一根记忆”换轻快——记住这句话，上面选型表里最难决的“拿不准”一行，你也就有了自己的判断依据。

10.5　实战对比：一张图看懂差距

口说无凭，上实验。同一个序列预测任务（带噪声的正弦波，模型要隔着很长的窗口抓住周期），普通 RNN 与 LSTM 各训练一次，递推预测几十步，结果见下图：普通 RNN 的预测曲线前几步还像样，越往后越飘，最后干脆自说自话地发散；LSTM 全程贴着真值走。这与上一讲的理论严丝合缝：递推预测时误差要跨越的步数越来越长，普通 RNN 的便签早已把“正弦的相位”稀释殆尽，LSTM 的传送带却把相位信息原样送到了几十步之外。

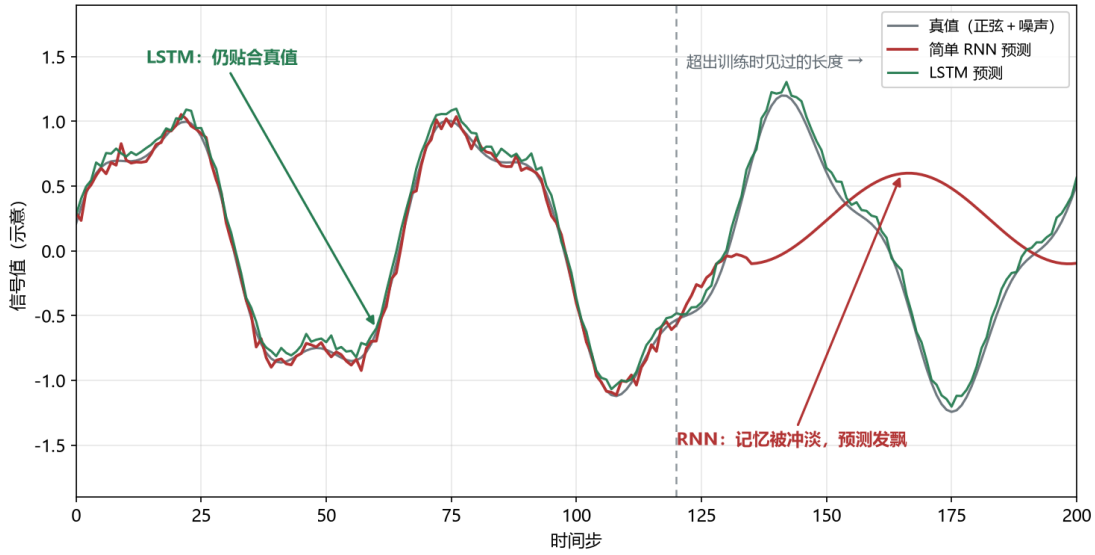


图 10.4　同一个长序列预测任务：普通 RNN 的预测曲线几步之后迅速漂移、发散，LSTM 全程贴合真值——传送带加门控的价值一眼可见

短窗口、短依赖的任务上，两者差距其实不大——别养成“无脑 LSTM”的迷信，10.4 节选型表才是正解。代价也要念清楚：LSTM 参数与单步计算约是普通 RNN 的四倍，嵌入式部署时这笔账不能不算。

回看上一讲引用过的那张经典展开图，现在你该看出新的门道了：

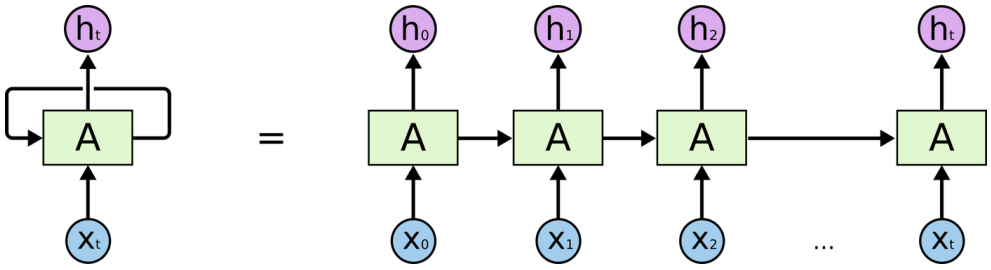


图 10.5　RNN 按时间展开的经典图：隐藏状态 h 沿横向箭头逐站传递——普通 RNN 的这条箭头每站都被重写，换成 LSTM 后它背后多了传送带 c

图片来源：Christopher Olah, ”Understanding LSTM Networks” 博客文章, 2015

第 9 讲你第一次见它时，那条横向箭头只是一根“接力棒”；今天你知道：普通 RNN 把全部记忆押在这一根箭头上，LSTM 给它配了一条不被覆写的传送带当靠山。工程上还有两个常见加强，各

一句话：**双向**（bidirectional）——序列正着读一遍、倒着读一遍，两份便签拼接，适合“整句都读完了再做标注”的离线任务；**多层堆叠**——把 LSTM 像积木一样竖着摞，深层抓抽象规律。PyTorch 里一行 `nn.LSTM(input_size, hidden_size, num_layers)` 全都有，调用返回的输出里就带着 (h, c) 两个状态——课后题让你亲手跑。

10.6　2026 年的选型决策：什么时候还用 RNN 家族

诚实的时局图先画出来：2017年 Transformer 登场后，**大规模 seq2seq 任务——翻译、摘要、对话、文本生成——已被注意力机制全面接管**（第 11 讲）；今天的旗舰大模型里找不到一根循环箭头。但“被接管”只发生在数据富饶的文本大陆。在数据贫瘠的工业地带，2026 年的版图是这样的：

表 10.3　2026 年序列模型选型速查：按你的数据与部署条件对号入座

场景	首选	理由
海量语料的文本生成/翻译/对话	Transformer	数据越多注意力越强，训练全并行
小数据工业时序（传感器/设备状态/遥测）	LSTM/GRU	样本少时结构先验宝贵；模型小、训练便宜
嵌入式/端侧/在线低延迟	GRU	参数少 25%，单步快，状态内存恒定
流式处理（边收边算、状态要携带）	RNN家族	无论多长随身只带一份状态；Transformer 每步重看全历史

更该记住的是：**LSTM 的三个发明没有死，只是搬了家**。门控——现代架构里的门控线性单元（GLU）当激活函数用；传送带——“用加法通道保梯度”早已是标配设计（ResNet 直接受其启发）；“状态即记忆”——大模型推理时的 KV 缓存本质上就是一份可携带的状态。看 2024–2026 年挑战 Transformer 平方复杂度的那些挑战者：Mamba 的选择性状态空间，就是“带门控的循环递推”。学 LSTM 不是学古董，是学序列建模的通用语法——语法不过时，过时的只是某种句式。

前沿速递：xLSTM 与状态空间模型的攻防战（2024–2026）

2024 年 1 月，对，就是 1997 年那位霍赫赖特——他的团队提出 **xLSTM**：给老 LSTM 装上指数记忆容量与新式可并行的单元（mLSTM/sLSTM），公开叫板 Transformer；2025 年团队放出 70 亿参数的 xLSTM-7B，在同等规模上与 Transformer 语言模型打得有来有回。另一条战线，Mamba（2023 底）与 Mamba-2（2024）用“选择性状态空间”把线性复杂度的序列建模做成 2024–2026 年最热的方向之一，基因组、长文档、音频等超长序列任务上全面开花。2026 年中立的观察是：**超大模型与通用文本，Transformer 仍是霸主；中小数据、超长序列、端侧部署，循环/SSM 系在持续反攻**。而工业界每天在产线上跑的传感器时序预测——风场、油田、ROV 遥测、相机状态——至今大量是 LSTM/GRU。你手里的 `temporal_state` 序列与遥测行，正是这几条路线竞争的主战场。

项目连线：门控思想的三张项目入场券

六目拼接：任务书对 `temporal_state` 头的明确要求是“考虑门控结构”（第 一 章）——输入 $[B, T, 24]$ 的相机状态序列（匹配数、增益、运动估计等），预测下一步参数。落地时你会遇到本讲全部选型题：T 取几步？LSTM 还是 GRU？门控怎么帮忙忽略“误匹配突刺”那几步（输入门把异常帧的新货拦在门外）？8 个组各自的选择与理由，就是一场现成的选型表实操。**水下双目**：伪标签过滤的“时序一致”一重（第 二 章）是门控的朴素版——上一帧标签可信度低（遗

忘门开大、旧货放走），这一帧的“新知识”才获准进来；更进一步，ROV 的 OSD 遥测行是典型时序文本，用 GRU 给遥测趋势建模，反过来能给伪标签的“稳定性”打分。**数据基座**：对话式问数的记忆管理（第 三 章）把三扇门翻译成了工程语言——遗忘门：把三分钟前的废话题降权出上下文；输入门：把本轮确认的数据表名、口径写进上下文；输出门：只暴露与当前问题相关的记忆给回答模块。A 组“多轮对话不串题、结果固定”的验收线，考的就是这套机制想不想得清楚。实验四（12 月 21 日）做在线优化时，你的系统要边收新数据边更新自己的认知——那一刻你就是在给系统拨门。

10.7　从这一讲到 2026

LSTM 的王座 2017 年交给了 Transformer，但它的“退居二线”退得很有尊严。2024–2026 年，时序预测领域冒出一批**时序基础模型**：谷歌的 TimesFM（思路就是“时序版 GPT”——解码器结构，用上千亿个真实时间点喂出来的）、亚马逊的 Chronos 家族（2025 年 10 月刚更新到 Chronos-2）、Salesforce 的 Moirai（原生支持温度、节假日这类外部协变量，工业数据场景常被首选）。它们的共同卖点是**零样本预测**——没见过你的数据，拿来就能预测，在多项公开数据集上已开始追平甚至超越传统模型；学术界 2025 年也出现了拿它们与“含 LSTM 的传统基线”正面对比的评估研究，工业过程建模的专用基础模型到 2026 年仍在活跃发展。

可把镜头切回工厂车间，画风立刻不同：传感器数据流、嵌入式设备、低功耗端侧，2026 年大量服役的仍是 LSTM/GRU，理由朴素得就像 10.4 节那张选型表——**小、快、省**。研究与工业实践共同给出的选型判断，一句话说完：**数据多、算力足、要跨域泛化，上时序基础模型；数据少、要在单片机上实时跑，GRU 依然是首选**。所以“该记的记、该忘的忘”这门门控手艺，2026 年还在风场、油田、ROV 遥测和你的单片机里每天运行——学 LSTM 不是学古董，是学小数据时序问题的看家本领。

本讲课后题，是让普通 RNN 与 LSTM 在同一条赛道上打一场擂台赛——你当裁判。五要素如下。

第 10 讲课后编程题（用 AI 实现）

题目：指挥 AI 辅助做一组对比实验。生成 600 点序列 $y = \sin(2\pi t/50)$ 加 8% 噪声；滑动窗口“前 40 步预测第 41 步”；分别训练**简单 RNN**（`nn.RNN`）与 `LSTM`（`nn.LSTM`）各一次，隐层同为 64、学习率同为 0.001、训练轮数相同；然后各自从序列中段起**递推预测 60 步**（预测出的点拼回输入接着预测），画出“真值 vs RNN vs LSTM”对比曲线一张、两条训练 loss 曲线一张，并用 50 字以上写结论：LSTM 赢在哪、代价是什么（参数量、耗时各打印出来对比）。

交付物：(1) 代码文件 `rnn_vs_lstm.py`；(2) 对比曲线 png 与 loss 曲线 png 各一张；(3) 100 字以上结论（谁贴合、谁漂移、差多少）；(4) 一段 50 字以上的“我改了 AI 代码里的什么、为什么”。

提示词建议（可直接抄）：“我是零基础学生，正在学 LSTM。请用 Python + PyTorch 写一个对比实验：(1) 生成 600 点序列 y = sin(2*pi*t/50) + 0.08 倍高斯噪声；(2) 滑动窗口：用连续 40 个点预测下一个点，整理成训练数据；(3) 分别用 nn.RNN 与 nn.LSTM（hidden_size=64，Adam 学习率 0.001，训练 50 轮）各训练一个模型；(4) 从第 300 点起各自递推预测 60 步（把预测值拼回输入继续预测）；(5) 画两张图：预测与真值对比曲线、两条训练 loss 曲线，保存为 png；(6) 打印两个模型的参数量与训练耗时；(7) 每行中文注释，并用 50 字解释：为什么递推预测越长，简单 RNN 漂得越远？”

验收线：两张 png 生成且现象与 10.5 节一致（LSTM 贴合真值、RNN 漂移或幅度跑偏）；能对着打印出的参数量回答“LSTM 多在哪”（三扇门多出的那几套矩阵）；能对同学讲清“递推预测”与“单步预测”的区别；至少一处自己的修改（把窗口从 40 加到 80、换成 GRU 三方对比、去掉噪声看两条曲线怎么变，都算）。**代码是 AI 写的没关系，讲不清楚就是没做。**

加分项（任选其一）：(1) 加入 GRU 打三方擂台，三色曲线一张图，用 10.4 节选型表的语言点评裁判结果；(2) 窗口长度取 10/20/40/80/160 各跑一次，画“窗口越长、RNN 退化越狠”折线，给上一讲的梯度消失一张实验证据；(3) 把噪声从 8% 加到 30%，观察哪个模型先崩，并用门控的语言解释为什么。

课后阅读：教材《深度学习模型与算法基础》第 187–203 页，与本讲内容对应；读 LSTM 结构图时，用“传送带 + 三扇门”的语言把每条箭头讲一遍；读 GRU 一节时，对照 10.4 节的加权平均图自己推一遍公式。

LSTM（长短时记忆网络）

用“细胞状态传送带 + 三扇门”治住梯度消失的循环网络，1997 年由 Hochreiter 与 Schmidhuber 提出。

细胞状态（cell state）

LSTM 的记忆主干 c_t：默认原样前移、只在门控批准时增删，信息可平流直达远方，梯度沿加法通道几乎不衰减。

门（gate）

一个 sigmoid 小网络输出 0–1 的“开度”，逐元素控制记忆的取走/写入/放出；连续可导，开度本身可学习。

遗忘门（forget gate）

决定旧记忆每格保留几成；$f_t \approx 1$ 永志不忘，$f_t \approx 0$ 一笔勾销。

输入门（input gate）

决定候选记忆每格写入几成——该记的记，杂讯拦在门外。

候选记忆（candidate）

本步“值得记的内容”草稿 $\tilde{c}_t$，经输入门审核后才上传送带。

输出门（output gate）

决定细胞状态向外（给 h_t）暴露几成——档案全记着，但此刻只说该说的。

GRU（门控循环单元）

LSTM 精简版：更新门一肩挑“留旧/迎新”（加权平均），重置门管草稿参考多少旧记忆；参数少 25%，效果大体相当。

更新门（update gate）

GRU 里此消彼长的一扇门：旧记忆乘 $1-z$、新货乘 z，一个旋钮管两头。

重置门（reset gate）

GRU 里管“写新草稿时回看多少旧便签”的门。

第 11 讲　注意力模型

本讲地图

本讲讲什么：注意力机制（Attention）与 Transformer——今天所有大模型共用的那颗“心脏”。**为什么值得学：**你天天在用的 GLM、DeepSeek、ChatGPT，以及你们三个项目里的大模型和立体匹配模型，核心零件都是本讲这一个。**学完你能做什么：**看懂 Q/K/V 与自注意力的每一步矩阵运算，读懂 Transformer 结构图，理解大模型为什么能当 agent 配工具；课后编程题带你把注意力画成热图“亲眼看见”。**课后阅读：**许庆阳《深度学习模型与算法基础》第 204–229 页。

如果你在手机上玩过文字接龙，就知道规则：给几个字，往下补一个最像样的字。其实大模型的底牌就是文字接龙——第 9 讲我们已经见过“今天/天气/很/好”式的逐字预测：模型每一步都在给候选字排概率。但把“接龙”从能接三个字，接到能写代码、做翻译、陪你聊天的 ChatGPT，中间隔着一座大山。这座山的名字，叫**注意力（Attention）**。

2017 年，八位谷歌工程师联名发表了一篇论文，标题读起来就像向旧时代宣战：*Attention Is All You Need*——“注意力就是你所需要的一切”。论文宣告：RNN 也好、第 10 讲苦心设计的三扇闸门 LSTM 也好，做序列任务都可以不要了；一台纯粹由注意力搭起来的机器（就是本讲的主角 **Transformer**），当年就把机器翻译做到了世界第一。近十年后的今天，从你手机里的输入法联想，到数据中心里上万张显卡训练的万亿参数大模型，全是这台机器的徒子徒孙。这一讲，我们就把它的心脏剖开看个明白。

11.1　为什么要注意力：从 Seq2Seq 的“失忆症”说起

先回忆第 9 讲的**序列到序列（Seq2Seq，Sequence to Sequence）**架构：编码器把整句中文“读”进去，压成一个固定长度的向量；解码器再从这个向量出发，一个词一个词地“写”出英文。这套架构漂亮，但有个先天缺陷：**不管输入是 5 个词还是 500 个词，统统要压进同一个固定大小的向量**。5 个词还好，500 个词就像让你听完一整节课、只许记一张便签纸——开头讲了什么，写到后半段早忘了。我们管这叫长句的“失忆症”，它的病根和第 9 讲讲过的梯度消失一脉相承：信息要一个时间步一个时间步地传，传得越远，丢得越多。

再看人是怎么做的。让你把“我昨天在实验室修好的那台相机”翻译成英文，写到 *camera* 这个词的时候，你的眼睛会怎么办？你会**回头看一眼原句**，目光直接落在“相机”两个字上——你并不是靠死记硬背整句话，而是“写到哪儿、回头看哪儿”。人处理长句靠的是这种随取随用的回瞄，而不是一张越塞越满的便签。

2015 年，Bahdanau 等人第一次把这种“回头看”教给了机器：解码器每写一个词，都回头对原句所有词“扫一眼”，按相关度分配注意力，把相关词的信息多拿一点、无关词少拿一点。这一招让长句翻译质量猛涨。但它当时是挂在 RNN 身上的外挂——真正的大戏在 2017 年：既然“回头看”这么好使，**干脆把 RNN 也扔了，只留注意力**。这就是《Attention Is All You Need》的野心。

先补一口数学。好消息是：整台 Transformer 的核心运算，你在第 0 讲就学过了。

先补一口数学：点积、矩阵乘法与 softmax

翻回开课导引（第 0 讲）的向量一节：两个向量的**点积（dot product）**＝对应元素相乘再相加，它衡量两个向量“方向有多一致”——方向越接近，点积越大；互相垂直，点积为零。本讲只需要再往前走一小步：

矩阵乘法就是“把很多点积排成一张表”。A 是 $m\times n$ 矩阵、B 是 $n\times p$ 矩阵，则 AB 是 $m\times p$ 矩阵，其中第 (i,j) 格 $=A$ 的第 i 行与 B 的第 j 列的点积。所以“一批向量跟另一批向量两两点积”这种事，一次矩阵乘法就全算完了。

softmax 是第 3 讲分类输出层的老朋友：把任意一组分数压成一组和为 1 的正数（“百分比”），分数高的占比大。请记住本讲的一个惊人事实：**自注意力从头到尾，就是几次矩阵乘法加一次 softmax，没有别的魔法。**

11.2　Q/K/V 三兄弟：一次“查字典”式的信息检索

注意力把“找重点”这件事拆成了三个角色，江湖人称 Q/K/V 三兄弟：

- **查询（Query，简称 Q）**：我在找什么？——当前这个词发出的“提问”。
- **键（Key，简称 K）**：我有什么标签？——每个词挂在门口的“索引卡”。
- **值（Value，简称 V）**：我实际装的内容。——真正要被取走的信息。

查字典：Q/K/V 到底像什么

你拿着一个问题去翻字典。你的问题是 **Q**；字典里每个词条头上的“词条名”是 **K**；词条正文是 **V**。你不会从 A 到 Z 每条都读——你按“词条名跟我的问题有多相关”来分配眼神：越相关的词条看得越久、记得越多，无关的一扫而过。最后你脑子里的收获，是各条正文按相关度**加权混合**的结果。注意力干的就是这件事，只不过字典不是按字母排序查的，而是按**向量相似度**查的——相似度怎么算？点积。全书到这儿不知第多少次出现点积了，它就是“两个向量有多像”的万能尺子。

具体怎么算？以“小猫追蝴蝶”里的“追”为例：它拿自己的 Q 去和句中每个词的 K 做点积打分，再用 softmax 把分数变成一组和为 1 的“关注度”，最后按关注度对所有词的 V 加权求和——求和结果就是“追”在这个语境下更新后的表示。三步流程如下图。

