件，**本地小模型**：用 Ollama 跑 Qwen2.5-1.5B（内存紧张就 1.5B，

宽裕可上 3B）或 Gemma2-2B，**温度设 0**——同一问题两次回答必须一致，这是第 三 章 A7 条款的教学版。第二件，**演示库**：SQLite 或 DuckDB 装下四个精简查询模型。第三件，**三个白名单工具函数**：`query_rows`（按条件查明细）、`agg_stats`（求和/平均/计数/最值）、`describe_columns`（列名与含义说明）——模型手里只有这三件，没有任意执行。第四件，**SQL 安全检查**：只读（拦下 INSERT/UPDATE/DELETE/DROP）、列名白名单（库里有这列才放行）；铁律再次上墙：**模型只提出 SQL，程序执行 SQL**——模型碰不到数据库，碰数据库的只有被检查过的语句。

R-GCN 实验。先用 networkx 造一张 50–100 个节点的小图：设备、部件、故障、文档四类实体，“设备–装有–部件”“部件–发生–故障”“故障–记载于–文档”几类关系边；再转成 PyTorch Geometric 的图数据，训一个**两层 R-GCN**（关系图卷积网络：按关系类型分别聚合邻居信息的图神经网络），做链路预测——遮住图上一部分边让模型猜“这两点之间该不该有条边”，用 AUC 或 Hit@10 报成绩。真实系统用三层 R-GCN 给检索证据排序（第 三 章），你们这两层小网练的就是同一门手艺，看下图：

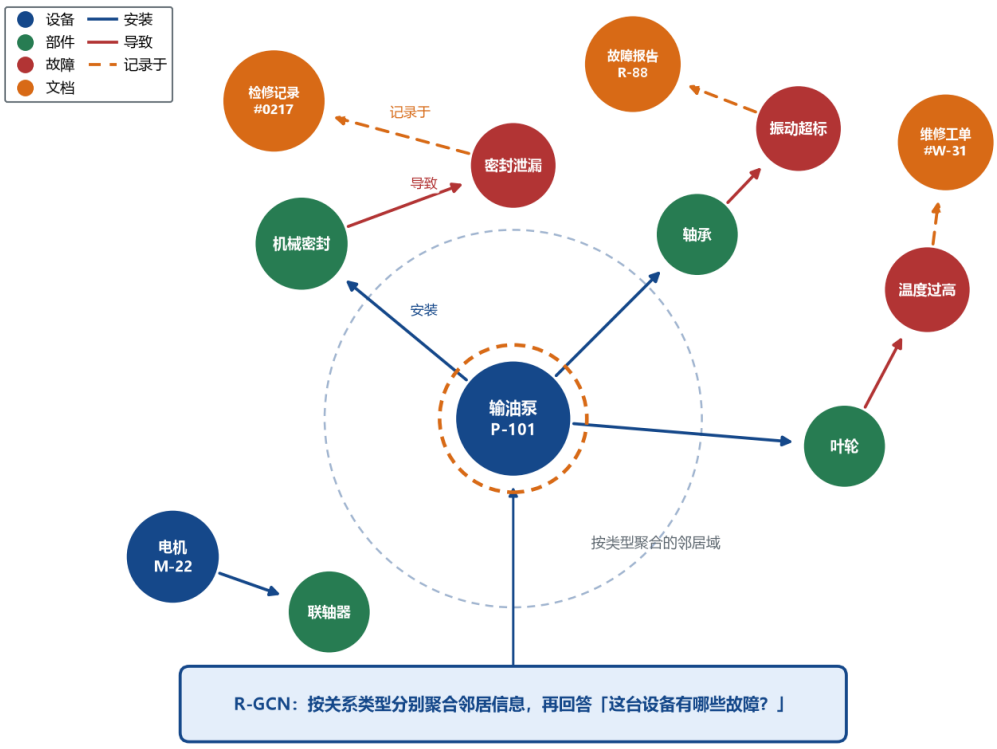


图 14.2　R-GCN 按关系类型聚合邻居：不同颜色的边是不同类型的关系，节点“信谁”取决于它沿各类关系连着谁；第 14 组在 50–100 节点的小图上训两层 R-GCN 做链路预测，练的就是这套聚合的本事

项目连线：问数与图谱，第一篇都有老巢

第 2 讲：反向传播——R-GCN 再“图”，它也是一层层前向、把误差一层层反传的网络；第 8 讲：本地部署与量化——Ollama 就是“小模型搬回家”的现成工具；第 11 讲：注意力与 agent——“查询-回答-引用”的循环正是 agent 范式；第 3 讲：评估——30 题怎么算对错、AUC 怎么读，全按它的方法论来。

另外记住你们的**对抗联动**：第 15 组会拿造假样本“污染”你们的测试题库——在 30 题里混进几条似是而非的假记录，看你们的安全检查与引用核对能不能接住。这个联动不是找茬，是两边都涨本事的合练。

14.4　第 19 周周五提交清单

提交线：2027年 1 月 22 日（第 19 周周五）完全提交离场。你的组交齐下表七样，每样都与第 三 章的总清单模板对应。

表 14.3　第 14 组提交清单（2027 年 1 月 22 日）

交付物	格式	说明
问数 demo 代码包	zip，含 README 与启动脚本	一键启动脚本 + 建库脚本 + 演示库；README 写清模型下载与运行步骤，别人拿到能跑
30 题测试集 + 标准答案	表格	**标准答案独立编写、先行复审**——系统生成的 SQL 不得反过来充当标准答案；含 ≥10 题原创边界题
R-GCN 实验 notebook	ipynb+ 曲线	建图、训练、评估全流程可复跑，loss 曲线与 AUC/Hit@10 齐全
本地 vs 在线对比报告	PDF，2–4 页	30 题上准确率/延迟/成本三列对比，附“何时用本地、何时用在线”建议
小组项目报告	PDF，8–15 页	按第 三 章模板：摘要/背景/方法/实验/结论/分工声明
证据包	zip	四次实验各一包，命令 + 日志 + 产物三件套，日志为终端完整输出
AI 使用声明	一页	哪些事 AI 做了、人做了什么核验，逐条列清