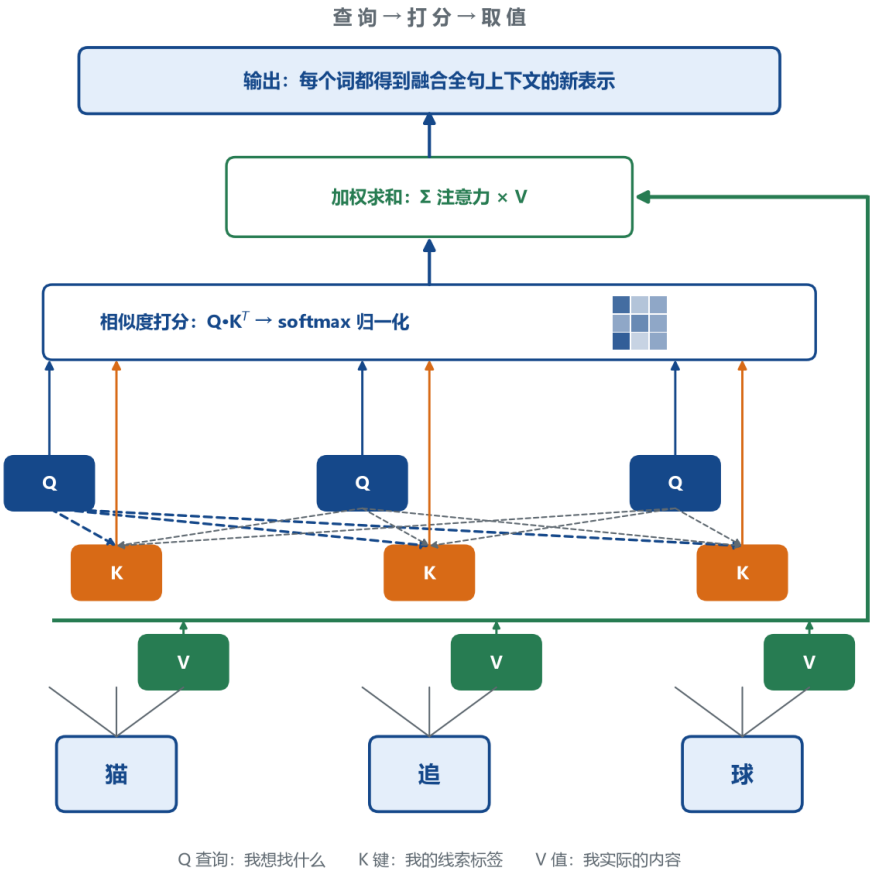


图 11.1　图题：注意力 Q/K/V：查询、键、值三路打分，softmax 归一化后对值加权求和

写成公式只有一行。设 Q、K、V 分别是把全体词的查询、键、值向量各自摞成的矩阵，d_k 是键向量的维度：

$$\text{Attention}(Q,K,V) \;=\; \text{softmax}\!\left(\frac{QK^{\top}}{\sqrt{d_k}}\right)V \tag{11.1}$$

人话翻译：$QK^{\top}$ 一次性算出“每个词对每个词”的相似度大表；除以 $\sqrt{d_k}$ 是给分数降温；softmax 把每行分数变成关注度百分比；最后乘 V，按百分比把大家的内容调和成自己的新表示。一个细节：为什么除以 $\sqrt{d_k}$？维度一大，点积动辄几十上百，softmax 会被“喂饱”成接近 0 或 1 的极端分布，梯度几乎消失；除以 $\sqrt{d_k}$ 把分数拉回温和区间，一句话，防止打分两极分化。

顺带认一下注意力的两大门派。上面这个“点积打分”的版本叫**点乘注意力**（dot-product attention），如今 Transformer 全家用的都是它；而 2015 年 Bahdanau 初版用的则是**加性注意力**（additive attention）——把 Q 和 K 拼在一起，喂给一个小的前馈网络，让它吐出一个相关度分数。打个比方：加性是“请一位小裁判逐对打分”，分打得不算差，但只能一对一对慢慢算；点乘是“全体两两点积，一次矩阵乘法一气呵成”——GPU 天生爱吃这种能全并行的菜。速度碾压之下，点乘成了现役标准，加性则留在教科书和考古现场。

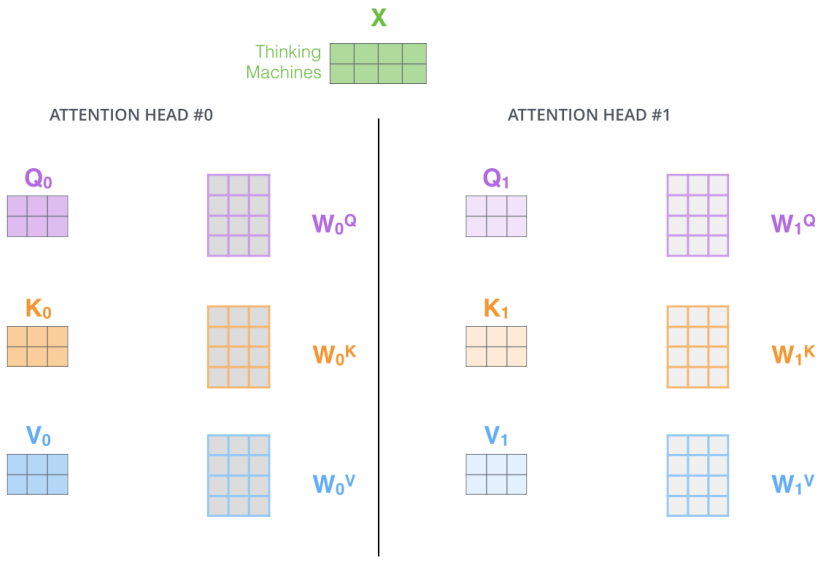


图 11.2　图题：缩放点积注意力的计算流程：打分、掩码、softmax、加权求和

图片来源：来源：Jay Alammar, The Illustrated Transformer 博客文章, 2018

用代码写出来不过十来行，你完全可以亲手跑一遍：

```python
import numpy as np

def softmax(x):
    e = np.exp(x - x.max(axis=-1, keepdims=True))  # 减最大值防溢出
    return e / e.sum(axis=-1, keepdims=True)

def attention(Q, K, V):
    d_k = Q.shape[-1]
    scores = Q @ K.T / np.sqrt(d_k)   # 第1步：点积打分（@就是矩阵乘法）
    A = softmax(scores)               # 第2步：softmax变成"关注度"
    return A @ V, A                   # 第3步：对V加权求和

# 三个词，词向量取2维（示意数据）
Q = np.array([[1, 0], [0, 1], [1, 1]], dtype=float)
K = Q.copy()                          # 自注意力：键就是词自己
V = np.array([[1, 2], [3, 4], [5, 6]], dtype=float)
out, A = attention(Q, K, V)
print(A.round(2))                     # 3x3注意力矩阵，每行和为1
```

跑完你会发现：A 是一个方阵，第 i 行第 j 格就是“第 i 个词分给第 j 个词的关注度”，每行加起来等于 1。下一节我们把这个矩阵画成热图——那是理解注意力最直观的一张画。

纸笔算一次注意力：三个词的“圆桌会议”

不用电脑，拿张草稿纸。设某句话有三个词，查询向量 $q=(1,0)$，三个键、值向量如下：

词	键 k_i	值 v_i
小猫	$(1,0)$	$(1,0)$
追	$(0,1)$	$(0,1)$
蝴蝶	$(1,1)$	$(1,1)$

第一步，算点积打分：$q\cdot k_1=1$，$q\cdot k_2=0$，$q\cdot k_3=1$（本题取 d_k 缩放不改变相对大小，从略）。

第二步，softmax：$\mathrm{softmax}(1,0,1) \approx (0.42,\ 0.16,\ 0.42)$。第三步，加权求和：$0.42\times(1,0)+0.16\times(0,1)+0.42\times(1,1)=(0.84,\ 0.58)$。做完请回答：如果“追”的查询和“小猫”的键更垂直，输出会更靠近谁？这个三步流程，和上面 Python 代码的哪三行一一对应？

知识补齐：训练好的 Q/K/V 到底学到了什么

可能有人要追问：Q、K、V 这三个变换矩阵是哪来的？答案朴素得可能让你失望：**训练开始之前，它们就是三个随机矩阵**——三个刚入职的新员工，谁提问、谁打标签、谁发货，全凭胡来。所谓训练，就是让预测误差当考官：该关注哪个词没做对，就反过来微调这三个矩阵。练成之后，分工才慢慢长出来：每个词的 Q 变成“这个词在查询什么语义”——“它”的 Q 专门发出“我在找我能指代的名词”的询问；每个词的 K 变成“我能提供什么标签”——“猫”的 K 明晃晃写着“我是名词、我是实物、我在前半句”；每个词的 V 则是“我的实际内容”——被别人取走之后，真正融进对方新表示的那部分信息。一句话：**Q/K/V 的角色分工不是设计者规定的，是梯度从数据里练出来的**——下一节热图里“它”指向“猫”的那格最亮，就是练成的证据。

11.3　自注意力：一句话内部的“圆桌会议”

上面的流程里藏着一个关键设定：Q、K、V 全都来自**同一句话**的词向量——每个词既是提问者，又是被查的字典，还是被取走的内容。所以这一招叫**自注意力（self-attention）**：一句话里的所有词同时互相打分、互相取经，像开了一场圆桌会议，每人发言一次，所有人都能按相关度吸收别人的信息。

自注意力一出场，两个老大难当场解决。**第一，距离不再是问题**：第 1 个词和第 100 个词直接握手，不像 RNN 要一站一站传话传 99 站（第 9 讲那张梯度一路衰减的图还历历在目）。**第二，指代消解（coreference resolution）**——代词“它/他/她”到底指谁——变成了一张看得见的表。看下图：“它”那一行，最亮的格子正好落在“猫”那一列：模型不知道语法书上的条文，但注意力分数诚实记录了“它”在向“猫”取经。

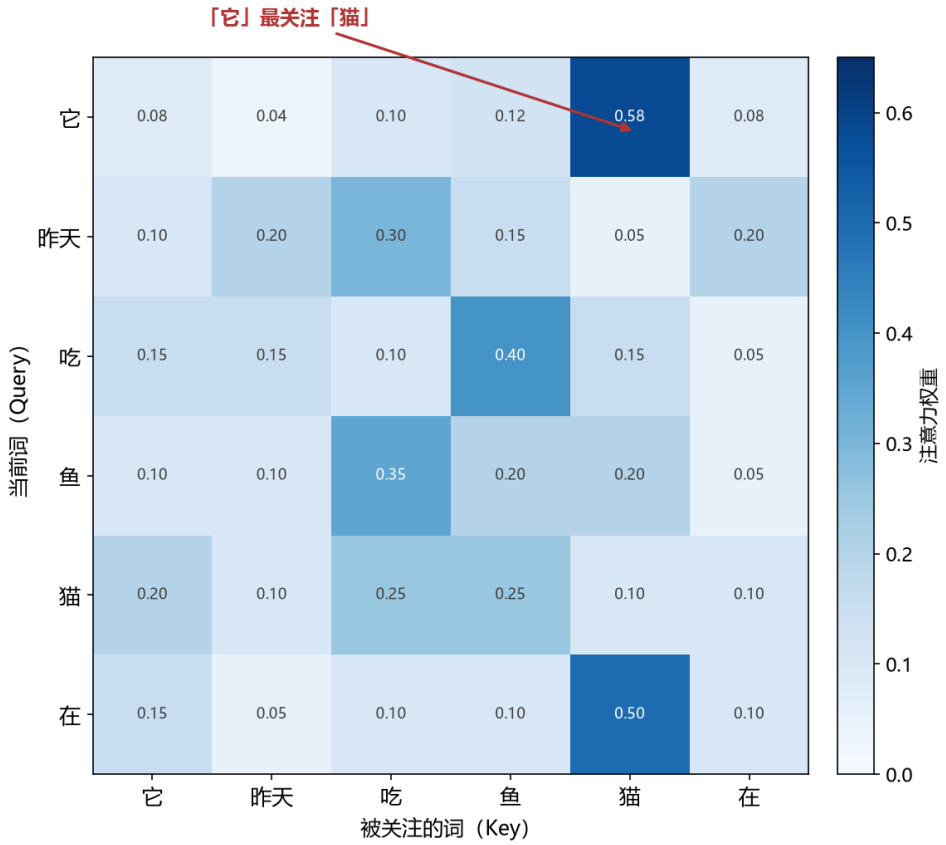


图 11.3　图题：注意力权重热图：“它”那一行最亮的格子指向“猫”，指代消解被看得见地记录下来

这类热图也是调试序列模型的重要工具：哪两个词的联系没被模型抓住，热图上一眼就能看出来。课后编程题就让你亲手画一张。

11.4　多头注意力与位置编码：一支足球队而非一个球星

11.4.1　多头注意力：八支荧光笔各画各的重点

只做一次注意力，等于只用一支荧光笔在句子上画重点——画得再好也只是“一种眼光”。**多头注意力（multi-head attention）**的思路：把词向量切成 8 份，每份独立做一遍 Q/K/V 注意力，这叫 8 个“头（head）”；最后把 8 个头的输出拼接起来再做一次线性变换。实践中发现，各个头真的会**自发分工**：有的头盯着语法结构（主语配谓语），有的头盯着语义指代（“它”配“猫”），有的头只关心相邻词。像你做阅读理解时同时用“找主语”“找指代”“找关键词”三支荧光笔——多支笔并行，互不干扰，最后一起交卷。

11.4.2　位置编码：给天生的“脸盲”贴工位号

但注意力有个天生的缺陷：它**不认顺序**。把“猫追狗”打乱成“狗追猫”，两句话算出来的注意力分数集合一模一样——因为在注意力的世界里，每个词只是一张写着内容的卡片，谁来谁往它根本不关心。这可不行，“猫吃鱼”和“鱼吃猫”是天壤之别。解决办法是给每个词贴一张**位置编码（positional encoding）**的“工位号”，把位置信息掺进词向量再进注意力。三代方案一句话说完：2017 年原版用不同频率的正弦、余弦函数生成位置信号；BERT、GPT-1 用的可学习位置向量（每个位置一个可训练向量，简单粗暴）；如今主流大模型（LLaMA、Qwen、DeepSeek 等）标配的 **RoPE（Rotary**

Position Embedding，旋转位置编码）把位置“旋转”进 Q 和 K 的乘法里，让模型外推到更长的上下文时更稳——知道名字、知道它管什么即可。

11.5　Transformer 全家福：用注意力搭一整栋楼

把上面的零件装配起来，就是 2017 年论文里的 **Transformer**，整体结构见下图。

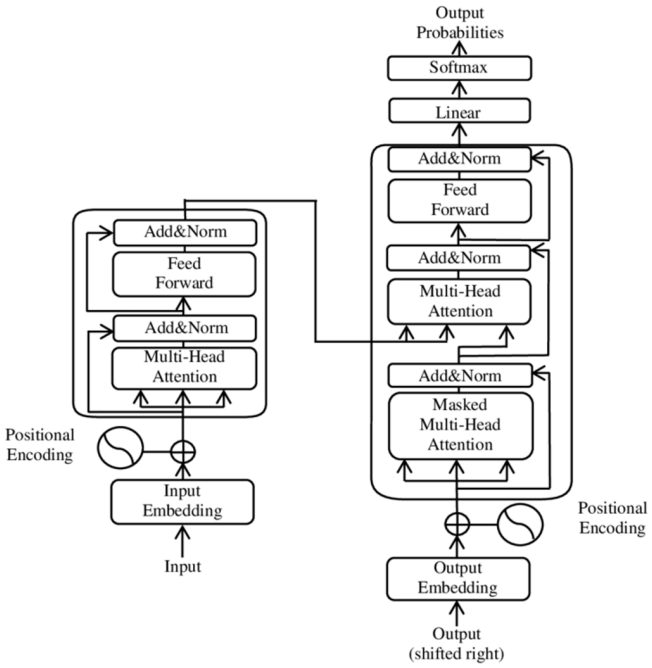


图 11.4　图题：Transformer 整体结构：左侧编码器栈与右侧解码器栈，各由多层相同的块堆成

图片来源：来源：Vaswani et al., Attention Is All You Need, NeurIPS 2017

左边一摞是**编码器（encoder）**，右边一摞是**解码器（decoder）**，各叠 6 层，每层是一个标准化的“块（block）”。拆开一块看看，你会发现全是老朋友：一个多头自注意力子层，加一个逐词的前馈网络 MLP（第 2 讲的全连接），每个子层外面还套着**残差连接**与**层归一化（LayerNorm）**——残差捷径正是第 7 讲 ResNet 那条“高速公路”，让几十层堆得起来；层归一化是第 6 讲批归一化的表亲，改为沿特征维度归一化、不依赖同批样本，对序列任务更友好。一句话记住它：**Transformer 块 = 注意力 + MLP + 两条捷径**。

知识补齐：LayerNorm vs BatchNorm——一个按人、一个按科

第 6 讲学过批归一化（BatchNorm，简称 BN），现在让它和层归一化（LayerNorm，简称 LN）当面比一比。想象一张 3×6 的成绩单：3 名学生，每人 6 门课。**BN 是“按科”统计**：对每一门课，跨 3 名学生算均值和方差，再把这门课的成绩拉齐——它回答“这门课在全班里算高还是算低”。**LN 是“按人”统计**：对每一名学生，跨自己那 6 门课算均值和方差，给自己的成绩单单独拉齐——它回答“这个人的整体水平如何”。差别看着温和，后果却要命：BN 的统计量依赖“同科的一批人”，**batch 一小就失灵**——批里只剩一两个人，“全班均值”毫无意义，推理时单条样本更是无班可依；LN 对每个样本自给自足，**batch=1 也照常工作**，句子长短不齐也毫不在乎。NLP 恰恰是“句子变长变短、batch 时大时小”的重灾区——这就是 Transformer 全线选 LN 的原因。一句口诀收尾：**BN 沿 batch 维统计（按科），LN 沿特征维统计（按人）**。

解码器比编码器多一样东西：**交叉注意力（cross-attention）**——它的 Q 来自解码器自己，K 和 V 来自编码器的输出，相当于翻译时“回头看”原文的那一眼。这一眼的神奇之处在于“对齐”是自己长出来的：把 *Hello World* 翻译成“你好，世界”，解码器写到“世界”这个词时，交叉注意力的目光会稳稳落在 *World* 上，写到“你”时落在 *Hello* 上——传统机器翻译要靠语言学专家手工整理“词对齐表”，这里一套注意力自己就学会了。另外解码器的自注意力是“因果”的：生成第 t 个词时不许偷看第 t 步之后的词，不然就是抄答案。实现上是一张**下三角掩码**：把所有“未来位置”的注意力分数统统写成 $-\infty$，softmax 一算（$e^{-\infty}=0$），这些位置的关注度恰好归零——像一场只许翻看已答页面的考试，后页一律盖住。

按“拆掉哪半栋楼”，后续模型分成三大家族，你天天听说的模型都能对号入座：

表 11.1　Transformer 三大家族：都拆自同一栋楼

家族	代表模型	注意力方式	擅长任务
只用编码器	BERT	双向，前后都看	理解：分类、检索、匹配
只用解码器	GPT系列	因果，只看上文	生成：写作、对话、代码
编码器–解码器	T5、原版	编码读入 + 交叉注意力	翻译、摘要等“转换”

ChatGPT、GLM、DeepSeek 走的都是“只用解码器”路线：模型本质在玩文字接龙，接得足够好，智能就涌现出来了（马上讲到）。

Transformer 还有一张压过 RNN 的王牌：**并行**。RNN 第 t 步必须等第 $t-1$ 步算完，天然串行，GPU 有一身力气没处使；自注意力把“所有词对所有词”的打分变成一两次矩阵乘法，整句话同时算完，显卡满载奔跑。**并行意味着训练快，训练快意味着敢把模型做大**——这就引出了本讲最壮观的情节：堆规模。

11.6　规模定律：从 GPT-1 到 2026 年的大模型

2018年起，OpenAI 沿“只用解码器”路线一路堆料：GPT-1 有 1.17 亿参数，GPT-2 涨到 15 亿，GPT-3 直接跳到 1750 亿，下图是对数坐标下的这条增长线——注意纵轴每格代表 10 倍，图上看着平缓，实际是天文数字式的爆炸。到 2026 年，前沿大模型的参数规模已进入万亿级（且多以混合专家方式组织，见下面的前沿盒子）。