14.5　AI 分工与人工必做

分工原则一句话：AI出草稿，人出签字——30 题的每份标准答案，编写的人负全责。

表 14.4　第 14 组 AI 分工与人工必做

AI可以自动做	人必须亲自做
写三个工具函数与建库脚本草稿	**独立编写 30 题标准答案（先于测试，AI 不得反推）**
生成测试问题草稿与边界题点子	**逐题核对回答与引用**：数字点回数据库原文
解释报错信息、给排查建议	运行 demo 并录屏留证
起草对比报告与 README 文字	自己出一页 SQL 注入测试题打自己的防线
建议提示词与输出格式写法	在测试集与指标上签字，对每个数字担责

14.6　防糊弄指南与满分标准

防人工糊弄指南（第 14 组版）

(1) 标准答案不许让 AI 反推——先独立写答案、后跑系统，顺序反了就是“考生批卷”；系统生成的 SQL 永远不能反过来充当标准答案。**(2) 30 题里至少 10 题是原创边界题**——缺条件该追问的、说“前五条”就该只给五条的、查无结果该老实报空的；全挑软柿子捏不算测试。**(3) 编造成功率 = 该节点成绩作废**——每题的对错明细都要能对账，报“成功率 90%”就得拿得出 30 行明细。**(4) 代跑必须留运行者姓名与时间**，谁跑的谁负责。

防 AI 糊弄指南（第 14 组版）

(1) 本地模型温度必须设 0——不设 0，同一个问题两次回答俩样，可追溯性无从谈起。**(2) “模型说查到了”不算数**——必须点回数据库原文：数字、时间戳、行数，逐项对上才算对。**(3) 别把 API key 写进代码包**——用环境变量，交包前全文搜一遍。**(4) AI 写的 SQL 安全检查可能有漏**——比如没拦住 PRAGMA、ATTACH 这类旁门语句，注入测试要自己出题自己防，别迷信 AI 写的“万能拦截”。

按课程建议方案，实验一不计分，实验二至四各 10 分共 30 分，平时作业 20 分（十二讲章后习题十二选二）；期末成果满分 50 分（教师可调）。成果怎么拿满，看下面这张 50 分制细则。

表 14.5　第 14 组成果满分标准（50 分制）

分值	项目	怎么拿满
15	问数 demo	实验二建库 + 一句话问数跑通并录屏（8 分）；白名单工具与只读检查齐备、通过自设注入测试（7 分）
10	R-GCN 实验	建图规范、训练真实完成、loss 曲线齐全，报出 AUC 或 Hit@10 并有分析
8	测试集与评估	30 题独立标准答案 + 至少 10 题原创边界题；每题对错明细可对账
7	报告质量	组报告按模板 8–15 页，方法讲得清、图表齐全、结论有数据支撑
5	协作与台账	排班执行有记录；录屏与明细归档完整；求助记录规范
5	加分	如：给 demo 加“缺条件追问”并演示；加语义检索提升表名/列名命中率；对 R-GCN 做关系类型消融实验

14.7　四次实验安排

四个节点与全班同频（实验一 2026 年 10 月 12 日、实验二 11 月 2 日、实验三 11 月 23 日、实验四 12 月 21 日），但每场都有本组的命令级动作。先看总览表，再逐场展开；每场按“目标—步骤—验收证据—常见坑”执行：步骤照抄就能跑，证据一栏就是成绩。

表 14.6　第 14 组四次实验总览

节点	日期	本组动作	核心验收证据
实验一	10月 12 日	agent 调用你的工具函数，走通“查询-回答-引用”	完整循环日志/录屏
实验二	11 月 2 日	Ollama 部署 + 建库 + 一句话问数跑通	版本输出 + 问答录屏
实验三	11 月 23 日	R-GCN 链路预测真训练	训练日志 +loss 曲线 +AUC
实验四	12 月 21 日	30 题本地 vs 在线对比	明细表 + 对比表 + 账单

14.7.1　实验一（10 月 12 日）：当一回 agent，走通“查询-回答-引用”

目标：配好大模型编程助手的 agent 模式，让它调用你们写好的工具函数完成一次“查询-回答-引用”——亲身体验 agent 循环，理解“模型提要求、程序干活、结果回填”怎么转起来。

步骤：(1) 按课堂演示装好编程助手并切换到 agent 模式；(2) 先把三个工具函数写出草稿（AI 起草、人核对函数签名与 docstring——每个参数什么含义、返回什么 JSON，写不清 agent 就不会用）；(3) 把工具挂给 agent（或手写一个最简循环：模型输出工具名与参数 → 程序执行 → 结果回填对话）；(4) 提一个“2023 年 11 月 18 日 20:27 的轴向振动”式问题，要求 agent 给出“答案 + 所用 SQL+ 数据来源行号”三件套；(5) 把完整循环记录归档。

验收证据：一次完整“查询-回答-引用”的日志或录屏，含工具调用参数与返回原文。

常见坑：agent不调工具直接“背答案”——提示词里写明“必须调用工具函数作答，禁止凭记忆回答”；工具函数没有 docstring，agent 瞎猜参数含义，调用连连报错；历史对话不回填，agent 第二轮就忘了第一轮查到了什么。

14.7.2　实验二（11 月 2 日）：本地小模型上岗，一句话问数跑通

目标：本地部署 Ollama 小模型，建好演示库，白名单工具与 SQL 安全检查齐备，一句话问数端到端跑通。

步骤：(1) 装 Ollama，`ollama pull qwen2.5:1.5b`，**下载完再调用**（`ollama list` 能看到模型才算完）；(2) 把四个精简 CSV 导入 SQLite/DuckDB，记录各表行数；(3) 接模型：提示词限定“只输出一条 SQL，不输出别的”，温度 0；(4) 写 `check_sql()`：只读 + 列名白名单两道闸，先拦后跑；(5) 跑通三个样例问题（明细/聚合/列名说明各一），全程录屏。

验收证据：`ollama list` 输出、建库行数统计、三个样例问答的录屏（含 SQL 与查询结果）。

常见坑：Ollama 模型没下载完就调用，报错还以为代码有问题；模型输出带解释文字不是纯 SQL，解析当场崩——输出格式要在提示词里钉死；**SQL 注入测试题自己没防住**——自己写一页攻击用例（改数据的、查系统表的、多语句的）逐条打自己的闸门；温度忘设 0，两次回答不一致被同学当场抓包。

14.7.3　实验三（11 月 23 日）：小图上真训练——两层 R-GCN 做链路预测

目标：在自建的小知识图谱上真实训练两层 R-GCN 做链路预测，画 loss 曲线、报出 AUC 或 Hit@10——这是“深度网络的模型训练实践”在本组的落点。

步骤：(1) networkx 建图：50–100 个节点、四类实体、至少三类关系，登记节点数/边数/各类关系条数；(2) 转成 PyTorch Geometric 的图数据，节点特征做简单编码并**归一化**；(3) 切边：训练边与验证边按比例分开，口径写清楚；(4) 训两层 R-GCN：正样本是被遮住的真边，负采样配对假边，二分类训练；(5) 每个 epoch 记 loss，画曲线；(6) 验证集上报 AUC 或 Hit@10。

验收证据：建图统计、训练日志、loss 曲线、AUC/Hit@10 数字及其含义解释。

常见坑：**节点特征忘了归一化**，训练曲线抖成锯齿；图太小（不足 50 节点）指标虚高到不好意思；负采样全用“隔着十万八千里”的点对，模型学成“看距离猜边”；torch-geometric 与 torch 版本不匹配装不上——按官方版本对应表锁定版本再装。

14.7.4　实验四（12 月 21 日）：本地 vs 在线——30 题上一张对比表

目标：用在线大模型 API（或在线 embedding 检索）对比本地方案，在固定的 30 题上量出准确率、延迟、成本三笔账，形成“什么时候用本地、什么时候用在线”的结论。

步骤：(1) 固定 30 题测试集（样例题 + 原创边界题，标准答案早已独立编写）；(2) 本地方案全量跑一遍，记每题对错与耗时；(3) 在线方案同样提示词再跑一遍，记对错、耗时与花费；(4) 填三列

对比表：准确率/中位延迟/API 花费；(5) 写不少于 100 字的建议：什么问题本地够用、什么问题值得上在线、断网时怎么兜底。

验收证据：30题逐题明细表（两方案各一份）、三列对比表、建议段、花费账单。

常见坑：成功率统计基数写错——分母是 30，不是“跑完的那些题”，崩掉的题记失败不算弃权；对比时本地与在线用了不同提示词，变量没控住比了白比；只算准确率不记延迟与成本，“在线更好”的结论说不清代价；在线 API 当天抽风没有兜底——留一次重跑时间。

第 15 组　数据基座 · 虚假内容识别与对抗训练组

本章地图

讲什么：第 15 组任务书——办一场缩小版“造假者 vs 监考 vs 判官”对抗：让在线大模型按配额造假，规则 + 小模型当监考，攒出 ≥500 条真假混装数据集，训出真假分类器，报出宏观 F1，并交一份带重叠率数字的泄漏自查报告。**为什么值得读**：真实系统那次把 0.88 改判成 0.74 的审计（第 三 章）证明：不能证明没泄漏的成绩不算成绩。你们组的每一样交付物，都在练“诚实”这门手艺。**学完你能**：按配方指挥 AI 造假并记好账；独立完成规则基线 + 小模型两版分类器；给出带数字的泄漏自查，看得穿“指标虚高”的把戏。

一句话概括你们组：全班唯一“官方允许造假”的小组——你们的业务是让 AI 扮演造假者日夜出题，把你们自己的分类器练成测谎仪；而且测谎仪几斤几两，要按最苛刻的口径如实招报。铁律先立：**大模型只能改说法，永远改不了事实——目标假数值由程序给定**。

15.1　组档案

表 15.1　第 15 组档案

项目	数据基座治理（项目三，工程资料结构化与本地数据查询系统，详见第 三 章）
组号与组名	第 15 组 • 虚假内容识别与对抗训练组
人数与投入	5 人；每周合计 6–8 小时（组内自行排班，建议每周固定 2 次）
负责的能力	对抗训练 demo 的教学版——缩小版“造假者 vs 监考 vs 判官”攻防闭环
上游依赖	真实系统现成材料：26 个造假家族清单与 2,000 条交付数据集的“配方”（假 1,000 条按易 300/中 400/难 300 分层）——你们做缩小版
联动依赖	第 14 章（第 14 组）：你们的假样本可进它的测试题库（对抗联动）
工作目录	`tasks/d03_adversarial/`（`fake/` 放造假产物，`dataset/` 放混装数据集，`model/` 放分类器）
硬件要求	TF-IDF+ 逻辑回归 CPU 秒级出结果；embedding 小网络笔记本可训；API 花费是主要成本，须记账

先说清楚你们和真实系统的关系：真实系统那场攻防是“重装部队”——四角色闭环、3,300 道封存考卷、蒸馏库 2,654 条、每轮配额 30 条直接造假 +10 条语言造假 +10 条纯文字造假、最多 5 条润色。你们是“轻装分队”：同一套打法，规模全线缩小——4 轮对抗、≥500 条数据集、≥10 个造假家族。**配额可以缩，但配额制度不能省**：每轮造多少条，开学第一周写进配方文档，之后一轮一记账。求助通道照旧：组内互助 → 项目助教 → 导师，求助附完整命令与完整输出。