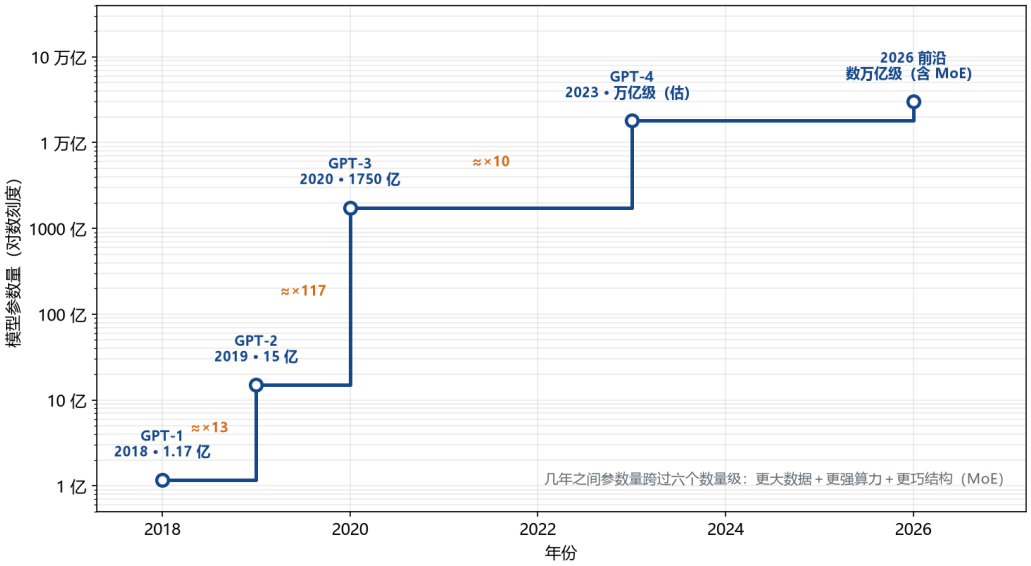


图 11.5　图题：大模型参数量的对数增长时间线：GPT-1 到 2026 年前沿模型

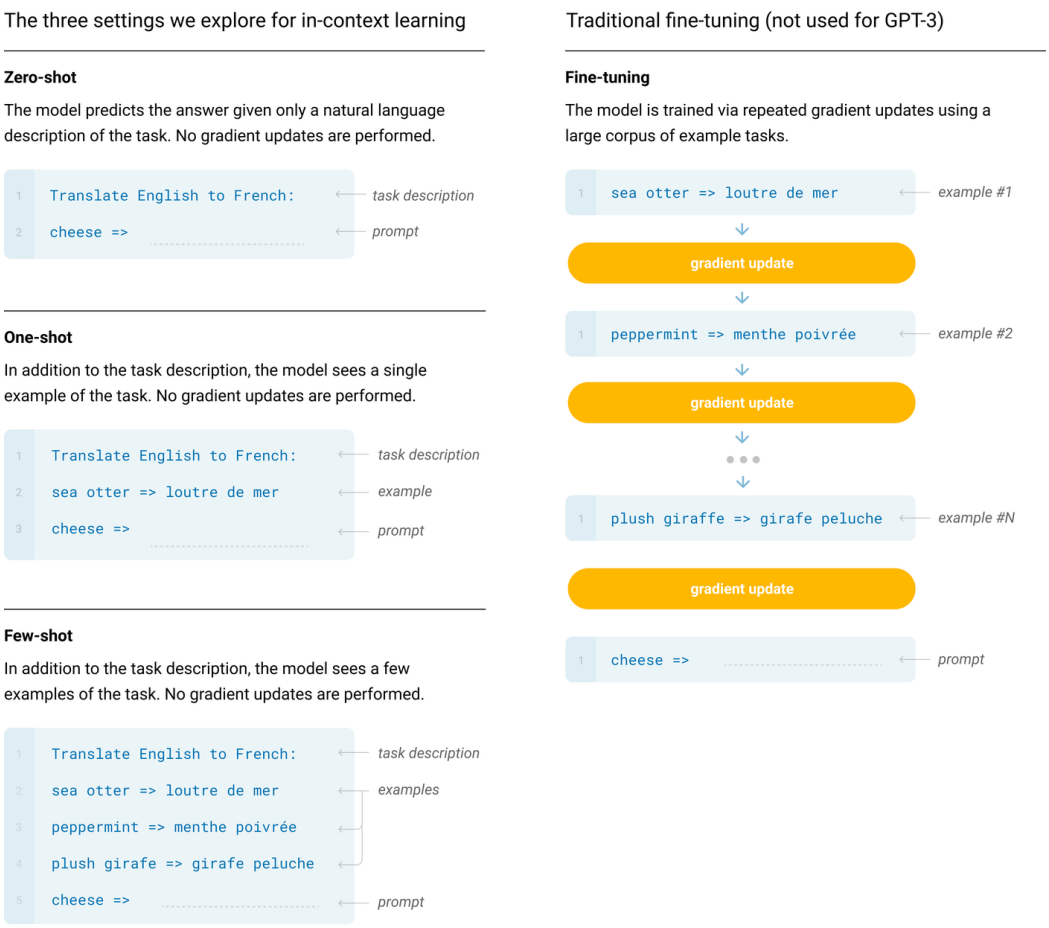


图 11.6　图题：GPT 系列的演化：结构一脉相承，规模逐代暴涨

图片来源：来源：Brown et al., Language Models are Few-Shot Learners (GPT-3), NeurIPS 2020

光堆参数还不够，还得喂够数据和算力。2020 年的规模定律（scaling laws）给出一条冷静的经验规律：只要模型、数据、算力按比例一起加大，模型损失会沿幂律曲线**可预测地**下降——“大力”

真的能出“奇迹”，而且出多少奇迹可以提前算出来。这给了资本下注的底气。更戏剧性的是**涌现能力（emergent abilities）**：参数规模跨过某个坎后，模型突然会做小模型完全不会的事（比如多步推理），量变到质变的机理至今仍是研究热点。

GPT-3 论文还带来一个使用方式的革命：不用再微调，把几个例子直接写在提示词里，模型“照猫画虎”就能上手新任务——这叫少样本学习（few-shot learning）。你在实验课里写的那些提示词，理论源头就是这篇论文。

文本这边的故事讲完了，图像怎么搭上这班车？答案是 2021 年的 **CLIP**。它用 4 亿对“图片 + 文字说明”做对比学习：配对的图和文在向量空间里拉近，不配对的推远。练成之后，CLIP 拥有一种奇特能力——把一张图变成一个和“词语”同类的向量。从此**图像第一次以“词”的身份进入了注意力的世界**：它可以跟文字直接做点积比相似度（零样本分类），可以被当成多模态大模型的一个“词”读进去。今天文生图模型的文本理解、多模态大模型的视觉部分，血统都能追到 CLIP。它是从本讲通向第 12 讲文生图的桥梁。

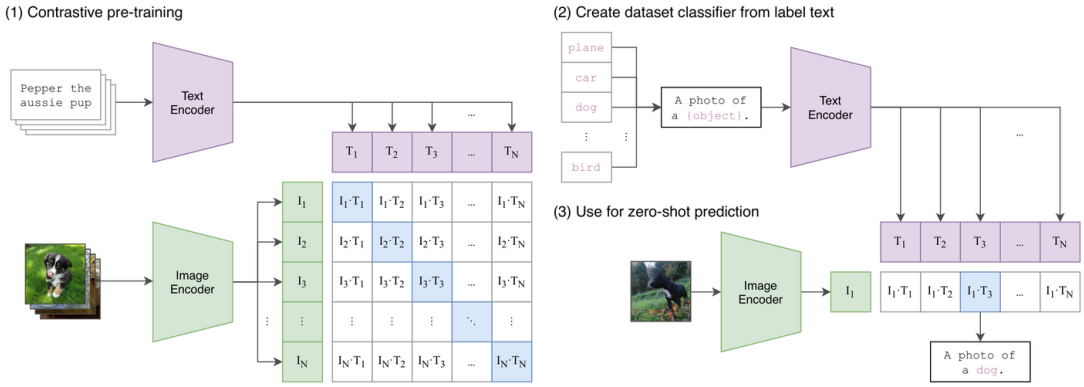


图 11.7　图题：CLIP 的图文对齐：同一空间里“图向量”与匹配的“文字向量”互相靠近

图片来源：来源：Radford et al., Learning Transferable Visual Models From Natural Language Supervision (CLIP), 2021

11.7　Agent：注意力 + 工具 = 会干活的模型

大模型光会“说”还不够，得会“干活”。让模型干活的主流方案叫 **agent（智能体）**：给模型配一套工具（读文件、改代码、执行命令、查数据库），让它在“思考”与“行动”之间循环。以你在实验一（10 月 12 日）要配置的 VS Code 编程助手 Cline 为例，它的 agent 模式是一个循环：**思考**（读任务，规划第一步）→ **调用工具**（输出“我要读某文件”）→ **观察**（框架执行后把结果塞回对话）→ **再思考**（根据观察决定下一步）……直到任务完成。

这个循环为什么成立？关键就在注意力：每一轮决策，模型都要在“系统提示 + 几十轮历史对话 + 工具返回的一大段结果”里**挑出真正要紧的几段**来决定下一步——干的正是自注意力的本职工作。工具清单写进系统提示，模型按格式输出“工具名 + 参数”，框架负责真去执行。所以 agent 没有任何超自然成分：**它就是注意力 + 一份工具说明书 + 一个循环**。第 0 讲那张“思考–行动–观察”循环图，现在你能从原理层面读懂它了。

前沿速递：2024–2026，注意力还在进化

- **长上下文**：前沿模型的上下文窗口已做到百万 token 级，一次读进几十本书。瓶颈在注意力本身：词数翻倍，计算量翻**四倍**（n^2 增长），于是 FlashAttention、稀疏注意力等工程化改进成了标配（名词级了解即可）。

- **混合专家（MoE，Mixture of Experts）**：把一个巨型模型拆成很多“专家”小网络，每个词只激活其中一两个。例如 DeepSeek-V3 总参数 6710 亿，每个词只激活约 370 亿——**模型更大，跑起来反而更省**，是 2025–2026 年前沿模型的标配架构。
- **推理模型**：OpenAI的 o1（2024）、DeepSeek-R1（2025）学会“先想后答”：回答前先生成很长的思维链草稿，用推理时的算力换准确率，数学与编程成绩直逼顶尖人类选手。
- **多模态统一**：文本、图像、音频、视频逐渐被统一成“token”喂进同一个 Transformer——GPT-4o、Gemini 能听能看能说；视频生成模型（如 Sora）把视频切成时空小块当词来“接龙”。

你这门课用的 GLM、DeepSeek，全都长在本讲的族谱上。学完这一讲，你手里等于有一张大模型的心脏解剖图。

项目连线：你的三个项目里全是注意力

- **水下双目（二）**：项目采用的 FoundationStereo（CVPR 2025 最佳论文提名）是一个“用注意力看立体图对”的立体基础模型——它的骨干用注意力在左右两目特征之间反复“找对应点”，正是本讲思想与第 11 讲之外立体匹配任务的合体；你们要做的教师–学生伪标签流水线，蒸馏的正是这个注意力骨干的本事。
- **数据基座（三）**：Gemma 12B本地大模型当问数智能体，挂 8 个白名单工具（查表、算指标、画图……）——架构上就是 11.7 节 Cline 的同款“思考–调用–观察”循环；Granite 语义向量模型把文档变成向量做检索，而“把一切变成向量”正是注意力时代的通用语言。
- **六目拼接（一）**：九个“参数头”轻量路线之外，端到端路线的 RopStitch 引擎内部就含 Transformer 骨干——拼接问题被表述成“六路图像 token 互相做注意力”，全景图是注意力“开会”开出来的。
- **实验一**：你在 10 月 12 日亲手配置的 Cline agent，就是 11.7 节的落地现场。配完回头再看本节，你会有“原来如此”的快感。

11.8　从这一讲到 2026

2017 年那篇论文的标题，今天读来像句预言：注意力确实“就是你所需要的一切”。2026 年的 GPT、Claude、Gemini、Qwen、DeepSeek 全部构建在这一讲的自注意力之上，并且沿着三条线狂奔。**第一条线：上下文长度**。百万 token 上下文从论文指标变成了生产就绪的标配：Gemini 2.5 Pro 标配 100 万 token（企业版 200 万），Claude Sonnet 4 也达到 100 万——一次读进几十本书、一整库代码；撑起它的是前沿盒子里提到的 FlashAttention 这类注意力工程，正是它们把 n^2 的账单压到付得起。**第二条线：Agent 化**。2024 年底 Anthropic 发布开放协议 **MCP（Model Context Protocol，模型上下文协议）**，业界戏称“AI 的 USB 口”——不管什么大模型、什么工具（读文件、查数据库、调接口），都按同一套插口标准对接，一个完整的 Agent 生态沿这个口长了出来；你在实验一（10 月 12 日）配置的 Cline 编程助手，就是这条线上的一员。**第三条线：把钱花在哪**。Agent 干活时，一轮就要在上下文里背上几万 token——单轮上下文可超 5 万 token、不缓存时一轮成本 0.05–0.20 美元；所以“上下文管理”与“缓存”成了大模型工程的核心手艺，相当于 11.7 节那个思考–行动循环的水电费账本。

你此刻与任何一个大模型的每一次对话，都在这一讲的数学里运行。这也是全书最值得你“常回来看看”的一讲——大模型的下一页历史，多半还写在这颗心脏上。

第 11 讲课后编程题（用 AI 实现）

题目：做一个“注意力可视化器”，亲眼看见自注意力。取句子“小明把书放回书架因为它看完了”，用 HuggingFace 的 `transformers` 库加载 `bert-base-chinese`，抽取某一层某几个头的自注意力权重（没有网络条件时，可让 AI 生成一个合理的 6×6 模拟注意力矩阵代替），用 matplotlib 画成热图（行列标注分词结果），找出“它”那一行最亮的格子，并用 50 字以上解释：这一格为什么亮？它符合什么语言直觉？（提示：指代消解。）

交付物：代码文件 `attn_vis.py` + 热图运行截图 + 50 字以上“我改了什么、为什么”。

提示词建议：

```
请帮我用 Python 写一个"注意力可视化器":
1. 用 transformers 加载 bert-base-chinese，输入句子
   "小明把书放回书架因为它看完了";
2. 取第 8 层的自注意力权重（outputs.attentions[7]）;
3. 挑一个注意力头，用 matplotlib 画 6x6 热图，
   行列标签用 tokenizer.tokenize 的分词结果;
4. 自动找出"它"那一行权重最大的格子，打印对应的词对。
请用中文注释解释每一步，把功能拆成小函数，方便我修改句子和层数。
```

验收线：代码能跑出热图；你能向同学讲清楚 Q、K、V 在这段代码里各对应哪个张量、热图的一行和一列各是什么意思；至少有一处你自己的修改（换句子、换层、把最亮格子框出来）。“代码是 AI 写的没关系，讲不清楚就是没做。”

加分项：换三个不同的句子（比如“它”真正指“书”的句子），对比最亮格子的移动；或把 12 层注意力逐层画出来，观察“它”的指向在第几层变得清晰。

注意力机制（Attention）

让模型对输入的不同部分按相关度分配“关注份额”的机制，本质是相似度打分、归一化、加权求和三步。

Query / Key / Value

注意力的三个角色：查询（我在找什么）、键（我的标签）、值（我实际的内容），类比问题、词条名、词条正文。

缩放点积注意力

用点积打分、除以 $\sqrt{d_k}$ 防止 softmax 饱和的标准注意力公式，Transformer 的基本零件。

自注意力（Self-Attention）

Q、K、V 全部来自同一序列，句子内部互相打分取经；并行计算，任意距离直接握手。

多头注意力（Multi-Head Attention）

把向量分组并行做多遍注意力，各个头自发分工盯语法、语义、邻近等不同角度。

位置编码（Positional Encoding）

给无序的注意力补上顺序信息，方案有正弦编码、可学习向量与主流的 RoPE 旋转位置编码。

Transformer

完全由注意力 + MLP + 残差与层归一化搭成的网络，分编码器、解码器两摞；今天所有大模型的底座。

token 与上下文窗口

大模型处理文本的最小单位（词或字块）；上下文窗口是一次性能“看进去”的最大 token 数。

规模定律（Scaling Laws）

模型、数据、算力成比例放大时，损失沿幂律可预测下降的经验规律，是“大力出奇迹”的理论依据。

涌现能力（Emergent Abilities）

规模跨过某个坎后突然出现的小模型不具备的能力，机理仍是开放问题。

混合专家（MoE）

巨型模型拆成多个专家网络，每个词只激活一小部分，做到“总参数巨大、单次开销很小”。

CLIP

用 4 亿图文对训练的图文对齐模型，把图像变成与文字同空间的向量，是多模态时代的桥梁。

第 12 讲　生成对抗模型

本讲地图

本讲讲什么：生成对抗模型（GAN）与它的“后浪”扩散模型——AI从“认东西”走向“造东西”的两代引擎。**为什么值得学：**文生图、文生视频、深度伪造与鉴伪、你们数据基座项目里每周在跑的“造假者–监考–判官”对抗训练，全是本讲思想的现场。**学完你能做什么：**看懂 GAN 的极小极大博弈与训练拉锯，说出 GAN 家族的演进脉络，理解扩散模型为什么接管了生成任务，并对“眼见为实”保持工程师式的警惕。**课后阅读：**许庆阳《深度学习模型与算法基础》第 230–248 页。

先讲一个老故事。伪钞犯印刷假钞，越印越像；警察学习鉴定，越查越严。每破获一批假钞，伪钞犯就改进工艺；每升级一次工艺，警察就升级装备——**双方在对抗中共同进化**，最后假钞逼近真钞，警察练就火眼金睛。2014 年，蒙特利尔大学的博士生 Ian Goodfellow 把这场军备竞赛写进了一个损失函数，论文叫 *Generative Adversarial Nets*（生成对抗网络，缩写 GAN）。据他本人后来在访谈里说，这个想法诞生于和同学在酒吧的一次神聊——“让两个网络互相欺骗，不就都能变强吗？”——他回家当晚就把代码写了出来，第一晚就跑通了。深度学习巨头 Yann LeCun 曾评价它是“近二十年深度学习最酷的想法”。

这一讲是全书的压轴之一。前十一讲里的网络几乎都在“认”世界：给图，吐类别。从这一讲起，网络要学会“造”世界：给噪声，吐图像。而“造假”与“鉴伪”这对矛和盾，会一路延伸到你毕业时都将面对的现实问题——深度伪造。

12.1　生成模型与判别模型：无中生有 vs 看图说话

先把两类模型分清楚。**判别模型（discriminative model）**做的是“看图说话”：输入一张图，输出类别——它是第 0 章到第 10 章的全部主角。它在学“图和类别的对应关系”，数学上记作 $p(y\,|\,x)$：给定图 x，类别 y 的概率。**生成模型（generative model）**做的是“无中生有”：输入一个随机的噪声向量，输出一张图。它要学的是数据本身的样子，即 $p(x)$——世界上所有猫照片共同服从的那张“概率分布地图”，然后从地图上任一处“采样”出一张全新的、训练集里从来没有过的猫。

什么叫“学会数据的分布”

说模型“学会了猫的分布”，意思是：它脑子里有一张“猫概念地图”，地图上每个坐标对应一张可能的猫图（耳朵角度、毛色、姿态……），靠近地图中心的坐标是“最典型”的猫，边缘是罕见的猫。所谓“生成一张图”，就是在地图上随手戳一个点，把那个点“翻译”成像素。“这张图像不像真的猫”，就看它是否落在地图上概率高的区域。生成模型千千万，目标都是一张：把这张地图学到手。

这套“学分布”的思路在数学上站得住吗？站得住。神经网络理论里有一条万能逼近式的底气：**任一概率分布，都可以被一个足够大的可微函数逼近到任意程度**——分布无非是一张从“样本坐标”到“概率密度”的对应关系，而“逼近对应关系”正是深度网络的本行。所以“学一个网络去生成符合分布的样本”不是玄学，是数学上有保底的工程路线——本讲的 GAN、稍后的 VAE 与扩散模型

走的全是这条路，区别只在用什么手段逼网络把分布学像。

表 12.1　判别模型与生成模型对比

	判别模型	生成模型
一句话	看图说话	无中生有
输入 → 输出	图像 → 类别	噪声向量 → 图像
学习目标	类别边界 $p(y\mid x)$	数据分布 $p(x)$
代表	CNN、ResNet、ViT	GAN、扩散模型
典型用途	识别、检测、分割	文生图、数据增广、鉴伪攻防

其实你在第 4 讲已经和生成模型打过照面：自编码器的解码器能从隐向量重建图像。但它只会“照着练过的样子修图”，不太会凭空造出全新的、多样化的样本。真正把“无中生有”做成一门艺术的，是接下来这场对决。

12.2　GAN：造假者与鉴定师的对决

GAN的架构里有两个网络，天然是对手（见下图）：

- **生成器（Generator，记作 G）**：造假者。输入一个纯噪声向量 z（比如 100 维、从标准正态分布随机采样），输出一张假样本。它从一开始就没人教过“猫长什么样”，全靠造假谋生。
- **判别器（Discriminator，记作 D）**：鉴定师。输入一张图（真的或假的），输出一个 0 到 1 之间的分数——它有多确信这是真货。

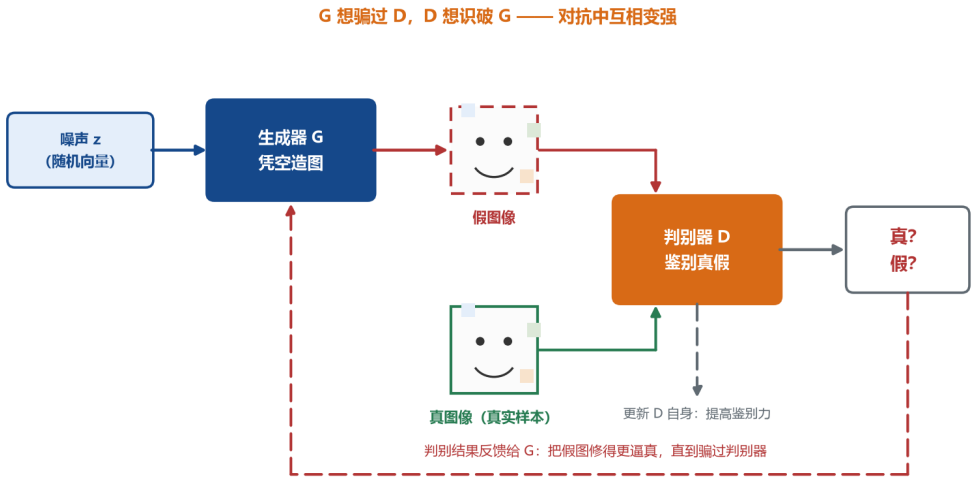


图 12.1　图题：GAN 架构：生成器把噪声变成假图，判别器给真假打分，双方在对抗中共同进化

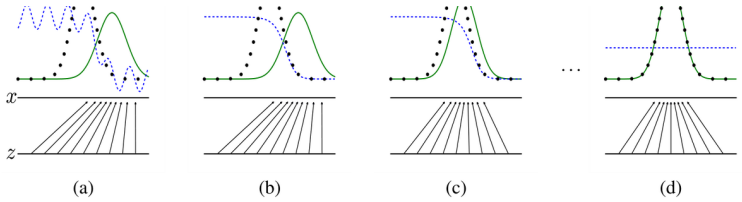


图 12.2　图题：GAN 论文原始框图：生成空间到样本空间的映射与判别器的双向夹击

图片来源：来源：Goodfellow et al., Generative Adversarial Nets, NeurIPS 2014

训练时两边同时开工：D 拿真图和 G 的假图混在一起练鉴定；G 拿 D 的鉴定结果当反馈，专门朝“能骗过 D”的方向改进。这盘博弈写成目标函数只有一行：

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}\big[\log D(x)\big] + \mathbb{E}_{z\sim p_z(z)}\big[\log\big(1-D(G(z))\big)\big] \tag{12.1}$$