15.2　项目背景与你的位置

第 三 章的高潮就是你们的原型：门控规则抓得住“没出处、证据打架、对象不清”，却抓不住**语义词造假**——无数值格式错误、无逻辑硬伤、纯粹在事实层面撒谎。系统的回应是四角色闭环：**造假者**（大模型 + 程序给定假数值）批量生产以假乱真的条目，骗过的题目回流给它下一轮“进化”；**监考**不确定必问、放行抽检 10%；**判官**只认数值与单位等硬证据，判“真”须置信度 ≥ 0.75；**库存管家**管蒸馏库与先例库，蒸馏样本以 3 倍权重回流训练。26 个造假家族按易中难分层，考卷 3,300 道封存待考。

然后是那个你们必须烂熟于心的故事：2026 年 9 月 15 日的审计发现考卷与训练数据事实级重叠 78%、逐字重复 15.7%，旧口径的 0.84–0.88 全部作废，重建考卷后真实水位落在 0.72–0.74——**“不是性能回退，是挤掉水分”**。审计还留下一条尾巴：300 道 OOD 纯文字造假考题 F1 只有 0.071–0.233，系统对语义造假近乎失明；缺口表 **G10**——“难负样本与对抗训练尚未形成成果”——就是你们的进场口。看下图，四角色闭环的每一环，这学期都要在你们手里缩样跑通：

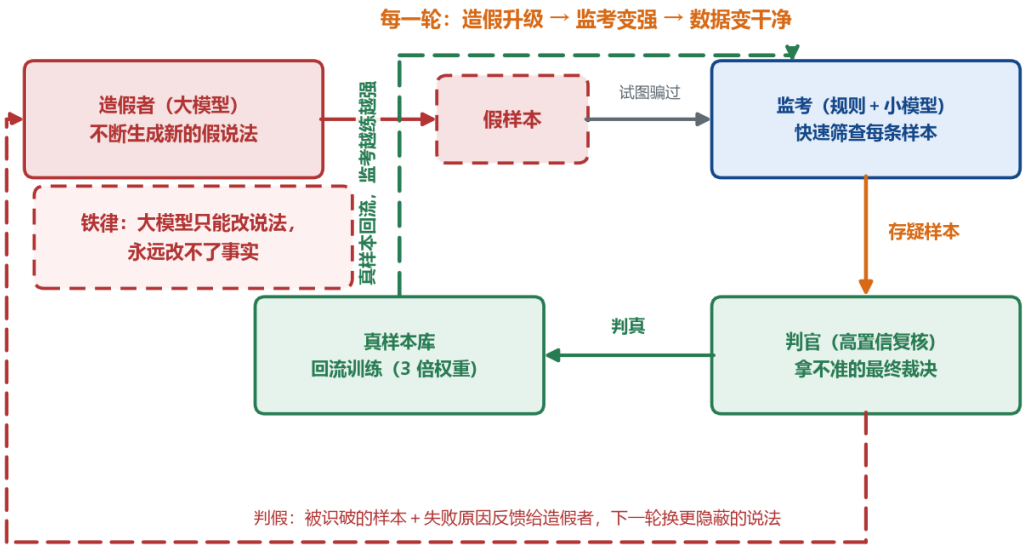


图 15.1　真假内容攻防闭环：造假者（大模型 + 程序给定假数值）出题，监考判定与追问，判官凭硬证据复核，蒸馏样本回流再训——第 15 组要把这个闭环缩样跑通四轮，并诚实报告成绩的水分检查

15.3　核心任务

六步任务一张表看明白，随后逐步说人话。

表 15.2　第 15 组六步任务：干什么与产出什么

任务	干什么	产出
造假者上线	在线大模型 API 按配额造假，每周一轮共 4 轮：直接造假/语言造假/纯文字造假三类，假数值由程序给定	造假样本流
监考上线	规则（数值越界/单位错/时间戳矛盾）+ 小模型判定，不确定必问、放行抽检	判定记录
数据集 v1	≥500 条真假混装：假 250（易 75/中 100/难 75）+ 真 250，洗牌编号，双金标	CSV
分类器	TF-IDF+ 逻辑回归基线，加一个 embedding 小网络对比	指标表
泄漏自查	算考卷-训练重叠率，**必须给出数字**	自查报告
蒸馏回流	判对的难样本 3 倍权重回流再训一轮，看宏观 F1 变化	前后对比

逐步展开。**造假者**的提示词框架是“程序给定目标数值，大模型只负责把说法包装得像真的”——比如程序把“排量 120”改成“135”，模型的任务是给 135 编一圈合情合理的话术；它无权自己发明数字。**监考**先上三条规则（数值越界/单位错/时间戳矛盾），再补一个小模型；拿不准的必须发问，不许硬判。**数据集**的公平性全在细节里：真假各半、难度分层、**随机洗牌**后按 `MIX_0001` 起编号——任何人不能从编号顺序猜出真假。每条样本挂**双金标**，看下面的知识盒。**分类器**两版：TF-IDF+ 逻辑回归是基线（词袋特征 + 线性分类器，第 1–3 讲的老朋友），embedding 小网络是把文字变向量再分类的小模型，两版同题比武。**泄漏自查**是本组的灵魂，呼应 0.88 → 0.74 的审计故事。

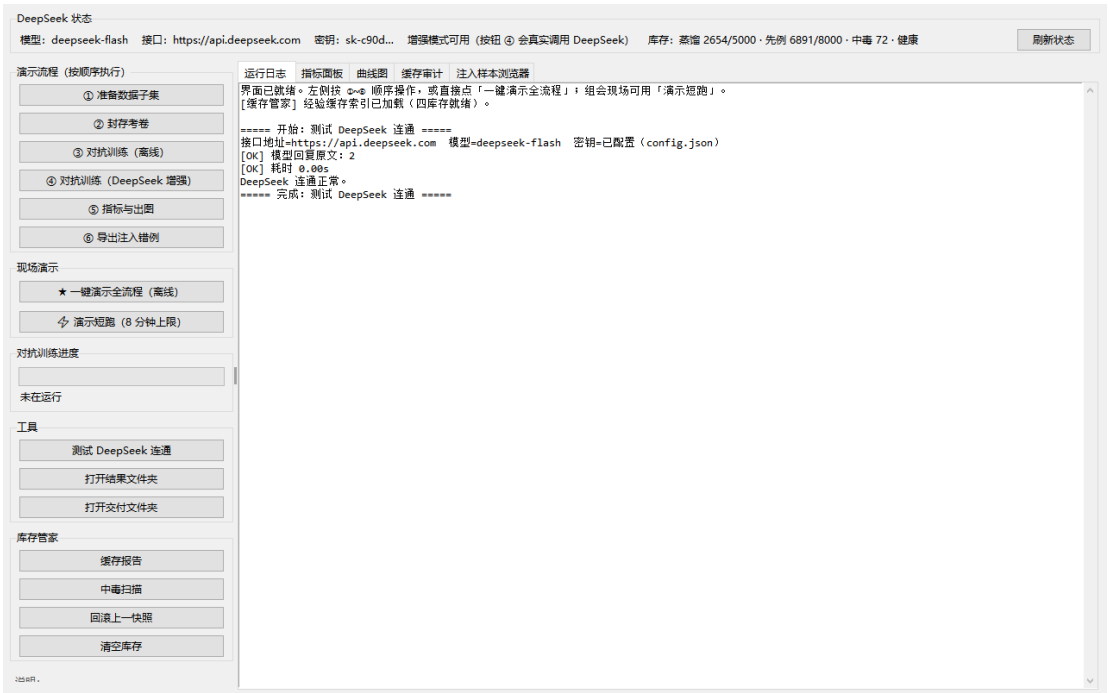


图 15.2　真实系统对抗训练 demo 主界面：左侧六步演示流程（准备数据子集 → 封存考卷 → 离线对抗 → DeepSeek 增强 → 指标与出图 → 导出注入错例），右侧运行日志，顶部常驻库存水位——第 15 组的六步任务就是把这流程缩样搬进 `tasks/d03_adversarial/`

图片来源：本项目 demo 截图（E:\中海油数据）

知识补齐：双金标——客观真假与判定口径分开记

每条样本挂两个标签：`gold_label` 是**客观真假**——这条内容到底是不是假的，由构造过程保证（程序改的数值，天生为假）；`disc_label` 是**判官当时的判定口径**——监考与判官把它判成了什么。两标分开，才能算出两种错：把假的判成真（漏报，最危险）与把真的判成假（误报，冤枉好人）。只留一个标签，这两种错就糊成一团，复盘时分不清该修监考还是该修判官——真实系统 2,000 条交付数据集正是双金标设计，你们的 v1 数据集照方抓药。

项目连线：造假的活，第一篇早埋了伏笔

第 3 讲：评估方法论——宏观 F1 的口径、查准查全的分母，全按它执行；第 6 讲：训练循环——蒸馏回流再训就是“数据回流、模型更新”；第 12 讲：生成对抗思想——造假者与监考就是一对“生成器 vs 判别器”，互相逼强的军备竞赛，赌注从假图像换成了假数据。

15.4　第 19 周周五提交清单

提交线：2027年 1 月 22 日（第 19 周周五）完全提交离场。你的组交齐下表七样，每样都与第 三 章的总清单模板对应。

表 15.3　第 15 组提交清单（2027 年 1 月 22 日）

交付物	格式	说明
对抗数据集 v1	CSV，≥500 条	假 250（易 75/中 100/难 75）+ 真 250；分层标签 + 双金标（`gold_label`/`disc_label`）+ 洗牌 `MIX_` 编号
造假配方文档	PDF 或 Markdown	≥10 个家族的缩小版配方：每家族给定义、示例、难度档、提示词模板与配额
分类器代码与指标表	代码包 + 表	基线与小模型两版对比：准确率/查准/查全/**宏观 F1**，回流前后各一份
泄漏自查报告	PDF，2–3 页	重叠率数字 + 检查方法 + 逐条结论——不许“自查通过”一句带过
小组项目报告	PDF，8–15 页	按第 三 章模板：摘要/背景/方法/实验/结论/分工声明
证据包	zip	四次实验各一包，命令 + 日志 + 产物三件套，含 API 调用与花费记录
AI 使用声明	一页	AI 当造假者的轮次与配额、人工抽检记录索引，逐条列清

15.5　AI 分工与人工必做

分工原则一句话：AI当反派，人当裁判——每条样本的真假，终审权在人。

表 15.4　第 15 组 AI 分工与人工必做

AI可以自动做	人必须亲自做
当造假者（按配额！假数值由程序给定）	**人工抽检 ≥100 条判真假（AI 不可代看）**
起草造假配方与提示词模板	核对双金标一致性：`gold_label` 与 `disc_label` 逐条对账
预标真假标签供监考参考	控制 API 花费：一轮一记账，超配额轮次作废
画指标曲线、起草报告文字	封存考卷、隔离回流样本（时间戳为证）
解释报错信息、给排查建议	在数据集与自查报告上签字，对每个数字担责

15.6　防糊弄指南与满分标准

防人工糊弄指南（第 15 组版）

(1) 抽检记录要留痕——抽了哪 100 条（编号区间）、你判对多少条，两个数字缺一不可；没有数字的“已抽检”等于没抽检。**(2) “全对”要小心**——真有 100% 吗？想想 0.88 是怎么死的：漂亮得反常的指标，第一反应该是怀疑测量，不是庆祝。**(3) 真假失衡或忘了分层洗牌 = 数据集重做**——编号能猜出真假的数据集没有考试资格。**(4) 编造 F1= 该节点成绩作废**——指标表每个数字都要能与训练日志对上账。

防 AI 糊弄指南（第 15 组版）

(1) 铁律：大模型只能改说法，永远改不了事实——目标假数值由程序给定；造假者“越权自创数值”的样本一律作废，不得入集。**(2) 考卷的 AI 复核结论绝不回流进蒸馏集**——判官看了考卷再出题，等于泄题。**(3) API 用量记账**——每轮调用次数与花费写进台账，超配额的轮次整轮作废。**(4) AI 判真假只有“参考一票”**——它没看过你的数据库，终审永远是人。