人话翻译：别被符号吓住，两边各说各的话。后半句里，D 想让 max 成立——真图 x 判成 1（$\log D(x)$ 越大越好），假图 $G(z)$ 判成 0（$1-D(G(z))$ 越大越好）；前半句里，G 想让 min 成立——它生产的假图要是能骗得 D 打出高分 1，$1-D(G(z))$ 就趋近 0，D 一无所获。一句话：**D 想识破 G，G 想骗过 D，双方在同一个函数里较劲**。

这场博弈的终点在哪？博弈论早有答案（见下面的盒子）：理想的平衡点是生成分布 p_G 与真实分布 p_{data} 完全重合——假样本以假乱真，D 再也无从下手，只能对一切图都输出 $\frac{1}{2}$，闭着眼抛硬币。这时候造假这门手艺就“毕业”了。

博弈论一分钟：纳什均衡

石头剪刀布的最优策略是什么？以 $\frac{1}{3}$ 的概率随机出拳。一旦你偏科（比如爱出石头），对手就能针对性克制你。这个“谁单方面改变策略都占不到便宜”的状态，就是**纳什均衡（Nash equilibrium）**——博弈论经济学家约翰·纳什提出的概念，他的故事被拍成电影《美丽心灵》。GAN 的训练就是在两个网络的策略空间里找纳什均衡。麻烦在于：第 3 讲训练分类网络是“一个人下山”，沿梯度往下走就有唯一谷底；GAN 是“两个人拔河”，你往东他往西，**没有现成的下山路线**——这正是下一节训练拉锯战的根源。

12.3　训练：一场没有裁判的拉锯战

GAN的训练循环是交替进行的：每批数据先更新 D 几步，再更新 G 一步，如此往复。核心代码骨架值得背下来：

```
for epoch in range(50):
    for real_imgs, _ in loader:
        # ---------- 第一步：更新判别器 D ----------
        z = torch.randn(len(real_imgs), 100)     # 采一批噪声
        fake = G(z)                              # 生成器造假
        d_loss = loss(D(real_imgs), ones) \
               + loss(D(fake.detach()), zeros)  # detach: 这笔账不记到G头上
        opt_D.zero_grad(); d_loss.backward(); opt_D.step()

        # ---------- 第二步：更新生成器 G ----------
        g_loss = loss(D(G(z)), ones)             # G的唯一目标：骗过D
        opt_G.zero_grad(); g_loss.backward(); opt_G.step()
```

三处细节：`detach()` 把假图从 G 的计算图里摘出来，让“批评 D 判错假图”这笔账不会顺着梯度算到 G 头上；更新 G 时用的是 `loss(D(G(z)), ones)`——G 唯一的目标函数就是“让 D 把我的假货判成真货”；两边各用各的优化器（`opt_D`、`opt_G`），互不干扰。

那么 GAN 的 loss 曲线长什么样？见下图：两条曲线互相咬合、上下拉锯，不像第 3 讲那条平滑下降的坦途。**这不是 bug，是常态**——D 的 loss 降了说明鉴定变严，G 的 loss 随之升了说明造假受阻、于是奋起直追，反过来又把 D 的 loss 顶上去。GAN 的 loss 只是“战况晴雨表”，你看不出“收敛”没有，只能看出“谁暂时占上风”。所以 GAN 训练好不好，历史上主要靠人眼看生成样本。

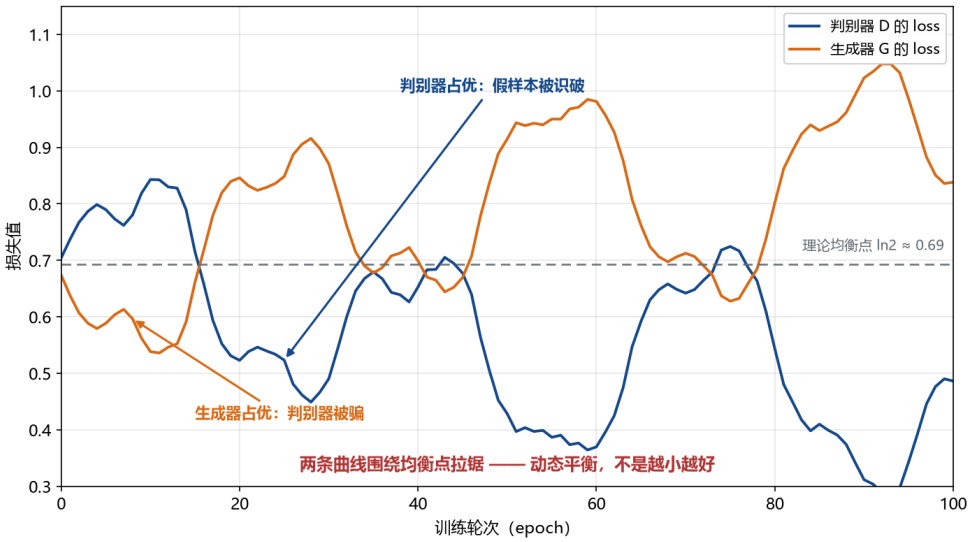


图 12.3　图题：GAN 训练拉锯：生成器与判别器 loss 交替波动，谁也压不死谁

拉锯战里最著名的翻车现场叫**模式崩塌（mode collapse）**：G 某天发现某一种猫总能骗过 D，于是它干脆不再冒险，把所有噪声都映射成这一种猫——**多样性死亡**，你想要一千种猫，它只会画一种。工程上还有一整套止血技巧，这里名词级过三句：一是**交替更新**上多让 D 走一两步，别让鉴定师先垮；二是**标签平滑（label smoothing）**，把真标签 1 改成 0.9，别让 D 变得过度自信、一句话判死；三是**谱归一化（spectral normalization）**，给 D 的权重矩阵加“限速带”，防止它一朝练成就能一刀封喉。后来的 WGAN 则干脆换了把更顺滑的“尺子”来度量两个分布的距离，让 loss 终于有了参考价值。记住：调 GAN 曾经是深度学习里公认的手艺活——这也是它后来被扩散模型取代的伏笔。

12.4　GAN 演进长廊：从模糊噪点到以假乱真

2014到 2018 年，GAN 家族五年间跑完了一场进化冲刺，几个里程碑如下表。

表 12.2　GAN 家族演进速览

年份	名字	一句话贡献
2014	GAN	开山之作：博弈式生成，结构还是全连接
2014	条件 GAN（cGAN）	给 G 加“点菜”条件：要哪个类、要什么文字
2015	DCGAN	卷积版 GAN：训练首次稳定，成为标配基线
2017	CycleGAN	无配对数据域迁移：马变斑马、照片变油画
2018–2020	StyleGAN 系列	风格分层注入：高分辨率人脸可控生成

DCGAN 把 G 和 D 全部换成了卷积结构——G 用转置卷积（第 5 讲卷积的“反操作”，把小图逐步上采样放大）从噪声“画”出图像，D 用普通卷积做鉴定，再配上批归一化等稳定训练的手段，GAN 第一次稳定地在图像任务上开花。**条件 GAN** 在输入里拼接一个条件（类别标签、文字描述），生成从“抽盲盒”变成“点菜”。**CycleGAN** 最神奇：不需要“同一匹马的同角度照片”，仅靠两堆毫无配对关系的图片（一堆马、一堆斑马），靠“过去再回来必须还是你”的循环一致性损失，学会了两个世界之间的翻译——马变斑马、夏天变冬天、照片变莫奈。**StyleGAN** 则把生成过程分层控制：先定大格局（脸型），再定中层次（五官），最后补细节（雀斑、发丝），一路生成分辨率与逼真度都惊人的人脸。



图 12.4　图题：GAN 生成人脸的五年进化：从 2014 年的模糊噪点到 2018 年后的以假乱真

图片来源：来源：NVIDIA StyleGAN 2 官方样张（NVlabs/stylegan2）

上图是这场进化的直观成绩单：2014 年论文里的手写数字模糊得像隔着毛玻璃，2018 年 StyleGAN 的人脸已经能在视觉上骗过人类——请注意其中几张是真照片、几张是生成的，你未必分得出来。顺便回答一个常见疑问：StyleGAN 生成的人脸之所以一张一个样，正因为 G 学到的是整张“人脸分布地图”而不是某几张脸——这是对模式崩塌的反向印证。

12.5　另一条路：变分自编码器（VAE）

GAN 在五年进化冲刺的同时，另一条更“文静”的生成路线也在悄悄长大——**变分自编码器**（Variational Autoencoder，VAE，2013 年提出，比 GAN 还早一年）。回忆第 4 讲的自编码器：编码器把图片压成一个隐向量，解码器再从隐向量还原。它离“生成”只差一步，却是最致命的一步：那个隐空间是“碎”的——只在训练见过的那几个点附近能重建，随手戳一个新点，解码器吐出来的多半是乱码。

VAE 的药方一句话：**把“一个点”换成“一团云”**。编码器不再把一张图压成一个坐标，而是压成两个向量——**均值**与**标准差**，等于给这张图画了一份“模糊画像”：它大概住在哪个片区、活动半径多大；然后从这团云里**随机采一个点**，交给解码器还原。“模糊画像 + 随机采样”两个设定，逼得隐空间处处连续、片片相接——随便戳一个点，解码器都见过长得差不多的邻居，都还原得出一张合理的图。福利随之而来：在两张图的云之间连一条线、沿线戳点，解码器会给你一串“从猫渐变到狗”的过渡帧——隐空间可以做插值了。

训练 VAE 的损失函数是两部分相加。一部分是**重建误差**：还原的图要像原图（第 3 讲那些回归损失的老朋友）；另一部分是 **KL 距离**：把每张图的云往**标准正态分布**上拉，不许某张图的云飘到天南海北、跟别人老死不相往来。一句人话：**还原得要像，云还得守纪律**。代价也藏在纪律里：为了让隐空间处处“合理”，VAE 生成的图天然带一层“平均脸”式的糊——它总在给“什么位置该有什么”打安全牌，锐度让位给了稳健。至于名字里的“变分”，来自统计学的变分推断，你只需记住它管的事：用“均值加标准差的云”去近似真正的数据分布——名词级了解即可。

把两条路线并排一摆，性格差异一目了然：

表 12.3　VAE vs GAN：两种生成哲学

	VAE	GAN
生成质量	偏糊：“平均脸”式的安全牌，锐度不足	犀利：细节逼真，但可能模式崩塌
训练	稳：损失函数一条、普通反向传播，loss 曲线老实下降	难伺候：双人拔河、loss 拉锯、要一整套止血技巧
本质	学分布的**参数**：把每张图编成“均值 + 标准差”，再约束它向标准正态看齐	学一场**博弈**：在真假对抗中把生成分布逼向真实分布
2026 年的余脉	潜空间基建活在扩散模型里（下一节）	判别器思想活在鉴伪检测器里（本章末节）

GAN 与 VAE 之外，生成家族其实还有第三条路，和第 11 讲的接龙一脉相承——**自回归模型**（autoregressive model）：像写字一样一个一个往外“蹦”。PixelRNN、PixelCNN 把图像按行列排好顺序，从左上角第一个像素起，逐像素预测下一个像素；GPT 式的图像模型则把图切成小块当 token，逐 token 往外接。文字接龙接的是字，图像接龙接的是像素块——任务设定完全同构，变的只是接龙的对象。

临收尾再记一笔，供你在 2026 年认亲：Stable Diffusion 全名里的 *Latent*（潜空间），指的正是先由一套 VAE 把图片压进低维潜空间、再在潜空间里做扩散去噪——**2026 年扩散模型的潜空间技术，直系祖先就是 VAE**。这条路线没有死，它转岗当了扩散时代的基建。

12.6　后浪：扩散模型

GAN统治图像生成五六年后，2020 年一种老思路新包装的模型悄悄崛起，2022 年全面接管，它就是**扩散模型（diffusion model）**。它的核心想法朴素得惊人，分两步：

1. **前向过程（加噪）**：拿一张真实图片，一小步一小步地往上撒高斯噪声，撒 1000 步，直到原图彻底淹没成一片纯噪声。
2. **反向过程（去噪）**：训练一个网络（通常是 U-Net），给它看“加噪到第 t 步的图”，让它预测这一步撒进去的噪声是什么——预测对了就把噪声减掉。训练好之后，从一张纯噪声出发，网络一步步把噪声“雕刻”掉，图像便从混沌中浮现。

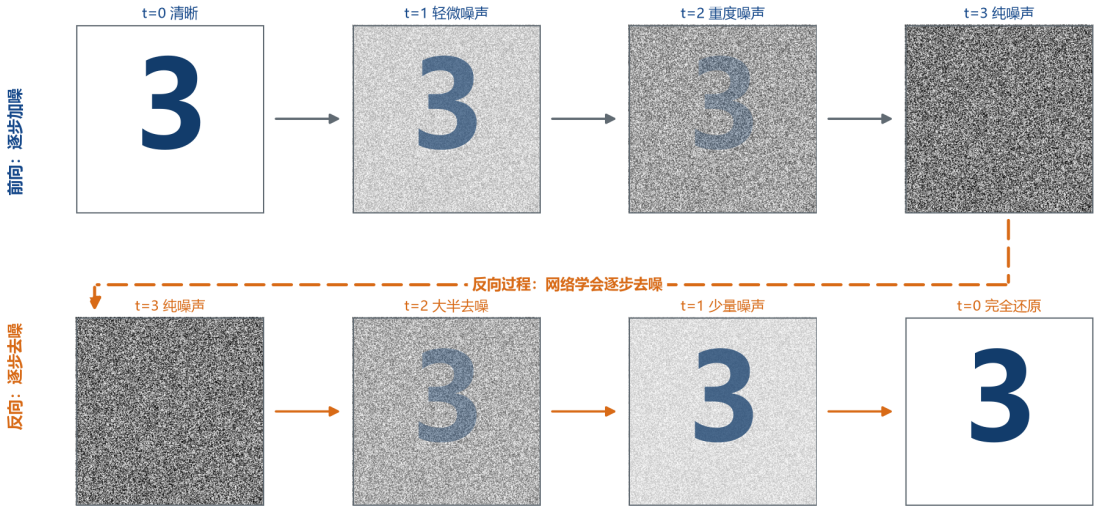


图 12.5　图题：扩散模型：前向过程逐步加噪直至纯噪声，反向过程学会一步步去噪还原

训练目标更是朴素到令人发笑：预测的噪声和真实撒的噪声算一个均方误差——就是第 3 讲最古老的回归损失。没有博弈、没有拔河，loss 老老实实往下降。下图把整个流程摊开给你看。

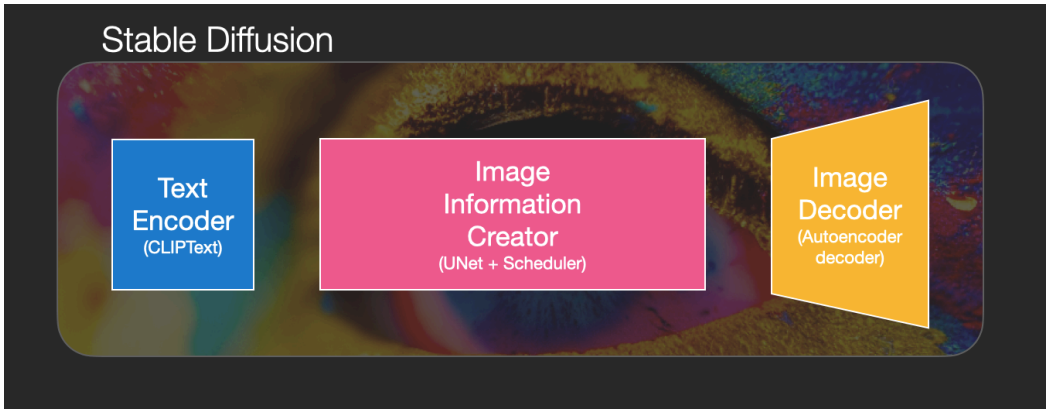


图 12.6　图题：Stable Diffusion 流程：文本编码器理解提示词，U-Net 在隐空间里迭代去噪，解码器还原成图

图片来源：来源：Jay Alammar, The Illustrated Stable Diffusion 博客文章, 2022

2022 年，Stable Diffusion 和 Midjourney 相继出圈，“一句话生成一张画”从此进入大众生活；2024–2026 年，同样的技术沿着时间轴扩展，文生视频（Sora、可灵等）成了新战场。扩散为什么能取代 GAN 的主流地位？两个原因：**训练稳**——没有对抗博弈，loss 曲线终于能看懂，调参不再靠玄学；**模式覆盖全**——以拟合数据分布为目标，天然不易模式崩塌，多样性好。代价也明摆着：**采样慢**——GAN 一次前向就出图，扩散要跑几十步去噪迭代。近两年蒸馏与一致性模型正把这几十步压到几步甚至一步（名词级了解），速度差距在快速弥合。还有个彩蛋：翻看 Stable Diffusion 的结构图你会发现，它的去噪网络里塞满了自注意力层，文本理解靠的是 CLIP——**第 11 讲的注意力，在这里一个都不少**。全书到这里合流了：识别靠它，生成也靠它。

12.7　深度伪造与鉴伪：攻防共生（工程伦理专场）

技术从实验室走进社会，一定是矛与盾一起长大的。**深度伪造（deepfake）**是生成模型最刺眼的社会副作用：换脸视频把明星的脸安到别人身上，声音克隆用几秒钟素材就能模仿一个人，电信诈骗已经出现“冒充熟人视频通话”的真实案例。与之对抗的**鉴伪检测器**，本质上就是本讲的判别器再就

业：在真图与 AI 图之间画边界，专找生成内容留下的指纹——发丝边缘的涂抹感、眨眼节奏的不自然、眼镜框在镜片两侧不对称、背景文字的扭曲。



图 12.7　图题：深度伪造真伪对比：左侧换脸痕迹与右侧真实人脸的细微差异是检测器的切入点

图片来源：来源：网络公开的深度伪造真伪对比样例（FaceForensics++ 等检测研究常用）

更深一层的攻防是**对抗样本（adversarial example）**：在图片上叠加一层人眼几乎看不见的扰动，就能让训练好的识别模型把熊猫认成长臂猿——攻击者用“生成一小段扰动”的思想制造模型盲区；防御方则把这些对抗样本加进训练集重新训练，让模型提前“见过世面”，这叫**对抗训练（adversarial training）**，相当于给模型打疫苗。造假与鉴伪、攻击与加固，是同一枚硬币的两面，谁也消灭不了谁，只能螺旋上升——理解了这一点，你就理解了本讲标题里“对抗”二字的分量。

工程伦理：造假的能力越大，克制的分量越重

三条红线，请当作职业操守记住：第一，**不制作、不传播**深度伪造内容去冒充他人、伪造证据，这在我国已明确触碰法律红线——2025 年起施行的 AI 生成合成内容标识办法要求 AI 生成内容必须“亮明身份”；第二，做工程时**不许用生成数据冒充实测数据**交差，实验一至实验四的所有证据都必须是“命令 + 日志 + 产物”三件套，生成的数字再漂亮也是假的；第三，**对“眼见为实”保持怀疑**是新一代工程师的基本素养——你看到的视频、听到的声音，都可能是某台服务器上一次前向传播的输出。能力越强，越要自觉。

当一回“鉴定师”：给 AI 图找破绽

打开任一文生图工具生成一张“街景人物”图片（或从网上找一张疑似 AI 图），和同学互相挑破绽，比谁先找到三处。经典破绽清单：手指数量与关节；牙齿的排列；眼镜框在镜片两侧是否连贯；招牌和路牌上的文字是否成字；耳朵左右是否对称；背景里栏杆、砖缝这类直线是否突然扭曲。玩完想一想：你刚才做的事，就是把“人肉判别器”的注意力分配到纹理与几何的异常上——鉴伪检测器做的同一件事，只不过它的“直觉”是几百万个卷积核。你也可以反过来玩：把找到的破绽写成提示词让 AI 重画，看它能否骗过你——恭喜，你刚体验了一轮极小极大博弈。