按课程建议方案，实验一不计分，实验二至四各 10 分共 30 分，平时作业 20 分（十二讲章后习题十二选二）；期末成果满分 50 分（教师可调）。成果怎么拿满，看下面这张 50 分制细则。

表 15.5　第 15 组成果满分标准（50 分制）

分值	项目	怎么拿满
13	数据集与配方	≥500 条分层齐全、洗牌可验证、双金标一致（8 分）；配方 ≥10 家族且每家族有可复现示例（5 分）
12	分类器与指标	规则 +TF-IDF 基线（5 分）；embedding 小网络与回流对比（7 分）；宏观 F1 口径正确、曲线齐全
8	泄漏自查与如实度	给出重叠率数字与检查方法；抽检留痕；“人工确认计数”如实登记——虚报一处，本栏清零
7	报告质量	组报告按模板 8–15 页，方法讲得清、图表齐全、结论有数据支撑
5	协作与台账	排班执行有记录；API 账本无缺失；求助记录规范
5	加分	如：自创造假家族并被配方收录；对纯文字造假做专门评测并给出盲区分析；抽检升级为双人独立判 + 仲裁

15.7　四次实验安排

四个节点与全班同频（实验一 2026 年 10 月 12 日、实验二 11 月 2 日、实验三 11 月 23 日、实验四 12 月 21 日），但每场都有本组的命令级动作。先看总览表，再逐场展开；每场按“目标—步骤—验收证据—常见坑”执行：步骤照抄就能跑，证据一栏就是成绩。

表 15.6　第 15 组四次实验总览

节点	日期	本组动作	核心验收证据
实验一	10月 12 日	AI 造假 10 条，人工 + 规则各判一遍	双份判定 + 失误清单
实验二	11 月 2 日	三条规则 +TF-IDF 基线，扩到 100 条	两版基线指标表
实验三	11 月 23 日	≥500 条数据集训分类器 + 泄漏自查	曲线 + 指标 + 重叠率
实验四	12 月 21 日	在线判官复核 + 蒸馏回流再训	三方案对比 + 回流增益

15.7.1　实验一（10 月 12 日）：先当一回受害者——AI 造假你来判

目标：配好大模型编程助手的 agent 模式并体验角色扮演：让 AI 当造假者生成 10 条假“设备振动记录”，人工与规则各判一遍，记录各自的失误——先挨一刀，再学防身。

步骤：(1) 按课堂演示装好编程助手并切换到 agent 模式；(2) 写造假脚本：程序先随机把记录里某个数值乘一个系数（**假数值由程序给定**），再把假数值与原话术一起交给 AI，让它把说法包装得像真的；(3) 生成 10 条假记录，混入 5 条真记录，洗牌编号；(4) 组内每人在不看答案的情况下一条条判，记录判定与把握程度；(5) 写三条规则（数值越界/单位错/时间戳矛盾）让程序再判一遍；(6) 对账：人工错几条、规则错几条、分别错在哪。

验收证据：15 条样本 + 人工与规则两份判定记录 + 失误清单。

常见坑：造假者太“良心”，假得一望便知——提示词要明确“以假乱真”的口径；人工判题过度自信——先判完再对答案，不许边看边猜；真/假样本不洗牌，顺序可猜，判定记录作废。

15.7.2　实验二（11 月 2 日）：规则当第一道闸，词袋小模型当基线

目标：把三条规则扩成规则引擎，再把样本扩到 ≥100条，训出 TF-IDF+ 逻辑回归基线分类器——两条基线的成绩就是以后一切改进的对照物。

步骤：(1) 用 pandas/sklearn 把数值越界、单位错、时间戳矛盾三条规则写成可配置的检查函数；(2) 按配方扩造假样本至 ≥100 条（AI 按配额造假，开始记账）；(3) `TfidfVectorizer` 提特征，`LogisticRegression` 训练，留出验证集；(4) 规则法与基线各算一份准确率/查准/查全/F1，填进同一张表；(5) 分析：哪类造假规则抓得住、哪类抓不住（纯文字造假大概率漏网——如实记下这片盲区）。

验收证据：规则检查代码、≥100 条样本、两版基线的指标表。

常见坑：规则设得太严把真样本全判假，查准率崩盘——规则是闸门不是筛子；类别失衡却只看总体准确率；sklearn 版本差异导致指标对不上——README 里锁版本。

15.7.3　实验三（11 月 23 日）：主力会战——500 条数据集、两个模型、一份自查

目标：完成本组主力实验：构建 ≥500条真假混装数据集，训练 TF-IDF+ 逻辑回归与 embedding 小网络两版分类器，画 loss 与 F1 曲线，并给出带数字的泄漏自查——这是“深度网络的模型训练实

践”在本组的落点。

步骤：(1)按配方补齐数据集：假 250（易 75/中 100/难 75）+ 真 250，**分层随机洗牌**，`MIX_` 编号，双金标逐条填好；(2) 训 TF-IDF+ 逻辑回归基线；(3) 训 embedding 小网络：把文本变向量（可用预训练词向量平均或轻量 embedding 层）接两层小网络分类，记录每个 epoch 的 loss；(4) 验证集上报**宏观 F1**——每一类各算 F1 再取平均，少数类的成绩不被多数类淹没；(5) 泄漏自查：算考卷-训练重叠率（近似重复一条、事实级重叠一条——同设备、同数值、同时间窗算同一事实变体），**给出数字**；(6) 对照 0.88 → 0.74 写一段不少于 100 字的心得。

验收证据：数据集 v1（CSV）、两版训练日志与 loss/F1 曲线、重叠率数字、心得段。

常见坑：**F1 只算二分类正类没算宏观**——易类把总分拉高、难类的糟糕被平均掩盖；泄漏自查流于形式——只写“已检查无泄漏”不给数字，按未做处理；洗牌前忘了按难度分层；同一事实的变体一边进训练一边进考卷——这正是审计抓到的头号水分，别自己再犯一遍。