前沿速递：2024–2026 生成式 AI 全景

- **进入工业流水线**：文生图（Midjourney、Stable Diffusion 3、Flux）用于广告与概念设计，文生视频（Sora、可灵、即梦）用于影视预演与短视频量产，文生 3D 开始给游戏和数字孪生喂资产——“生成”从玩具变成了生产力工具。
- **世界模型（world model）**：让模型从海量视频里学“物理规律”，预测世界的下一帧——物体该往哪掉、水花怎么溅。它被视为机器人与自动驾驶的“训练 simulator”，NVIDIA Cosmos 等已开源（名词级了解）。
- **鉴伪与水印在追赶**：C2PA内容凭证标准给图像盖上“出生证明”式的元数据链，Google 的 SynthID 把不可见水印嵌进生成的像素，平台强制标识 AI 内容——生成能力越强，溯源技术越是刚需，而两者至今都在“追赶者”的位置。

一句话总结这个时代：**生成的成本趋近于零，信任成为最贵的资源**。

项目连线：对抗训练每周都在你的项目里跑

- **数据基座（三）**：项目的“虚假内容识别与对抗训练”模块，就是 GAN 思想在数据治理上的变体：大模型（DeepSeek）当“造假者”，按它学到的真实数据模式批量生成 26 个数据家族的假数据；规则引擎加小模型当“监考”，逐条判定真假；LLM 当“判官”复核争议样本；被判定“以假乱真”的样本回流当训练集并加 3 倍权重——造假者下周就学会绕开旧监考，监考每周升级。这场军备竞赛真的在你们的实验室里每周运转，demo 界面与曲线见第二篇项目章。
- **六目拼接（一）与水下双目（二）**：坏帧（黑帧、拖影、过曝）和浑浊的水下雾，可视为真实世界的“生成式污染”——把这类样本主动加进训练集做鲁棒性训练，与对抗训练是同一个思想：先见假，后识真。
- **串起三个项目**：数据基座里的“监考/判官”是判别器思想，数据基座里的问数智能体是第 11 讲的 agent，水下项目的 FoundationStereo 是第 11 讲的注意力骨干——你会发现全书没有一章是孤立的。

12.8　从这一讲到 2026

生成式模型在 2024–2026 年完成了一次“换代”，三条新闻值得你记住。**第一，训练范式从扩散转向流匹配**：**流匹配**（Flow Matching，工程上常用其特例**整流流** Rectified Flow）把扩散那套“一千步逐步去噪”换成了“直接学一条从噪声到数据的直线路径”——训练更快、更稳，采样步数更少；它

已是 2025–2026 年图像与视频生成的主流训练范式，还扩展到了语音，另有 ProReflow 等工作把多步扩散蒸馏成几步甚至一步生成。**第二，视频生成进入群雄时代**：Sora 2 与可灵（快手 Kling 2.5/2.6）、Veo、Runway、Seedance 同台竞技，可灵 2.6 还开放了**运动控制 API**——把真人演员的动作直接迁移到生成角色身上，“文生视频”正从玩具走向生产线。**第三，“眼见为实”的防线在重建**：美国《Take It Down Act》（2025 年签署）把非自愿深度伪造**入刑**、并要求平台限时删除；欧盟 AI Act 强制 AI 生成内容明确标注；C2PA 内容凭证（给图像盖“加密签名的出生证明”）开始被监管采纳——但《华盛顿邮报》实测发现，8 个主流平台里有 7 个会在上传时**剥离**这些凭证。攻防远未收官，结论朴素：**溯源、检测、立法，三管齐下，缺一不可**。

所以，这一讲的博弈论教你看懂的是 2026 年最重要的一场工程攻防：造假者用生成模型武装到牙齿，而鉴定师、立法者与密码学家正用同一套“对抗”思想筑墙——“深度伪造与鉴伪”一节的鉴伪检测器、前沿盒子里的水印与凭证、上面这些法律条文，都是这场博弈的新玩家。你将来选择站哪一边，都得先懂这一讲。

第 12 讲课后编程题（用 AI 实现）

题目：用 AI 辅助训练一个 DCGAN，在 MNIST 上生成手写数字（50 个 epoch，CPU 半小时内可跑完，或用 Colab；实在没有条件，可改用 `diffusers` 库调用预训练扩散管线生成数字，交付要求相同）。重点观察**模式崩塌**：随着训练推进，生成的数字种类是越来越丰富，还是越来越窄（比如只剩几个 1 和 7）？

交付物：代码文件 + 一张 5×10 生成样本网格图 + 生成器/判别器双 loss 曲线 + 50 字以上观察记录（有无模式崩塌迹象，依据是什么）。

提示词建议：

```
请帮我用 PyTorch 写一个最小可跑的 DCGAN，在 MNIST 上生成手写数字：
1. 生成器：100 维噪声经转置卷积上采样到 28x28，末层用 tanh；
2. 判别器：普通卷积下采样，末层用 sigmoid，图像先归一化到 [-1,1]；
3. 训练 50 个 epoch，Adam 学习率 0.0002，每个 batch 先更新 D 再更新 G；
4. 每 10 个 epoch 保存一张 5x10 生成样本网格图，同时记录两个 loss；
5. 结束后用 matplotlib 画双 loss 曲线。
我只有 CPU，请把网络宽度和批量大小控制在 30 分钟内能跑完的规模，
中文注释解释每一层的作用和交替更新的用意。
```

验收线：代码能跑完并出图；你能向同学讲清楚这段代码里“谁在学谁”——D在学分辨真假、G在学 D 的鉴定标准；至少一处你自己的修改（噪声维度、加深一层、换学习率）并说出看到的变化。“代码是 AI 写的没关系，讲不清楚就是没做。”

加分项：把无条件 GAN 改成条件 GAN，指定“只生成数字 7”；或对比训练 50 与 200 个 epoch 的样本多样性，写 100 字分析过训练与模式崩塌的关系。

判别模型（Discriminative Model）

学习类别边界 $p(y \mid x)$ 的模型，看图说话；本课前十一讲的主角。

生成模型（Generative Model）

学习数据分布 $p(x)$ 并能从中采样出新样本的模型，无中生有。

生成器（Generator）

GAN 中的造假者：把噪声向量映射成假样本，目标是骗过判别器。

判别器（Discriminator）

GAN 中的鉴定师：给输入打“真/假”分数，深度伪造检测器的直系祖先。

极小极大博弈（Minimax Game）

G 的最小化目标嵌在 D 的最大化目标里，双方在同一函数中较劲的数学表述。

纳什均衡（Nash Equilibrium）

任何一方单方面改变策略都占不到便宜的稳定状态；GAN 训练寻找的理想终点。

模式崩塌（Mode Collapse）

生成器只产出少数几种“安全”样本、多样性死亡的著名训练事故。

条件 GAN（cGAN）

给生成器附加条件（类别、文字）的变体，生成从抽盲盒变成点菜。

CycleGAN

无配对数据之间做风格迁移的 GAN，靠循环一致性损失实现马变斑马式“翻译”。

StyleGAN

风格分层注入、可高分辨率可控生成人脸的 GAN 系列，2018–2020 年以假乱真的代表作。

扩散模型（Diffusion Model）

前向千步加噪、反向学会逐步去噪的生成模型，2022 年起取代 GAN 成为图像生成主流。

对抗训练（Adversarial Training）

把对抗样本纳入训练让模型提前“见世面”的加固手段，本讲与数据基座项目的共同关键词。

第二部分

第二篇　项目背景篇：三个真实项目到底在做什么

第二篇导读：三个项目全景

第二篇地图

讲什么：三个真实项目的来龙去脉——六目拼接、水下双目自动标记、数据基座治理，外加全班 15 个小组的分工地图与人人要守的课程框架。**为什么值得读**：第三篇的任务书只告诉你“做什么、怎么交”；这一篇告诉你“为什么做、船开向哪”。背景清楚了，任务书上的每条要求才不是黑话。**学完你能**：用一句话说清自己项目的甲方痛点与任务，找到自己组在 15 组地图上的位置，把四个实验节点和最终提交日标进自己的日历。

先设想一个场景。开学第三周，组长把任务书拍到你面前：“组 5，双台账加 A/B 对比，回归脚本每周跑一次，第 19 周周五交。”如果你不知道台账为谁而建、A/B 在比什么、回归到底回归什么，这份任务书就是一堵墙；如果你读过对应的项目章，你会发现每一条要求背后都有一个具体的人和具体的难处——台账是给评测员看的，A/B 是在拿新模型和老算法比武，回归是在防止“改好一处、弄坏三处”。**这一篇的任务，就是把那堵墙提前拆掉**：拿到任务书之前，先知道船开向哪。

0.9 三个项目：先来一张速览表

这门课的作业不是教科书后面的习题，而是三个**正在运行的真实工程项目**。全班 15 个小组（每组 5 人）分属其下。三个项目的甲方背景、一句话任务、组数与课程支撑讲次见下表，随后每个项目再给一段人话介绍；完整的现场背景、系统架构与难点剖析，请进入第二篇各章细读。

表 4 三个真实项目速览（15 个小组，每组 5 人）

项目	甲方 / 背景	一句话任务	组数	课程支撑
六目拼接	360Stitcher 在研实时全景系统	把实时 360° 全景拼接系统里九类人工调参，升级为九个可学习的神经网络“参数头”	8	第 0–3、5–11 讲
水下双目	中海油 HYSY123 FPSO	给 ROV 锚链检查视频装上双目立体感知：教师打伪标签，训练轻量学生模型	4	第 0、5–8 讲
数据基座	海洋石油平台	治理 126 份 CSV 共 3.1 亿行数据与 1001 页手册：结构化、一句话问数、虚假内容识别与对抗训练	3	第 3、9–12 讲

项目一：六目拼接。六台工业相机围成一圈，各管约 60°，系统把六路视频实时拼成一幅 360° 全景。过去拼得好不好，全凭九类手工调的参数；这学期要把它们升级成九个可学习的神经网络“参数头”，而且要配上回退保护——模型万一失手，系统自动退回老算法，现场一分一秒不能停。详见第 一 章。

项目二：水下双目自动标记。海上十几万吨的 FPSO 靠几条巨型系泊锚链“锚”在油田上方，锚链要靠水下机器人（ROV）定期下水检查，拍回的视频浩如烟海。项目用双目相机的左右两幅画面算出深度，让机器自动认出锚链每一环；再用大模型当“教师”打伪标签，过滤后训练轻量的“学生”模型，500 对锚链立体图对的数据家底由你们亲手搭建。详见第 二 章。

项目三：数据基座治理。某海洋石油平台的机房里躺着 126 份 CSV 文件、共 3.1 亿行运行数据，外加 1001 页工程手册，靠人翻一辈子翻不完。项目干三件事：把数据与手册**结构化**；搭“一句话问数”智能体——工程师说人话，大模型加图检索负责听懂并查数；训练“监考员”模型识别虚假内容，并和造假方反复过招、共同变强。详见第 三 章。

0.10 共同的课程框架：节点、证据与底线

三个项目背景各不相同，但游戏规则完全一致，三件事人人要记。

第一，四个实验节点加一条提交线。四次现场实验与项目节点并行推进，时间如下表。注意：课程层面四次实验有统一主题，但在你自己的项目里，每个节点各有具体动作——以六目拼接为例，实验一是用大模型 agent 解读 360Stitcher 代码，实验二是在本机完成 VS2022 编译部署并看到九行 [Provider] 日志与探针 12/12 通过，实验三是真训练加交付加 A/B 对比，实验四是复训与优化。各组动作的命令级细节，见第三篇各组任务书的“四次实验安排”一节。

表 5 四个实验节点与最终提交（所有项目通用）

节点	时间	课程层面做什么
实验一	10月 12 日	大模型 agent 模式配置及应用：装好编程助手，并用它解读自己项目的代码
实验二	11 月 2 日	匹配合适程序并执行本地部署：项目程序在自己电脑上编译跑通
实验三	11 月 23 日	深度网络的模型训练实践：亲手完成所属项目的训练任务
实验四	12 月 21 日	模型优化训练实践：把模型跑得又快又稳
提交线	第 19 周周五（2027 年 1 月 22 日）	全部成果提交验收，完全提交离场

第二，证据包三件套。凡声称“我做出来了”，必须同时拿出**命令、日志、产物**三件套：你是敲哪条命令跑出来的、终端输出了什么、留下了什么文件。截图可以修图，口说无凭，只有三件套齐了，成绩才认。

第三，底线只有一句话：AI 可以用，结果自己扛。这门课不禁止 AI，反而要求你用 AI——但每个交付物都可能被随机抽问，讲不清楚就是没做；实验数字必须是自己代码在自己数据上跑出来的。具体到项目里，还有五条防糊弄纪律：**证据包制、探针防绕过、A/B 数据说话、双台账、如实条款**（不得虚构训练精度，发现一处该节点成绩作废）。它们在项目章和任务书里会反复出现，这里先混个脸熟。

0.11 15 个小组地图：找到你自己的位置

全班 15 个组的分工见下表。六目拼接的 8 个组名与任务即第三篇第 1–8 组的章名；水下四组与数据基座三组同理，各组任务的精确边界以第三篇任务书为准。

表 6　15 个小组地图（组名与一句话任务）

组号	项目	组名	一句话任务
第 1 组	六目	匹配决策组	训练并交付 policy、mc、geometry 三个头：这一对画面怎么拼、信几分、几何参数是多少
第 2 组	六目	特征质量组	训练并交付 feature_uncertainty、descriptor_softening、temporal_state 三个头
第 3 组	六目	缝合融合组	训练并交付 seam_uncertainty、gain_field、fusion 三个头
第 4 组	六目	数据与标签组	录六路视频、抽帧、对齐遥测，制作 match/geometry/policy 三段标签
第 5 组	六目	测试集与评测组	建指标台账与数据量台账，A/B 对比 `ab_compare.py`，回归 `run_regression.ps1`
第 6 组	六目	端到端数据组	建 50 对以上真实图对数据集与 benchmark
第 7 组	六目	端到端训练组	在 RTX 4080 上以 `--epochs 200` 训练 `ropstitch.onnx`，探针 5/5
第 8 组	六目	端到端优化组	引擎单次调用 ≤300 ms、总预算 800 ms；fp32/fp16/int8 三档对比，int8 劣化 <10% 才交付
第 9 组	水下	数据组	采集并构建锚链双目数据家底（含 500 对立体图对）
第 10 组	水下	部署组	把训练好的学生模型部署到现场实时可用，并给出部署证据
第 11 组	水下	训练组	教师–学生伪标签闭环训练
第 12 组	水下	评测组	立体感知指标的评测与验收
第 13 组	基座	数据结构化组	把 126 份 CSV 共 3.1 亿行数据与 1001 页手册结构化入库
第 14 组	基座	智能问答与图检索组	搭“一句话问数”智能体：大模型 + 数据库 + R-GCN 图检索
第 15 组	基座	虚假内容识别与对抗训练组	训练“监考员”模型识别虚假内容，与造假方对抗过招

看这张表时留意一件事：**这门课为每个组都准备了“不等待”的设计**。实验一至三是离线自研：水下四个组开局就领到资产包（FoundationStereo环境、500 对锚链图对、初步模型），六目八个组用公开数据集（UDIS-D、MegaDepth、HPatches 等）与系统自带仿真模式先把方案跑通；实验四（12 月 21 日）才是合并联调——真实数据接入、跨组对账、统一验收。所以：跨组沟通节点要写进周计划，但任何一组都不需要等另一组开工。

0.12　怎么读这一篇

建议的读法分三步。**第一步**，先精读自己所属项目的章节（六目拼接读第 一 章，水下双目读第 二 章，数据基座读第 三 章），目标是能向室友复述“甲方是谁、痛点是什么、我们组要交什么”；**第二步**，翻开第三篇，先读使用说明（第 三 章），再精读自己组的任务书，把提交清单和四次实验抄进日历；**第三步**，把另外两个项目的章当“课外读物”略读——知道隔壁组在干什么，联调时才不会鸡同鸭讲。

最后提醒一句：第二篇不是读完就束之高阁的小说。第三篇任务书里大量术语——参数头、伪标签、对抗训练、双台账——的完整解释都在第二篇对应章节里，读任务书卡壳时，回到第二篇查，比问 AI 更可靠，因为这里的每个字都对着你们项目的真实系统。

项目一　六目拼接：给六只眼睛一个全景世界

想象你站在一座海洋石油平台的直升机坪中央。海风、管线、火炬塔、远处的补给船——四面八方都是需要盯住的东西。可无论把一台相机架在哪里，它一次也只能看住一个方向：普通监控相机的水平视野不过 90° 上下，人眼双眼合并的视野也只有约 120°，而平台四周是完整的 360°。装四台相机各看一面？值班员就得在四块屏幕间来回切换，脑子当拼图器。**更好的办法是让机器自己把画面拼成一幅完整的全景**——这正是全景拼接（panorama stitching）技术要做的事，也是你们项目一的全部出发点。

本章地图

讲什么：六目拼接项目的前世今生——需求从哪来、全景拼接五步曲的原理直觉、360Stitcher系统的真实架构与性能数字、为什么要给九类人工参数装上神经网络“参数头”、两条技术路线怎么对照推进、八个组如何分工。**为什么值得读**：第三篇任务书里的每个词（缝合线、增益、A/B、探针、回退）都长在这一章的枝干上。**学完你能**：向任何人讲清“六路画面如何变成一幅全景”“参数头是什么、为什么要学、学坏了怎么办”，并找到自己组的上下游位置。

下图是一幅典型的全景拼接成果：镜头原地转一圈，画面被摊平成一张长卷。项目一要做的，就是让这个过程**由六台相机实时自动完成**——不是拍完回家慢慢拼，而是每秒几十遍地拼下去。

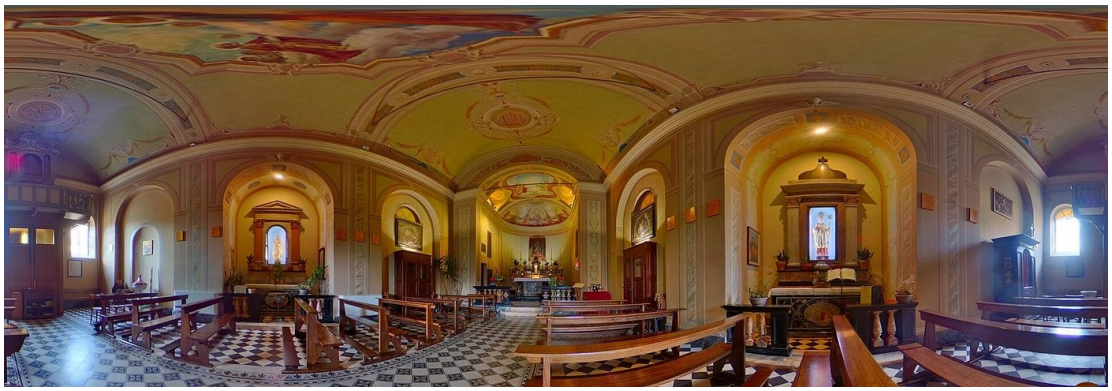


图 1.1　全景拼接成果示例：相机原地旋转一圈，视野被展开成一幅无缝长卷——这就是“全景拼接”四个字的样子

图片来源：维基共享资源（Wikimedia Commons）360° 全景示例图，仅用于课堂教学

1.1　需求从哪来：一台相机看不住一座平台

先说甲方视角的痛点。油气平台、炼化厂区、大坝、广场周界这类场所，安全监控有个共同的尴尬：**要盯的地方是一整圈，而相机只看得见一个扇面**。多装几台相机当然能补，但画面各管一段，值班员要靠人脑把它们“缝”起来——疲劳的时候，缝就漏了。有人会想到超广角或鱼眼相机，但它在后文 1.2 节会看到死穴：边缘畸变严重，且有限的像素被摊到 360° 上，远处目标小到没法辨认。下面这座海洋石油平台就是典型场景：管线密布、人员往来，四周海面与航道都值得关注，任何一块视野盲区都是隐患。



图 1.2　应用场景示例：海洋石油平台——四周海面、航道与设施构成完整的 360° 监视需求，单台相机无论如何架设都看不全

图片来源：维基共享资源（Wikimedia Commons）海洋石油平台照片，仅用于课堂教学

项目的硬件答案是**六相机环形阵列**：六只“眼睛”头朝外围成一圈，每只负责约 60°，相邻两只的画面留出一条重叠带——正是这条重叠带，让后续算法能把六幅画面严丝合缝地接起来。市面上的消费级全景相机（下图这类多镜头阵列）用的也是同一思路：与其造一只万能的超级镜头，不如让一群普通镜头协同作战。

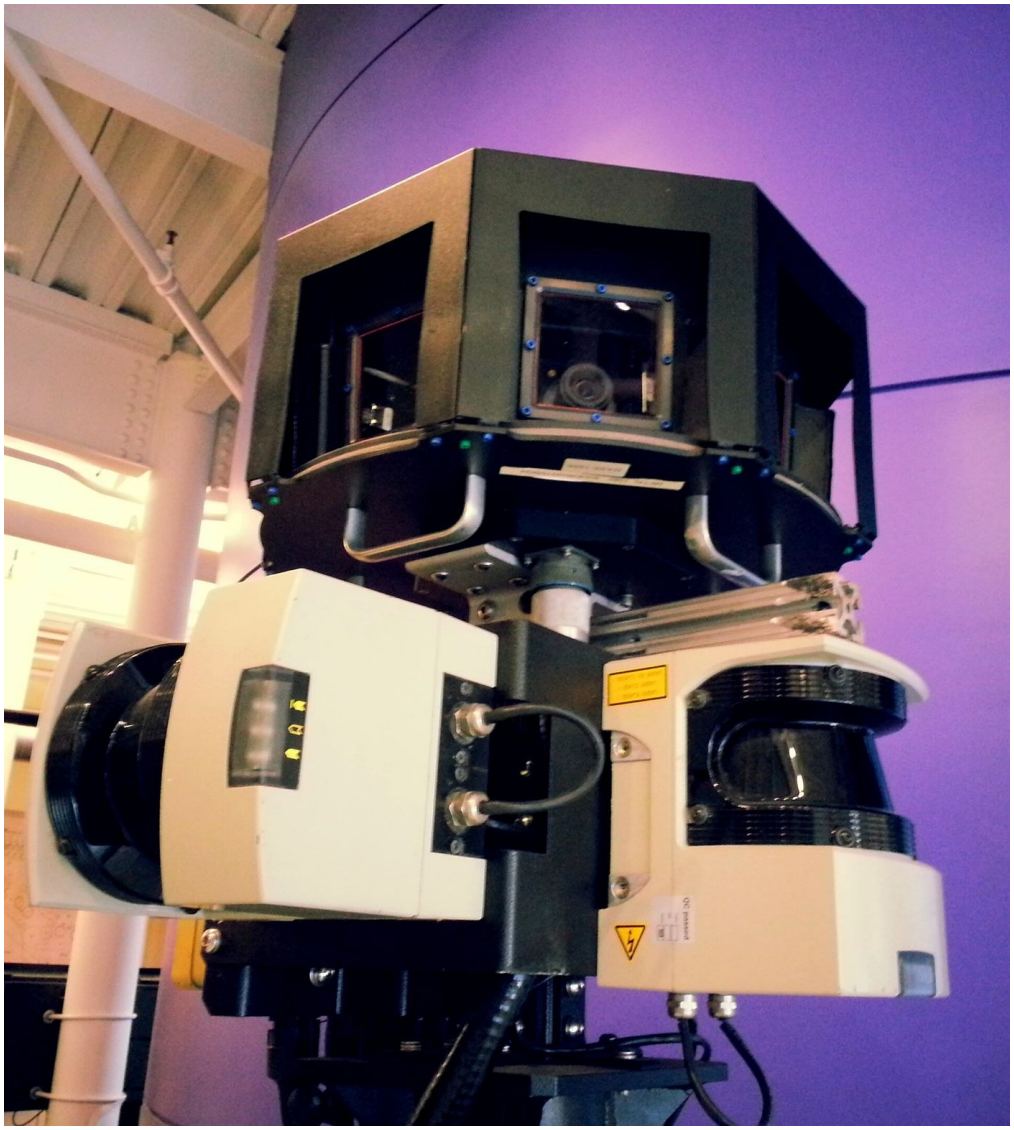


图 1.3　多相机阵列示例：若干镜头朝向不同方向、共享同一台处理单元——消费级全景相机与本项目六相机环阵是同一个思路

图片来源：维基共享资源（Wikimedia Commons）多相机全景阵列照片，仅用于课堂教学

我们的真实系统名叫 **360Stitcher**，是实验室在研的实时多相机全景拼接系统。看两张它的实际运行截图，你就能把“六目”两个字看明白：下图中，上排小窗标着“原始画面”，是六台相机各自看到的景象——一座红砖楼环绕、中央铺着草坪的园区庭院；下排那条黑底长卷，是系统把六路画面实时拼出来的全景。右下角的 FPS 读数 26.6，表示此刻系统每秒能拼出约 26 帧全景——注意是“拼出来”，不是“播放出来”，每帧都是六幅图现场算出来的结果。

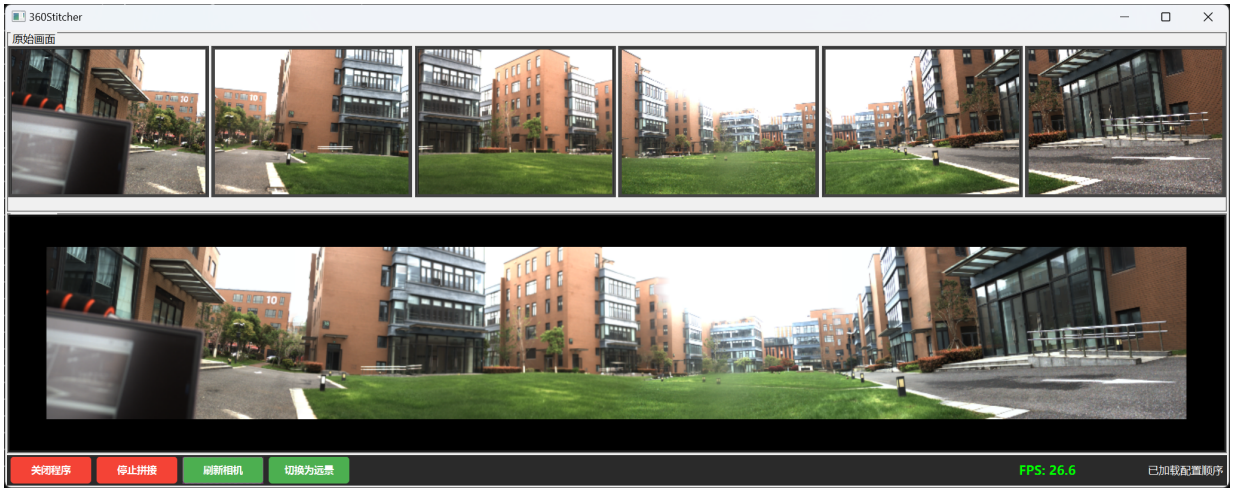


图 1.4　360Stitcher 实际运行截图（园区场景）：上排为六路相机的原始画面（红砖楼与草坪的庭院），下排为系统实时拼出的全景长卷；右下角 FPS 26.6 表示每秒约拼出 26 帧

图片来源：360Stitcher 项目组真实运行截图（六目拼接项目素材）

换一个场景它照样干活。下图的取景地是园区外侧的街道：斑马线、行道树、路边的共享单车与电动车，被六路画面分头拍下、无缝接成长卷。你仔细看长卷里那些横向连续的白线与绿化带——它们每个瞬间都在接受“六只眼睛看的是同一个世界”的检验，接歪一点都会立刻露馅。



图 1.5　360Stitcher 实际运行截图（街景场景）：斑马线、行道树与共享单车被六路画面接成一条连续长卷，横向的斑马线与绿化带是检验“接得正不正”的天然标尺；右下角 FPS 24.7

图片来源：360Stitcher 项目组真实运行截图（六目拼接项目素材）

最后回答“为什么不用一个超广角”。用数字算账最清楚：本项目相机单张原始图为 4608×3456 像素（约 1600 万像素）。按水平视野约 90° 估算，六目方案每度视野能摊到约 51 个像素；而假如用一只 360° 鱼眼把同样 4608 个像素摊满一整圈，每度只剩约 13 个像素——差了 4 倍，远处的人脸直接糊成马赛克。再看畸变：视野越广，镜头把直线掰弯得越厉害，画面边缘的人与物会被拉伸变形。六相机环阵用“普通镜头 + 算法拼接”换来了**高分辨率、低畸变的 360°**，代价是把难题从 optics（光学）转移给了 algorithm（算法）——也就是转交给了你们。

零基础扫盲：人眼 120°，全景 360°，长卷是怎么“卷”出来的

人眼双眼视野约 120°，是因为两只眼球朝前、各管约 60° 再拼起来——你会发现，**人眼本身就是一台“双目拼接”的机器**，本项目不过是让硅基眼睛重走了一遍进化路。那全景长卷为什么能摊平？设想把一圈相机看到的视野画在一只易拉罐的侧面，再把罐身剪开摊平——这就是**圆柱投影**（cylindrical projection）：把 360° 的环形视野展开成一张矩形长图。它的副作用你也见过：全景照片里，本来笔直的楼边在画面两端会略显弯——那是罐面被强行摊平留下的“折痕”。本章截图里的黑色长卷，正是这种圆柱投影长卷。

1.2　全景拼接五步曲：从六张歪图到一幅长卷

把六路画面变成一幅全景，系统每帧都要走完五步。先用下图建立全局印象，然后逐步拆解——这五步也是本章的骨架，后面 1.4 节的九个参数头，一个不落地长在这五步上。

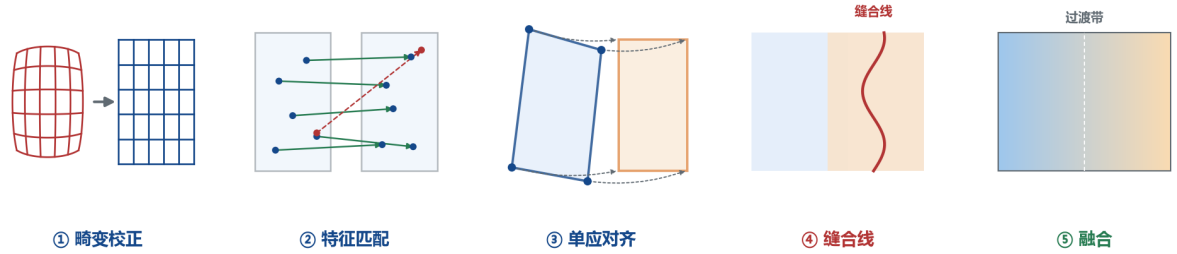


图 1.6　全景拼接五步曲总览：畸变校正把“哈哈镜”掰直，特征匹配找到两幅画面的“同名点”，单应对齐把画面搬正，动态规划在重叠区挑出最不显眼的缝合线，融合把接缝羽化到看不见

1.2.1　第一步：畸变校正——先把“哈哈镜”掰直

理想相机是针孔模型：光线走直线，方块拍出来还是方块。现实中的镜头是玻璃磨出来的，光线在边缘弯得更狠，于是画面里的直线会向内弯——整幅图像得像照过哈哈镜，这叫**桶形畸变**（barrel distortion）。畸变不除，后面算“两幅图差多少”时全都会失准：你以为两幅画面对不齐，其实是镜头在撒谎。

对策是**畸变校正**（distortion correction）：离线给每台相机做一次**标定**（calibration）——拍十几张棋盘格照片，解出这台镜头的“度数”（焦距、光心等内参和畸变系数），此后运行时每帧图像先按这组系数反解、掰直，再进入后续流程。你已经在 1.3 节的截图里提前见过它的道具：室内场景那块贴在白板上的黑白棋盘格，就是标定用的靶子。

1.2.2　第二步：特征提取与匹配——在两幅画面里找到“同一个点”

校正之后，算法要回答一个关键问题：左眼画面里的这栋楼角，对应右眼画面里的哪个像素？逐像素两两比对？两幅 1440×1080 的图有十几亿种配对，算到下个世纪。工程的办法是只挑**特征点**（feature point / keypoint）说话：选那些“挪一点就大不一样”的位置——角点（corner）。想象拼拼图时你会先对天空？不，你会先拼窗户角、旗杆尖，因为那里信息量最大。本项目用的 **ORB** 特征就是这么一对组合拳：用 **FAST** 算法飞快地找出角点，再用 **BRIEF** 给每个角点生成一段 256 位的二进制“指纹”，合称 ORB（Oriented FAST and Rotated BRIEF）。

匹配（matching）就是拿指纹互相打分：两段指纹越像，越可能是“同一个点”。打分的思想你其实学过——第 0 讲的点积就是最早的“合拍程度打分器”：对应位置相乘再相加，分高即相似。二

进制指纹的比对用“数不同位的个数”（汉明距离），思路同源：**都是把“像不像”变成一次可计算的算术**。每幅图取多少个特征点？本项目目前定为每图 1500 点——这是个写死的人工经验值，记住它，1.4 节它会成为被改造的对象之一。

1.2.3　第三步：几何对齐——一张 3×3 表格把画面“扶正”

找到了几十上百对“同名点”，下一步是求出两幅画面之间的**坐标换算表**：右图的点该挪到左图的哪里？这张换算表就是大名鼎鼎的**单应矩阵**（homography）。它的直觉见下图：同一块平面（一面墙、一片草坪）在两幅画面里各留了投影，单应矩阵能把一块平面上的四角精确搬到另一块平面的四角上——相当于把一张拍歪的照片“扶正”再“平移”到位。

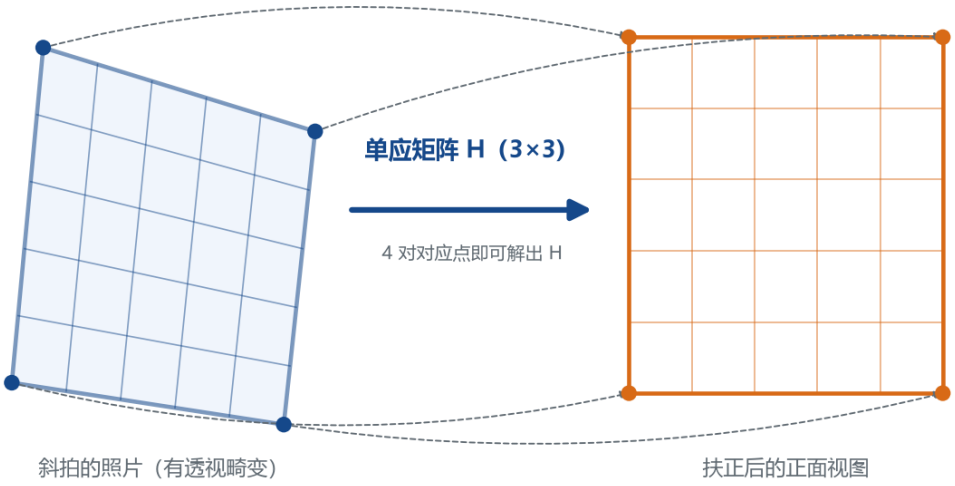


图 1.7　单应变换示意：同一平面在两幅画面中的投影之间存在一张 3×3 的“换算表”，能把平面 A 的四角映射到平面 B 的四角——斜拍的照片由此被“扶正”

知识补齐：3×3 矩阵、8 个自由度与 RANSAC 一句话

单应矩阵是 9 个数排成的 3×3 方阵，却只有 8 个自由度：因为整体缩放不改变映射效果（所有数翻倍，同一个点算出的位置不变），习惯上约定最后一个数为 1，于是真正待定的数只有 8 个。每对同名点能提供 2 个方程，所以**最少 4 对点就能解出它**——这就是为什么各种拼接软件都要求相邻照片留出重叠、并在重叠处找到至少几个可靠同名点。推导不必急，先记住直觉：**单应 = 两块平面之间的一张坐标换算表**。**RANSAC 一句话：**4 对点就能解一个单应，可万一其中一对是误匹配的“卧底”呢？**随机抽样一致**（RANSAC, Random Sample Consensus）的对策是赌概率：随机抽 4 对算一个解，拿它到全体点里验票，看多少对点“同意”这个换算表；换几批重抽，最后票数最多的解胜出——少数卧底翻不了盘。

360Stitcher 在这一步还有自己的工程取舍：六台相机刚性固定成一圈，相邻相机之间**只有水平方向的相对偏移，没有旋转和俯仰**，于是系统采用比完整单应更简单的**纯平移几何模型**——参数更少，解得更快，抗噪声也更稳。此外，为了让相邻帧的跟踪不每帧从零算起，系统用**相位相关**（phase correlation）在频率域里直接测出两帧间的平移量，作为帧间跟踪的“惯性基准”，ORB 匹配只在这个小范围里精修。

1.2.4　第四步：缝合线——让“刀口”藏在最不显眼的地方

对齐之后，重叠带里两幅画面“说的是同一件事”，但取哪边？最笨的办法是沿中线一刀切——可刀口要是穿过一辆正在移动的自行车或一盏忽明忽暗的灯，接缝处立刻穿帮。**缝合线**（seam）的使命，是在重叠带里找出一条**差异最小**的路径当刀口：让它沿着草地、墙面这些两幅画面最一致的地方走，绕开行人、车灯与反光。

找这条最优路径用的正是**动态规划**（Dynamic Programming, DP）：把重叠带想象成一张网格，每个格子的“代价”是两幅画面在该处的差异，缝合线是从上端走到下端的一条路径，DP 保证逐列累积代价最小时全局最优——像在山谷里找一条翻越起伏最小的山脊线。这一步的输出是一条逐像素的路径，告诉融合步骤“线的左边信左眼，右边信右眼”。

1.2.5　第五步：融合——把接缝“羽化”到看不见

即便对齐和走线都完美，两台相机的曝光、增益也总有细微差别，硬接仍会留下一道可觉察的色阶。**融合**（blending）负责最后一层柔化，本项目备着两种手段。**feather 融合**（羽化融合）：在缝合线两侧各划一个过渡带，两幅画面按“离线有多远”加权混合——越靠近左眼的地盘左眼权重越高，反之亦然，颜色渐变，肉眼无痕。**多频段融合**（multi-band blending）：把画面按空间频率分解成若干层（可以想成“大色块层”到“细纹理层”），低频层用很宽的过渡慢慢混，高频层用很窄的过渡快速换——大尺度亮度平稳过渡、小尺度细节不失真，是全景软件的看家绝活。而在融合之前，通常还有一步**增益补偿**（gain compensation）：按各相机重叠区的亮度统计，给每路画面乘一个系数，先把整体明暗拉齐，再让羽化收尾。

动手时间：用手机预演一遍五步曲

不用写一行代码，手机就能把本章五步全走一遍。**步骤**：(1)打开相机 App 的**全景模式**，原地缓摇，拍一张 360° 或 180° 全景——这就是“实时拼接”的单帧版；(2) 再切普通模式，对着同一场景拍两张水平错开、重叠约三分之一宽度的照片；(3) 放大重叠区，找出至少 4 对“同一个点”（窗角、标志牌、砖缝交点都行），在纸上记下它们在两张照片里的位置——你刚刚手工完成了“特征匹配”;(4) 数一数：4 对点、每对 2 个方程、单应矩阵 8 个自由度，恰好够解——这是第三步的最小配置；(5) 回到那张全景照片，找出缝合线走过的位置和一处你觉得“接得不太完美”的地方。**验收**：能向同学指出你那张全景照里“第五步融合”最好与最差的地方各一处，并说清为什么。**加分项**：故意在拍摄两张照片时把其中一张靠近镜头的手指拍进重叠区，看看全景模式会不会把它切掉或留下“半个你”——想想这是第四步缝合线在保护谁。

1.3　360Stitcher 长什么样：六个模块与一串硬数字

五步曲是“怎么拼”，现在看“怎么跑”。360Stitcher的设计哲学是**离线建模 + 在线执行**：标定、内参、相机次序这类一锤子买卖全部离线做好、存成配置；运行时只做必须每帧做的事。整条数据通路是一条流水线，六个工位见下表（职责按模块名与项目组公开结构整理）。

表 1.1　360Stitcher 的模块链：六路画面从相机到长卷的旅程

顺序	模块	职责（一句话）
1	MainWindow	Qt6界面门面：预览、按钮、编辑模式，人只跟它说话
2	HikCamera	对接海康相机 SDK，把六路大图从相机取进来
3	CameraWorker	每路相机一个工作线程，专职取流、不掉帧
4	FrameSubmitter	把六路几乎同一时刻的帧配成一组，排队提交
5	AsyncStitcher	异步调度：收到一组帧就交给拼接核心，绝不堵塞取流与界面
6	FastStitcher	拼接核心：ORB 匹配 + 纯平移模型 + 相位相关跟踪 + DP 缝合线 + 多频段/feather 融合

相机本体是海康工业相机（下图）。它与家用摄像头的区别不在“清楚”而在“守时”：工业相机的曝光与出帧由时钟或外部信号统一触发，六路画面的时间戳对得齐齐整整——要是六只眼睛各看各的时刻，运动物体在接缝处会凭空瞬移，拼什么都白拼。



图 1.8　工业相机示例（海康威视等品牌）：与家用摄像头相比，工业相机的价值在于分辨率、帧率稳定与同步触发——六路画面“同一时刻”才拼得起来

图片来源：海康威视（Hikvision）机器视觉官方产品图，仅用于课堂教学

性能上，系统挂着一串值得背下来的硬数字，见下表。注意“帧周期约 33 ms”与“FPS 天花板约 16”的关系：相机每 33 ms 交来一组新帧（约 30 fps），而 8 图静态基准测出的拼接吞吐天花板只有约 16 FPS——来料速度接近出料速度的两倍。**流水线两头速度不匹配，是所有实时系统的日常**，也是 1.4 节“参数头必须又小又快”的物理根源。

表 1.2　360Stitcher 性能数字速查

指标	数值	说明
相机原始分辨率	4608×3456（约 16 MP）	六路海康相机每路的大图
拼接工作分辨率	1440×1080	缩小后参与拼接运算，拿分辨率换实时性
相机帧周期	约 33 ms	相机每约 33 ms 交来一组新帧（约 30 fps）
8 图静态拼接吞吐	天花板约 16 FPS	基准测试下拼接核心的吞吐上限
界面实测帧率	约 18–27 FPS	本章各截图右下角读数的波动范围
ORB 特征点数	1500 点/图	人工经验值，1.4 节起待参数头接管

系统跑起来什么样？看下面这张遥测（telemetry）截图：终端里每帧滚出一行 `[Latency]` 日志，四个字段分别是 total（单帧总延迟）、stitch（拼接耗时）、frameAge（帧龄：这帧从相机出发到现在过了多久）、uiLatency（界面延迟）。截图里 total 约 32–63 ms，其中拼接核心稳定在 29–45 ms——系统正贴着 33 ms 的帧周期“低空飞行”。这套逐帧自报家门的数据流，就是你们第三篇任务书里“遥测”两个字的本体：**参数头好不好，最终都由这行日志和它背后的画面质量说了算**。