15.7.4　实验四（12 月 21 日）：在线判官复核，蒸馏回流再上一分

目标：接入在线大模型 API 当“判官复核”，对比纯本地/纯在线/混合三种判定的宏观 F1 与成本；把判对的难样本以 3 倍权重回流再训一轮，看 F1 能不能诚实地涨。

步骤：(1) 先封存考卷：从数据集抽 50 条锁进 `exam/` 目录，**先封存、后训练**，时间戳为证；(2) 三种判定各跑一遍考卷：纯本地（自家分类器）、纯在线（API 逐条判）、混合（本地判，拿不准的交 API 复核，每轮不超过 25 条——“不确定必问”的教学版）；(3) 各算宏观 F1 与花费，填三方案对比表；(4) 把本地判对的难样本以 3 倍权重回流，基线再训一轮，与回流前对比；(5) 写不少于 100 字结论：在线值不值、回流涨没涨、水分检查怎么做的。

验收证据：三方案 F1 与成本表、回流前后对比、结论段、API 账单。

常见坑：**回流样本混进考卷**——先封存再训练，顺序不能反；**把 API 复核结论写回流蒸馏集**——铁律第 (2) 条，泄题比不训还糟；成本超支不记账，台账缺一页本栏扣分；涨了零点几个百分点就宣布胜利——报数字时带上波动范围，诚实到最后一厘米。

附录　参考资料与图片来源

本讲义在编写过程中参考了大量教材、论文、开源项目与公开课程资料；正文插图分为三类：自绘示意图（本书原创，matplotlib绘制，脚本随源码归档）、真实项目素材（三个项目的工程数据与系统截图，仅教学内部使用）、网络公开图片（逐图标注来源）。以下分四节列出。

一、教材与主要参考书

- 许庆阳.《深度学习模型与算法基础》. 清华大学出版社, 2023 年 12 月第 1 版.（本课程教材）
- Ian Goodfellow, Yoshua Bengio, Aaron Courville.《深度学习》. 人民邮电出版社, 2017 年 7 月第 1 版.（参考书）
- Christopher M. Bishop. Pattern Recognition and Machine Learning. Springer, 2006.（贝叶斯视角经典）
- Richard S. Sutton, Andrew G. Barto. Reinforcement Learning: An Introduction. 2nd ed. MIT Press, 2018.（扩展阅读）
- Stanford CS231n: Deep Learning for Computer Vision 课程讲义. cs231n.stanford.edu.（卷积网络部分参考）
- Christopher Olah. Understanding LSTM Networks 博客文章. colah.github.io, 2015.（第 9-10 讲插图来源）
- Jay Alammar. The Illustrated Transformer / The Illustrated Stable Diffusion. jalammar.github.io, 2018/2022.（第 11-12 讲插图来源）

二、主要论文与开源项目（按讲次）

- 第 1-2 讲：F. Rosenblatt. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. Psychological Review, 1958; D. Rumelhart, G. Hinton, R. Williams. Learning representations by back-propagating errors. Nature, 1986.
- 第 4 讲：D. Erhan et al. Why Does Unsupervised Pre-training Help Deep Learning? JMLR, 2010; K. He, X. Zhang, S. Ren, J. Sun. Deep Residual Learning（MAE 思想同源）.
- 第 5-8 讲：A. Krizhevsky, I. Sutskever, G. Hinton. ImageNet Classification with Deep CNNs. NeurIPS 2012（AlexNet）; K. Simonyan, A. Zisserman. Very Deep Convolutional Networks. ICLR 2015（VGG）; C. Szegedy et al. Going Deeper with Convolutions. CVPR 2015（GoogLeNet）; K. He et al. Deep Residual Learning for Image Recognition. CVPR 2016（ResNet）; M. Zeiler, R. Fergus. Visualizing and Understanding Convolutional Networks. ECCV 2014; A. G. Howard et al. MobileNets. 2017; J. Long, E. Shelhamer, T. Darrell. Fully Convolutional Networks. CVPR 2015; L.-C. Chen et al. Encoder-Decoder with Atrous

Separable Convolution (DeepLab v3+). ECCV 2018; A. Kirillov et al. Segment Anything. ICCV 2023.

- 第 9-10 讲：J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. PNAS, 1982; M. I. Jordan. Serial Order: A Parallel Distributed Processing Approach. 1986; S. Hochreiter, J. Schmidhuber. Long Short-Term Memory. Neural Computation, 1997; K. Cho et al. Learning Phrase Representations using RNN Encoder-Decoder (GRU). EMNLP 2014; A. Gu, T. Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. 2023.
- 第 11 讲：A. Vaswani et al. Attention Is All You Need. NeurIPS 2017; J. Devlin et al. BERT. NAACL 2019; A. Radford et al. Language Models are Few-Shot Learners (GPT-3). NeurIPS 2020; A. Radford et al. Learning Transferable Visual Models From Natural Language Supervision (CLIP). 2021; OpenAI/o1 与 DeepSeek-R1 推理模型技术报告, 2024–2026.
- 第 12 讲：I. Goodfellow et al. Generative Adversarial Nets. NeurIPS 2014; A. Radford et al. Unsupervised Representation Learning with DCGAN. ICLR 2016; T. Karras et al. A Style-Based Generator Architecture for GANs (StyleGAN). CVPR 2019; R. Rombach et al. High-Resolution Image Synthesis with Latent Diffusion Models. CVPR 2022.
- 项目一素材：360Stitcher 项目任务书与项目理解报告（教学内部资料, 2026）; E. Rublee et al. ORB: An Efficient Alternative to SIFT or SURF. ICCV 2011; P. E. Sarlin et al. SuperGlue. CVPR 2020; P. Lindenberger et al. LoFTR. ICCV 2021.
- 项目二素材：中海油水下双目项目任务书与自动打标签调研（教学内部资料, 2026）; NVIDIA FoundationStereo (CVPR 2025) 与 Fast-FoundationStereo (CVPR 2026) 项目页; L. Yang et al. Depth Anything V2. NeurIPS 2024; UWStereo/FLSea/SQUID 公开水下数据集; Video Depth Anything (CVPR 2025); Stereo Anywhere (CVPR 2025); MonSter (CVPR 2025); DEFOM-Stereo (CVPR 2025).
- 项目三素材：工程资料结构化与本地数据查询系统总体方案（教学内部资料, 2026）; DeepSeek 对抗训练 demo 技术文档（教学内部资料, 2026）; Y. Zhang et al. DragonN-graph / R-GCN: M. Schlichtkrull et al. Modeling Relational Data with Graph Convolutional Networks. ESWC 2018; Microsoft GraphRAG: From Local to Global. 2024; Spider/BIRD Text2SQL 评测基准.