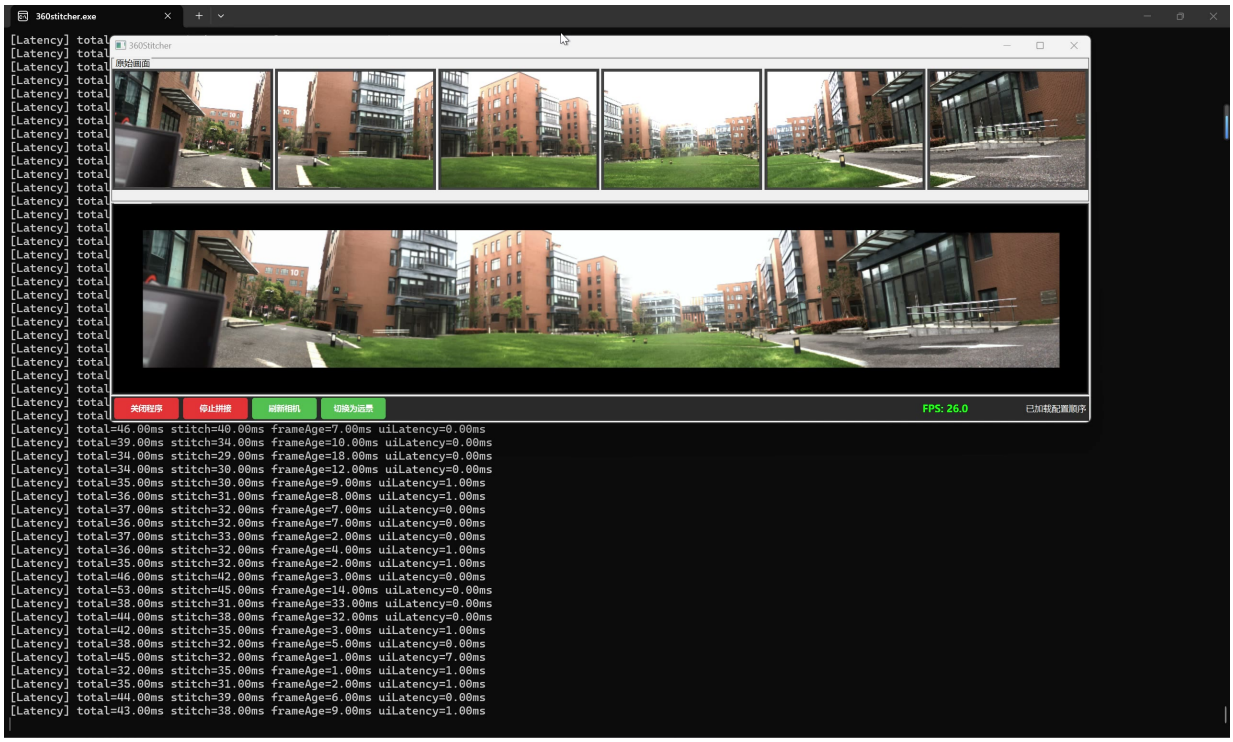


图 1.9　系统运行遥测实录：终端每帧打印一行 [Latency] 日志（total 总延迟 / stitch 拼接耗时 / frameAge 帧龄 / uiLatency 界面延迟），前景 360Stitcher 正以 FPS 26.0 运行

图片来源：360Stitcher 项目组真实运行截图（遥测终端实录）

界面本身也有讲究。下图是室内场景：六路画面拍的是同一间办公室，左侧白板上那块黑白棋盘格正是第一步标定的道具。更值得注意的是底部工具栏——编辑模式提供“调整顺序、上移、下移、保存顺序”等按钮：六台相机谁挨着谁、长卷从哪路开始，都可以在界面里调好后存成配置。这再次呼应“离线建模 + 在线执行”：**凡是能提前想清楚的，都不占用每帧 33 ms 的宝贵预算**。

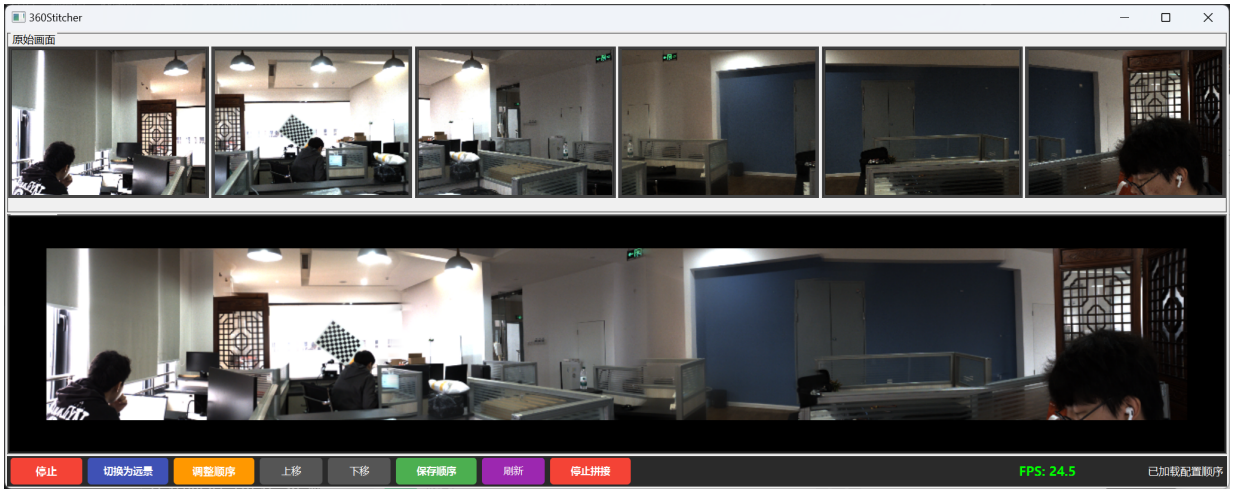


图 1.10　360Stitcher 室内场景与编辑模式：六路画面拍摄同一间办公室，白板上放置棋盘格标定板；底部“调整顺序/上移/下移/保存顺序”等按钮用于配置六路相机在全景中的排列次序

图片来源：360Stitcher 项目组真实运行截图（六目拼接项目素材）

最后一条，也是最容易被忽视的一条需求：**长期稳定运行**。监控类系统一开就是几周几个月，任何一次崩溃都是安全事故的种子。这个约束现在看似平淡，等你在 1.4 节见到“回退保护”时就会明白它有多深远——它甚至决定了整个项目敢不敢让神经网络上场。

1.4　为什么要“可学习”的参数：九个参数头

现在的 360Stitcher 是一位“经验丰富的老师傅”：ORB 取 1500 点、匹配的比例阈值、RANSAC 的距离阈值、融合的过渡带宽……九类关键参数全是人工经验值，写死在代码里或靠工程师现场手调。老师傅在熟悉的场子（本章截图那种光照均匀的园区）手艺精湛；可场景一换——傍晚逆光、雨后反光、相机被碰歪半度——质量立刻退化。下面这张截图就是一个“吃力现场”：从高楼俯瞰、天色将暗，画面大面积逆光且天空占比极高，长卷上部天空的接缝处已经泛起淡淡的拼接痕迹，FPS 也从二十几掉到 18.8。**这不是 bug，是固定参数的哲学极限**：为场景 A 调的参数，天生不可能同时是场景 B 的最优。

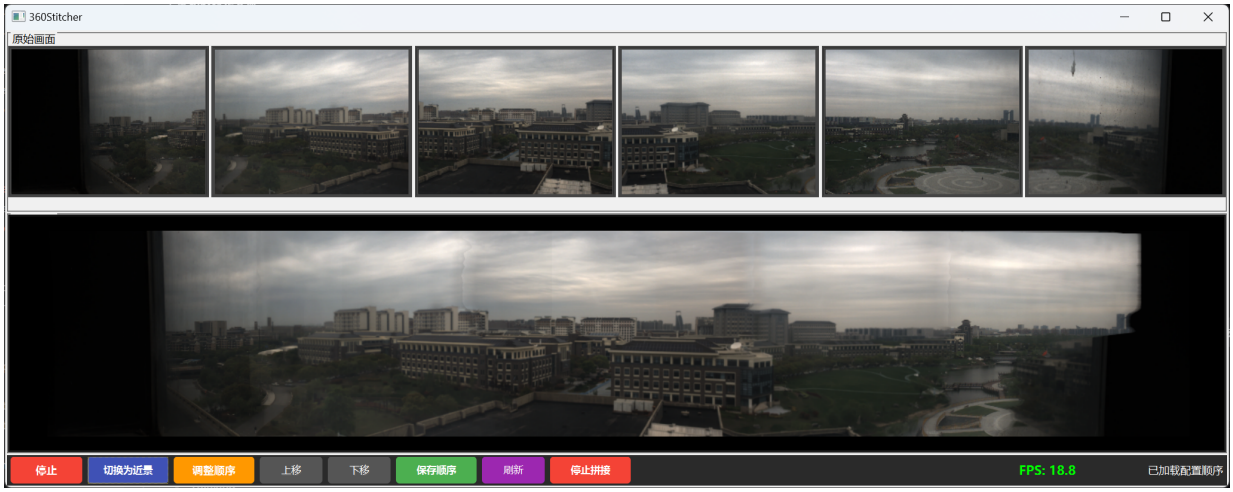


图 1.11　一个“吃力现场”（傍晚俯瞰）：逆光、低照度加大面积天空，长卷上部天空接缝处出现淡淡的拼接痕迹，FPS 掉到 18.8——固定参数换场景后开始露怯，这正是本项目要解决的问题

图片来源：360Stitcher 项目组真实运行截图（六目拼接项目素材）

工程上不是没试过改良。常见思路是**统计自适应**：让参数跟着统计量自动走，比如按重叠区亮度差自动调节融合强度、按匹配点对的分布自动收紧阈值。这类办法能救一部分场，但天花板明显：**规则终究是人写的**——你只能覆盖想得到的情况，而现场的场景组合是写不完的。光照、纹理、运动物体、相机微动，任何两两组合都可能让某条手写规则失灵，维护它们比重调参数还累。

项目的答案，是把九类参数从“写死的数”升级为**九个可学习的“参数头”**（parameter head）。参数头是挂在网络末端的一个小型神经网络分支：吃进系统现成的统计特征或特征图，吐出就该由“老师傅”凭经验给的那几个参数。九个头的名字、输入输出形状与职责见下表（职责按头的命名与形状整理，逐字定义以第三篇任务书与代码为准）。方括号是**张量形状**记号：B 表示一次一起送进网络的一批样本数（batch，批次），T 表示时间步数；[B,8] 就是“一批若干行、每行 8 个数”。

表 1.3　九个参数头一览：输入 → 输出与一句话职责

参数头	形状（输入 → 输出）	一句话职责
policy 匹配决策	[B,8] → [B,3]	看 8 项匹配统计量，三选一决定这一对画面走哪条拼接策略
mc 匹配质量	[B,10] → [B,1]	给本轮特征匹配打个置信分：分低就别硬拼
geometry 几何参数	[B,16] → [B,5]	从 16 项统计特征回归 5 个几何参数，接管手调的几何阈值
feature_uncertainty 特征不确定性	[B,1,32,32] → [B,3]	扫一遍特征图，报出 3 个“这帧特征有多不可靠”的指标
descriptor_softening 描述子软化	[B,1,32,32] → [B,256]	为 256 维描述子的比对生成软权重，让“生硬的位比对”学会看场合
temporal_state 时序状态	[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
seam_uncertainty 缝合不确定性	[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”
gain_field 增益补偿	[B,12] → [B,6]	从 12 个亮度统计量算 6 个增益系数，接管手动增益补偿
fusion 融合参数	[B,12] → [B,3]	从 12 项统计量算 3 个融合权重，决定多频段/feather 怎么混

九个头的组织方式见下图：从拼接系统引出九条分支，三个颜色分组恰好对应三个小组（组 1 管匹配与几何三头，组 2 管特征与时序三头，组 3 管缝合与融合三头）；每个头**独立训练、独立交付、逐头验收**，交付格式统一为 **ONNX**（Open Neural Network Exchange，开放神经网络交换格式——一种跨框架通用的模型文件格式，训练用 PyTorch，部署用 C++ 运行时加载同一个文件）。训练好一个头，就在系统里替换掉一类人工参数；没训练好，老参数原样顶岗，互不拖累。

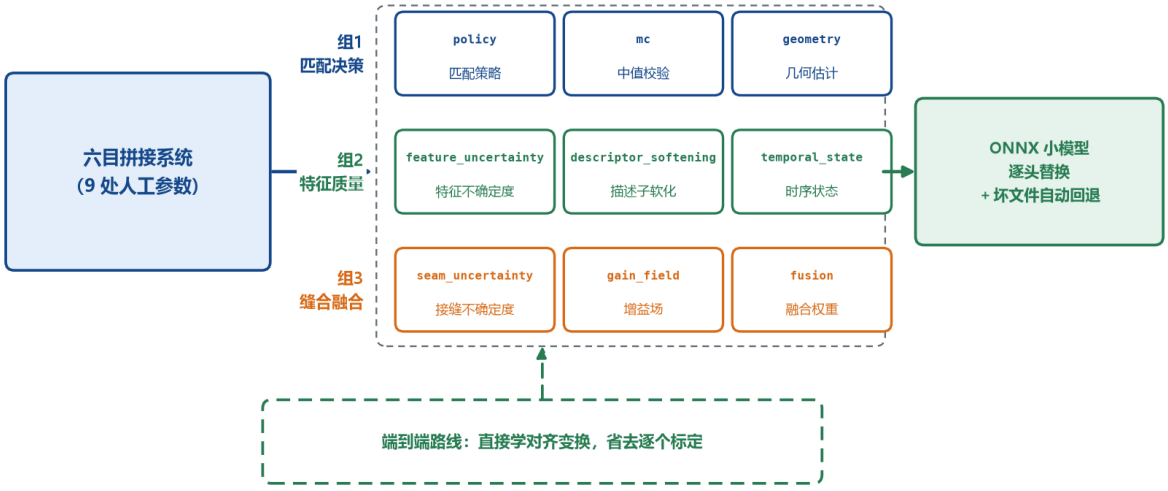


图 1.12　九头学习路线总览：拼接系统把统计特征交给 9 个参数头（三组颜色对应三个组），每个头独立训练、转成 ONNX 交付，身后站着回退保护

而让神经网络敢上岗的最后一道闸，是**回退保护**（fallback）。每个参数头加载时走 `[ProviderFallback]` 机制：模型文件损坏、推理出错、输出越界，系统不崩溃、不报异常退出，而是**自动退回原来的人工参数**，日志里记一笔，继续拼接。想通这件事你会拍案叫绝：参数头的产出只是“参数”，而参数错了的后果不过是“这一帧拼得没原来好”——**交付质量不合格，只会没用，不会坏事**。这条哲学给整个项目松了绑：大胆训练、大胆交付、快速迭代，最坏结局也只是回到起点，而每一次成功替换都实打实减一分人工调参的负担。

项目连线：九个参数头，对应第一篇的哪些讲

九个头不是天外来物，每个都能在第一篇找到血缘：policy、mc、geometry 这些“吃几个统计量、吐几个数”的小头，骨子里是第 1 讲的感知器加第 2 讲的反向传播——小型全连接网络；feature_uncertainty 与 seam_uncertainty 吃 32×32、128×256 的特征图，非第 5 讲的卷积网络莫属；temporal_state 读最近 T 帧，是第 9–10 讲时序模型的领地；交付阶段的 ONNX 与 int8 压缩，用的正是第 8 讲轻量化与量化的手艺；而“逐头训练、逐头验收”的方法论，全盘来自第 3 讲的训练与评估。第三篇任务书下达每个头的任务时，都会回指这些讲次——**第一篇学得不虚，第二篇才接得住**。

1.5　双路线对照：逐个学头，还是整网端到端

站在 2026 年的技术路口，改造一个拼接系统其实有两条路，本项目**两条同时修**。**路线一**就是上文的主角：九个参数头逐个学习——保留传统流水线的骨架，只把九处“老师傅经验”换成小网络。它的优点是渐进、可控、可回退：每个头形状明确、单独验收，坏了退回老参数；缺点也明摆着：九个头各自为政，头与头之间的联动（比如匹配质量影响缝合线选择）没人统筹。**路线二**则釜底抽薪：训练一个**端到端**（end-to-end）网络——代号 RopStitch——直接吃进相邻相机的画面，一次性输出对齐变换，跳过“手工五步曲”里最难的几何估计环节。路线二在实验室的 RTX 4080 显卡上训练，标准命令 `--epochs 200`（把全部训练数据完整过 200 遍），产物是 `ropstitch.onnx`，用探针验收要达到 5/5 全过。

为什么要花双倍力气走两条路？两个理由，一实一虚。**实的那条是工程保险**：单一技术路线一旦中途受挫（数据不够、精度不达、速度超标），项目就没有退路；两条路线共享组 6 的数据、互相校验，一条受阻，另一条照常交付。**虚的那条其实是科学方法**：只有 A/B 对照才能回答“学习式方法到底比人工参数好多少”这个严肃问题——同一段六路视频，老算法拼一遍、九头版拼一遍、端到端版拼一遍，三个成绩摆上桌，结论才有分量。这批对照结果还有更远的去处：项目组正把成果整理成论文投稿 *Image and Vision Computing* 并支撑基金申报——你在组 5 评测时跑出的那张 A/B 对比表，可能就是论文里的一行关键数据。

1.6　八个组怎么分工：一条流水线上的八个工位

先看清队伍全貌。项目由导师统筹，研究生袁博总体主责（九头方案与端到端路线的技术负责人），8个本科小组每组 5 人，另有 2 位毕业设计学生深度参与。八个组的名字、任务与数据来源见下表——把它和 1.2 节的五步曲、1.4 节的九个头对着看，你会发现**分工图就是系统图的镜像**：组 1–3 各领三个参数头，组 4 供给训练口粮，组 5 手握标尺，组 6–8 在另一条路线上复刻“数据、训练、优化”三段论。

表 1.4　八个小组分工与数据来源（公开数据先行，实验四接真机）

组号	组名	一句话任务	数据来源
组 1	匹配决策组	训练并交付 policy/mc/geometry 三个头	组 4 公开仿真标签（COCO/MegaDepth → simulation）开局即训；真机标签实验四接入
组 2	特征质量组	训练并交付 feature_uncertainty/descriptor_softening/temporal_state 三个头	仿真时序标签 + HPatches/MegaDepth patch 开局即训；真 patch 第 11 周迁移转真
组 3	缝合融合组	训练并交付 seam_uncertainty/gain_field/fusion 三个头	仿真标签开局即训；fusion 彩排先行，正式版第 13 周放行
组 4	数据与标签组	公开图对喂 simulation 管线产仿真标签（开局）；录六路视频制真机标签（实验三）	组 1–3 第一周即吃你的仿真标签；真机版实验四接入
组 5	测试集与评测组	双台账（指标台账 + 数据量台账）、A/B 对比 `ab_compare.py`、回归脚本 `run_regression.ps1`	上游组 1–4；产出全项目成绩单
组 6	端到端数据组	建 50 对以上数据集（UDIS-D 公开对→真机对）与 benchmark（基准测试集）	组 7 实验三开训公开版；组 8 同步开工
组 7	端到端训练组	RTX 4080 上 `--epochs 200` 训练 `ropstitch.onnx`，探针 5/5（实验三出公开版 v1）	组 6 UDIS-D 数据；真机数据实验四续训 v2
组 8	端到端优化组	引擎单次调用 ≤300 ms、总预算 800 ms；fp32/fp16/int8 三档对比，int8 劣化 <10% 才交付	复现公开版 v1 即可开工；真机版 v2 联调日补做

八张工位图里藏着三条硬规矩。其一，**组 5 的双台账是全项目的账本**：每个头训练用了多少数据（数据量台账）、验收指标是多少（指标台账），笔笔入库，杜绝“差不多就行”。其二，**组 8 的预算是物理红线**：参数头再聪明，若把整帧拖过预算，实时性破产即全盘皆输——fp32/fp16/int8 三档精度逐一测速测精度，int8（8 位整数量化，第 8 讲的老朋友）劣化超过 10% 就不许交付。其三，下

面这个红盒子里的事，比以上所有都重要。

防糊弄五条：本项目的成绩护城河

(1) 证据包制：任何“我做出来了”必须附命令、日志、产物三件套，缺一不算。**(2) 探针防绕过**：系统内置探针脚本（如部署验收的 `nine_head_probe.py` 要求 12/12 通过、端到端探针 5/5），探针不过，人工演示再漂亮也不算数——它专防“跳过步骤装样子”。**(3) A/B 数据说话**：新方法好不好，由 `ab_compare.py` 在同一批数据上的对比结果说了算，不由 PPT 说了算。**(4) 双台账**：指标台账与数据量台账笔笔可查，训练用的每一张图都有出处。**(5) 如实条款**：不得虚构训练精度与实验结果，**发现一处，该节点成绩作废**——工程世界里，造假不是扣分，是出局。