三、网络图片来源（逐图）

以下为正文 `webfig` 类插图的下载来源，全部为公开网页资料，仅用于课堂教学。

图 key	来源域名	说明
web_alexnet	cs231n.stanford.edu	AlexNet 原始架构图 (Krizhevsky 2012, 双 GPU 结构)
web_anchor_chain	commons.wikimedia.org	粗重锚链链环特写 (系泊特征)
web_attention_qkv	jalammar.github.io	多头注意力 Q/K/V 投影示意 (Jay Alammar)
web_camera_rig	commons.wikimedia.org	谷歌街景多相机阵列采集设备

图 key	来源域名	说明
web_cline_vscode	github.com	VS Code 中 Cline AI 编程助手编辑代码界面
web_clip_demo	arxiv.org	CLIP 对比预训练与零样本预测 (Radford et al. 2021)
web_cnn_typical	arxiv.org	CNN 层级特征可视化:Layer1 边缘→Layer5 物体 (Zeiler-Fergus 反卷积重建)
web_data_growth	commons.wikimedia.org	AI 训练数据点指数增长曲线 (Our World in Data)
web_deepfake_example	commons.wikimedia.org	著名 AI 深度伪造图: 教皇白色羽绒服假照 (2023)
web_deeplab_seg	arxiv.org	DeepLab 语义分割结果 (输入/CRF 前后对比)
web_diffusion_stable	jalammar.github.io	Stable Diffusion 三大组件 (CLIP-Text+UNet+ 图像解码器)
web_dl_history	arxiv.org	ILSVRC 历年冠军错误率里程碑 (AlexNet → ResNet vs 人类 5%)
web_fcn_paper	arxiv.org	FCN 端到端像素预测示意 (Long et al. 2015 原图)
web_fpso_ship	commons.wikimedia.org	FPSO"Mystras" 船形浮式生产储卧油装置照片 (甲板 + 火炬)
web_gan_paper	arxiv.org	GAN 训练演化四联图 (Goodfellow et al. 2014 原图)
web_gan_progress	github.com	StyleGAN 生成 1024px 逼真人脸 (NVIDIA 官方样张)
web_googlenet	arxiv.org	GoogLeNet Inception 模块 (带 1x1 降维)
web_gpt_family	arxiv.org	GPT-3 零样本/单样本/少样本 vs 微调 (Brown et al. 2020)
web_iris_flower	commons.wikimedia.org	变色鸢尾 (Iris versicolor) 花朵, 鸢尾花数据集类别之一
web_knowledge_graph	commons.wikimedia.org	知识图谱节点-边可视化示例 (实体关系)
web_labeling_tool	github.com	CVAT 标注工具界面 (图像 + 标签面板)
web_llm_agent_loop	lilianweng.github.io	ReAct 思考-行动-观察循环轨迹 (Lilian Weng 博客)
web_lstm_colah	colah.github.io	LSTM 门结构链式图 (Colah 经典图)
web_machine_vision_camera	en.wikipedia.org	海康威视工业相机产品照

图 key	来源域名	说明
web_mnist_samples	commons.wikimedia.org	MNIST 手写数字 0-9 样本网格
web_mobilenet	arxiv.org	标准卷积 vs 深度可分离卷积分解 (MobileNet 论文原图)
web_neuron_bio	commons.wikimedia.org	生物神经元结构示意 (树突/胞体/轴突/髓鞘)
web_offshore_platform	commons.wikimedia.org	海上油气生产平台黄昏照 (燃烧火炬)
web_ollama_local	github.com	Ollama 官方标志 (本地大模型运行框架)
web_panorama_example	commons.wikimedia.org	360° 室内全景拼接成果 (教堂内部)
web_resnet_paper	arxiv.org	ResNet 残差学习块原图 (He et al. 2015)
web_rnn_colah	colah.github.io	RNN 按时间展开图 (Colah 经典图)
web_rov_underwater	commons.wikimedia.org	Oceaneering 工作级 ROV 水下结构物作业 (框架/推进器/缆)
web_sam_demo	arxiv.org	SAM 可提示分割: 任务/模型/数据引擎 (Kirillov et al. 2023)
web_stereo_camera	commons.wikimedia.org	Kodak 双镜头立体相机硬件
web_transformer_arch	commons.wikimedia.org	Transformer 整体结构 (Vaswani et al. 2017 Fig.1)
web_turing	commons.wikimedia.org	图灵 16 岁经典肖像照片
web_underwater_turbid	commons.wikimedia.org	浑浊水体中的塞氏盘透明度测量照片
web_vgg16	commons.wikimedia.org	VGG16 结构块图 (13 卷积 +3 全连接)

四、真实项目素材与脚本

正文 `projfig` 类插图来自三个项目的真实工程资料，包括：六目拼接项目现场照片与系统遥测截图（`E:\六目拼接`）；HYSY123 FPSO锚链 ROV 检查视频帧、自建 500 对校正立体图对与分析可视化（`E:\锚链视频`）；数据基座治理系统架构图与对抗训练 demo 界面截图（`E:\中海油数据`）。上述素材仅限本课程教学使用，不得对外传播。全部自绘图的 matplotlib 生成脚本与来源清单随讲义源码归档，便于教师复现与修改。