前沿速递：拼接领域的“学习式改造”进行到哪了（2024–2026）

全景拼接正在被深度学习逐个工序地“换零件”。特征环节，SuperPoint 用自监督学习学出比手工 ORB 更稳的特征点；匹配环节，SuperGlue（2020）用图神经网络给特征点“拉郎配”，其 2024 年的蒸馏版 GIM 把这种匹配能力推广到了新场景；LoFTR（2021 起持续演进）更进一步，干脆放弃“先找点再匹配”，在特征图上直接做全局注意力匹配。对齐与融合环节，UDIS++（2023）等端到端拼接网络把“估计变换 + 网格重排 + 融合”一并学掉。2024–2026 年的趋势是拼接与基础模型、扩散模型进一步合流，视频流式拼接与开放场景鲁棒性成为热点。但要紧的判断是：**工业现场落地的主流仍是“传统流水线 + 学习式参数头”的混合路线**——纯端到端在可控性、实时性与可解释性上尚未过关。你们的 360Stitcher，恰好一屁股坐在当前工业界的主流路线上。

项目连线：四个实验节点，在六目拼接里各干什么

把 1.4–1.6 节的地图对到日历上：**实验一**（10 月 12 日）用大模型 agent 解读 360Stitcher 代码，画出模块链与数据流；**实验二**（11 月 2 日）本机部署——VS2022 编译通过、日志里出现九行 [Provider] 加载记录、`nine_head_probe.py` 探针 12/12 全绿；**实验三**（11 月 23 日）真训练、真交付、真 A/B——各头 ONNX 上系统，与老参数同场比武；**实验四**（12 月 21 日）复训与优化——精度、速度双复盘，int8 三档对比收官。之后是第 19 周周五（2027 年 1 月 22 日）完全提交离场。每个节点的命令级步骤、验收证据与常见坑，见第三篇第 1–8 组任务书的“四次实验安排”一节。

1.7　你将用到第一篇的哪些讲

第二篇读到这里，最后一件事是把项目与课堂焊在一起。下表给出 12 讲内容在六目拼接里的落点——别把它当复习提纲背诵，它是你以后“学了一讲就回来对号入座”的索引。

表 1.5　第一篇 12 讲在六目拼接项目中的落点

讲次	这讲给你什么	用在项目一的哪里
第 0 讲	点积：合拍程度打分器	描述子匹配与 policy/mc 头打分的直觉原点
第 1 讲	感知器与学习规则	一切参数头的祖宗：加权求和出参数
第 2 讲	反向传播	九个头逐头训练的引擎
第 3 讲	训练、验证与评估	逐头验收、A/B 对比、防过拟合的方法论
第 4 讲	自编码器：压缩与重建	“特征图进、小向量出”的头设计思路
第 5 讲	卷积与特征图	读懂 feature_uncertainty 等头的输入
第 6 讲	数据增强与训练技巧	组 4 数据制作与各头训练的调参手艺
第 7 讲	残差网络与演化	路线二端到端网络的骨架
第 8 讲	轻量化与量化	组 8 的 fp32/fp16/int8 三档与 ONNX 部署
第 9 讲	循环神经网络	temporal_state 头的时序建模
第 10 讲	LSTM/GRU 门控记忆	时序状态“该记的记、该忘的忘”
第 11 讲	注意力机制	学习式匹配（SuperGlue/LoFTR 一脉）的钥匙
第 12 讲	生成对抗思想	融合质量评估可借鉴“判别器”的挑剔眼光

对照之下你会发现一个规律：**课程进度与项目进度是咬合的**。10月中你在课堂上学反向传播，11月底项目里正好要交第一批训练好的参数头；12 月讲轻量化与量化，组 8 的 int8 优化恰好进入收官。这不是巧合，而是这门课的设计初衷：**学的每一样东西，几周内就在项目里用一次**。所以当第一篇某讲让你觉得抽象时，翻回本章对应行，看看它在真实系统里的模样——抽象概念一旦有了月薪 33 ms 的工作岗位，就好懂多了。

畸变校正（distortion correction）

用标定得到的镜头参数，把镜头造成的“直线变弯”反向掰直的预处理；本项目离线标定、在线逐帧执行。

ORB

特征提取组合拳：FAST 找角点 + BRIEF 生成 256 位二进制指纹，快而够用，是 360Stitcher 的特征引擎。

描述子（descriptor）

特征点的“指纹”：一串能代表该点外观的数值，匹配即指纹比对打分。

单应矩阵（homography）

两幅画面平面之间的 3×3 坐标换算表，8 个自由度，最少 4 对同名点可解。

RANSAC（Random Sample Consensus，随机抽样一致）

随机抽最小点集算解、全场投票、票多者胜的稳健估计法，专治误匹配。

缝合线（seam）

重叠区里“左图说了算”与“右图说了算”的分界路径，由动态规划找差异最小的一条。

多频段融合（multi-band blending）

把图像按空间频率分层、低频宽过渡高频窄过渡的融合法，消除接缝色阶的主力。

feather 融合（羽化融合）

在缝合线两侧按距离加权混色的简单融合法，多频段的轻量替身。

增益补偿（gain compensation）

按重叠区亮度统计给各路画面乘系数，先把整体明暗拉齐再融合。

参数头（parameter head）

挂在网络末端的小型分支：吃统计特征、吐系统参数；本项目九个头逐个替换九类人工参数。

ONNX（Open Neural Network Exchange）

跨框架通用的模型文件格式：训练用 PyTorch，交付用 ONNX，C++ 运行时直接加载。

回退（fallback）

参数头加载失败或输出异常时自动退回人工参数的保命机制（[ProviderFallback]），保证“只会没用、不会坏事”。

遥测（telemetry）

系统逐帧自报的家门：[Latency] 日志里的 total/stitch/frameAge/uiLatency，评测与调试的第一手证据。

A/B 对比

新旧方法在同一批数据上的对照评测（工具 `ab_compare.py`），项目里“谁更好”的唯一裁判。

项目二　水下双目视觉自动标记：让 AI 看懂海底世界

本章地图

讲什么：项目二的“总剧本”——从一段真实的 ROV 锚链检查视频出发：场景与需求；水下成像的四道坎；双目立体视觉原理（视差、基线、校正）；没有深度真值时的自动打标签流水线（本章高潮）；我们亲手构建的 500 对数据集；四个小组的分工与红线。**为什么值得学**：这是三个项目里“最像科幻”的一个——输入一对水下相机拍到的左右画面，输出每个像素的深度，并知道每个像素的结果有多可信。它把第一篇的卷积、注意力、蒸馏、部署全部串成一串，还要在真实 GPU 上真刀真枪地跑。**学完你能**：看懂 $z = bf/d$ 并手算深度；说清“没有真值怎么造标签”的三步流水线；明白你们组第 5–19 周的四个实验、人工必做与验收线。

先看一张“海上炼油厂”的照片。FPSO（Floating Production Storage and Offloading，浮式生产储卸油装置）是一种比足球场还长的大船：海底油井采出的原油在船上加工、储存，再卖给运油船——相当于把炼油厂从陆地搬到了海上。它没有腿，不能像固定平台那样站在海床上，那靠什么不被风浪吹跑？答案朴素得吓人：**用巨大的锚链把自己拴在海床上**。锚链一旦断裂，船就会在风浪里“跑锚”，输油管被撕扯、被迫停产甚至泄漏——对油田来说是大事故。所以行业规范要求：系泊系统的每根锚链，都必须定期下水体检。



图 2.1　FPSO（浮式生产储卸油装置）示例：海底油田的“海上炼油厂”，靠系泊锚链固定在作业海域

图片来源：Wikimedia Commons，FPSO 作业照片（CC 授权转载）

体检靠谁下水？靠 ROV（Remotely Operated Vehicle，遥控潜水器）：母船通过一根“脐带缆”给它供电、传视频，操作员在船上遥控它下潜一两百米，沿着锚链一寸一寸地拍。它是本项目的“摄影师”——没有它，就没有后面的一切数据。

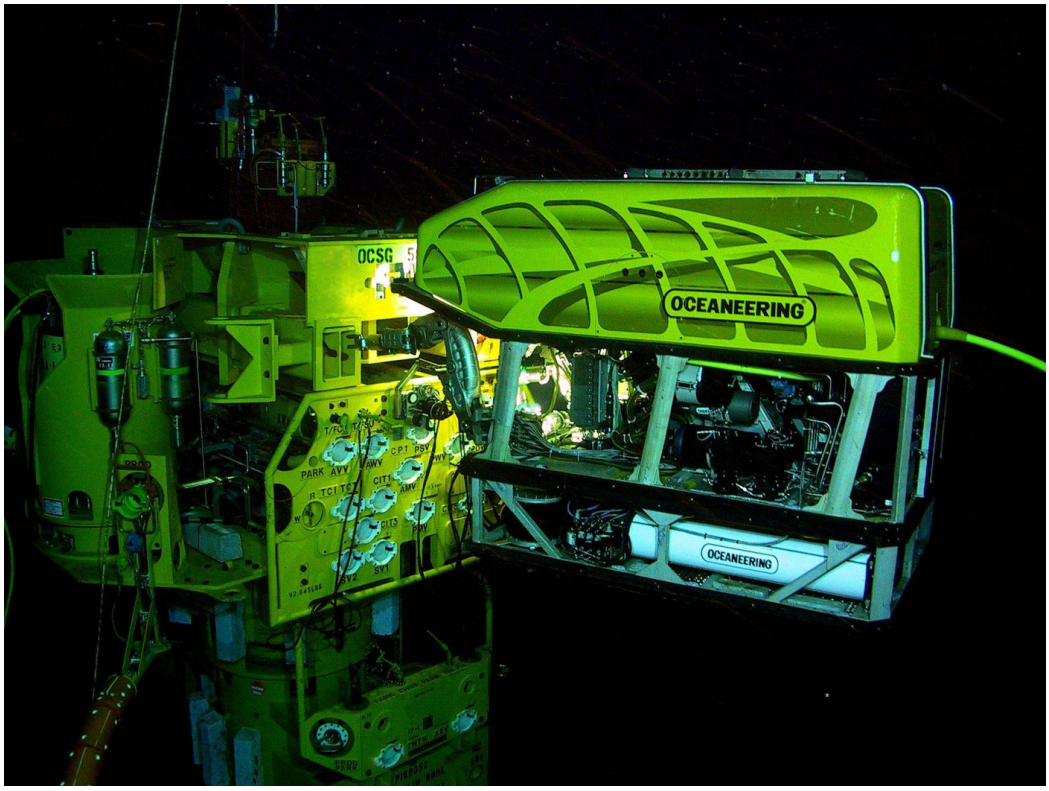


图 2.2　ROV 水下作业示例：由母船缆绳供电与通信，携带相机与传感器下水检查水下结构物

图片来源：NOAA Ocean Exploration / Wikimedia Commons，ROV 作业照片（公有领域授权）

这不是教科书上的摆拍。下面这一帧，就是我们拿到的**真实项目素材**——请先仔细“读”一遍这张图，练一练看图说话的功夫：画面中央斜穿而过的那根亮色“柱子”，就是长满海生物的锚链本体，周围漂浮的白色小点是水中的悬浮颗粒（俗称“海雪”）；左上角叠着一整块字符，叫 OSD（On-Screen Display，屏上显示），是作业设备“盖章”在视频画面上的遥测信息——任务名“HYSY123 FPSO Mooring Line #1 Inspection”（“海洋石油 123”号 FPSO 1 号系泊缆检查）、UTM 坐标、深度 186.0 米、离底高度 ALT 11.8 米、航向 HDG 158.0；右上角是作业方 COSL FUGRO 的标志；右下角是拍摄时间戳。凭这张图你已经可以“报出”拍摄位置和深度——**但画面里每个像素离相机多远，图上什么都没说**。这正是甲方要我们解决的问题。



图 2.3　本项目真实素材：HYSY123 FPSO 1 号锚腿 ROV 检查视频帧——亮色斜条为长满海生物的锚链，左上角 OSD 遥测给出任务名、UTM 坐标、深度 186.0 m、离底高度 ALT 11.8 m 与航向，右上角为 COSL FUGRO 标志，右下角为时间戳

图片来源：本项目真实数据（E:\锚链视频，2024 年 8 月 31 日 1 号锚腿检查视频）

2.1　场景与需求：甲方要“AI 看一眼就知道有多远”

先把镜头拉远。海上油气的开采与运输，是围绕海洋平台与 FPSO 展开的：平台采油、FPSO 储卸、穿梭油轮外运，而把它们与海床“拴”在一起的，正是系泊系统与锚链。这些钢铁巨物常年泡在海水里，腐蚀、磨损、海生物附着，任何一处疲劳断裂都可能酿成大事故，所以每年都要派 ROV 下水巡检。



图 2.4　海洋石油平台示例：海上油气开发的前线，其系泊与水下结构物需要 ROV 定期巡检

图片来源：Wikimedia Commons，海洋石油平台照片（CC 授权转载）

需求就从这里来。ROV 每年巡检拍下的视频是**海量**的：一次检查动辄几个小时、几个硬盘，而看片的只能是少数有经验的老工程师——盯着屏幕逐段回放，判断“这段链环有没有损伤、离相机大概多远”。又贵、又慢、又主观，看视频的人眼连“这段链条离镜头 8 米还是 12 米”都说不准。甲方（中海油，CNOOC）的诉求可以概括成三句话：

1. **构建数据集**：从海量检查视频里，构建出能训练、能测试的立体视觉数据集——视频不是数据，能进训练集的才是数据。
2. **解决标注难题**：水下没有深度真值（第 2.2 节细讲为什么），怎么给每对图像打出“每个像素多远”的标签？这是全项目最硬的一颗钉子。
3. **训练并交报告**：在数据集上完成训练实验，把结果与结论整理成一份能交付的训练报告。

整个项目的大图景，值得单独抄一遍，后面所有工作都为它服务：**输入一对水下双目相机拍到的左右画面，输出每个像素的深度，并知道每个像素的结果有多可信。**“每个像素有多远”叫稠密深度，“有多可信”叫置信度——第 2.4 节你会看到，后者恰恰是自动打标签能不能“自证清白”的关键。

团队配置：项目负责人张洪进，研究助理 1 名，大三本科生 4 个组（第 5–19 周进场，每周投入 6–8 小时），毕设生 1 名负责算法主线。四个组各守一条并行赛道、实验四合并联调，见第 2.6 节；每个人的具体任务书在第三篇（第 9 章至第 12 章）。

2.2　水下为什么难：光、匹配、泛化、真值四道坎

有人会问：立体测距在空气里早就是成熟技术了，自动驾驶车顶的双目相机都能量产，把它直接搬下水不就行了？恰恰不行。水把这件事的全部前提都改了。我们数出四道坎，每一道都对应后文的

一招“化劲”。



图 2.5　浑浊水下成像示例：悬浮颗粒与散射让画面发灰、对比度下降，这是水下视觉的常态

图片来源：Wikimedia Commons，浑浊水下照片（CC 授权转载）

2.2.1　第一道坎：光被水“吃掉”

太阳光扎进海水，会同时遭到**吸收**与**散射**的双重消耗，而且消耗是“挑食”的：不同波长的光被吸收的程度不同。看下图：红光波长最长，最先被水吃掉，几米之内就所剩无几；接着是橙、黄；能传得最远的只剩蓝、绿。这就是为什么你在纪录片里看到的深海一律偏蓝绿——不是导演调的色，是物理。

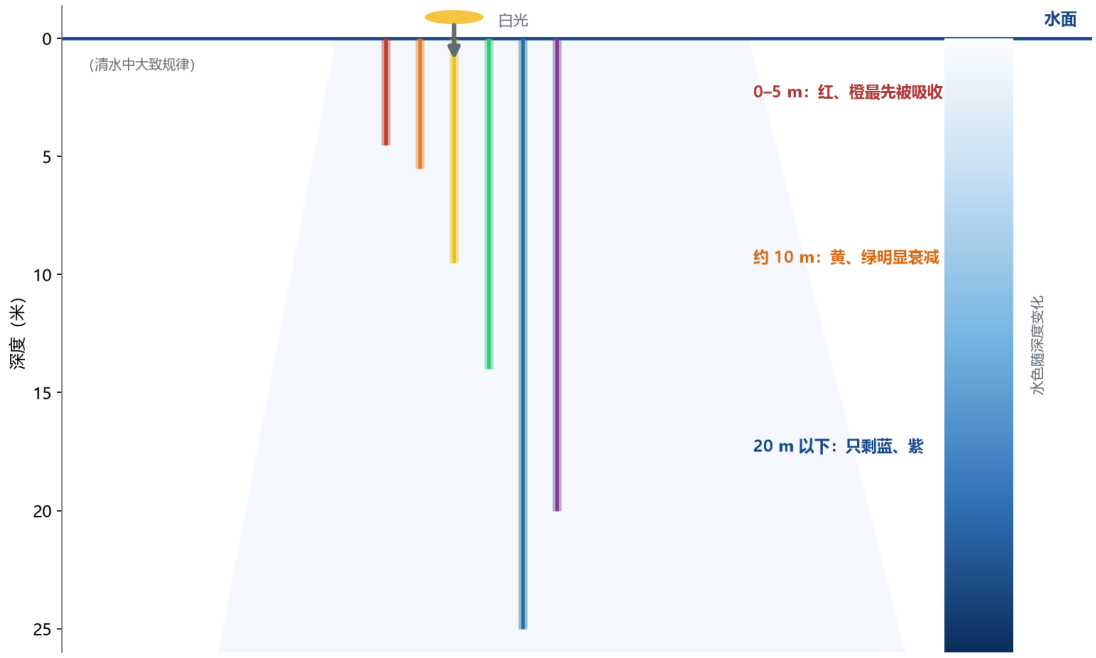


图 2.6　水下光衰减：白光入水后各色光按波长先后被吸收，红光“阵亡”得最快，只剩蓝绿能走远——水下照片偏蓝绿的物理根源

对工程来说这带来两个恶果。其一，**对比度崩塌**：ROV上的补光灯只能照亮眼前一小段，稍远的物体淹没在一盆“蓝绿色的汤”里，就像大雾天开车——水面下的“雾”（backscatter，后向散射）还会把灯光原路弹回镜头，形成一层灰蒙蒙的“水下雾”。其二，**颜色不可信**：空气里我们还能靠颜色、纹理辅助判断“这两块是不是同一个东西”，水下大家全是蓝绿色，颜色线索基本作废。

2.2.2　第二道坎：立体匹配的前提被破坏

第 2.3 节会讲到，双目测距的核心一步叫**立体匹配**：在左右两幅图里找到“同一个点”。它有一个默认前提——**同一个物体在左右图像里“长得一样”**（灰度、纹理一致）。空气里这个前提大体成立；水下，光照经过两段不同的水路，散射与吸收是随机的：左相机眼前的悬浮颗粒与右相机眼前的不是同一批，两束补光的角度也不同——同一个链环，左图里亮、右图里暗，是常态。“长得一样”的假设被水搅了，匹配就会找错对象，深度就会错。

2.2.3　第三道坎：深度网络泛化差

那不用双目，用第 12 讲那种单目深度大模型行不行？也有麻烦：深度网络是在**空气数据**上训练的，水的颜色分布、散射模式它从没见过——把水下照片喂给一个在陆地上百试百灵的深度模型，输出可能完全离谱。这叫**域偏移**（domain shift，训练数据的“世界”和真实使用场景不一样）。解药只有两条：要么用水下数据重新微调（所以我们先要造数据集），要么用对域偏移更鲁棒的基础模型（第 2.4 节的主角）。

2.2.4　第四道坎：没有深度真值

最致命的一道。在空气里，想给一张图配“每个像素多远”的真值，可以用激光雷达（毫米级精度）或结构光设备对着标；实验室里还能用机械臂移动相机精确控制基线。水下这些通通失效：光被水吃掉，激光雷达没了距离；浑水里的声呐分辨率又太粗，对不上像素。结果就是：**甲方提供的海量**

视频，一帧标注都没有——一个标签都给不了你。空手套标签，就是第 2.4 节要讲的“自动打标签”要创造的奇迹。

生活化类比：为什么水下照片都偏蓝绿——顺便理解“挑食”的光

拿一根吸管插进一杯清水，从侧面看，吸管还是原色；换一杯掺了牛奶的水，吸管立刻发灰发蓝——牛奶颗粒把光散射掉了，波长越长的红光“死”得越快。海水介于两者之间：纯吸收（像深水）先吃红光，浑浊（像近岸）再叠加散射。记住这幅图景：**水是光的“红先蓝后”的收费站**，走不远的光就是丢掉的信息。你们处理的真实视频之所以是那种蓝黑色，不是相机坏了。

2.3　双目立体视觉：两只眼睛如何“看出”距离

讲完了难，该讲原理了。好消息是：核心原理简单到可以闭眼复现——因为它就长在你脸上。

知识补齐（重点）：视差——大脑的“测距仪”

人有两只眼睛，间距约 6.5 厘米，这个间距叫**基线**（baseline）。同一物体发出的光，分别进入左眼和右眼，成像位置有一个小小的错位——这个错位叫**视差**（disparity）。你可以现在就验证：盯着眼前的手指，轮流闭左右眼，手指相对背景会“左右跳”；把手指放远，“跳动”就变小。**近的东西视差大，远的东西视差小**——大脑拿这个错位当测距仪，时刻在算你与世界的距离。双目相机就是把这套生理结构换成两台并排的工业相机：基线已知、焦距已知，测出视差，深度就是一个除法（见下文公式）。

把两台相机水平并排、同时拍一个物体，几何关系如下图：两相机光心相距基线 b，物体在深度 z 处，它在左右图像里成像的位置差了 d 个像素——这个 d 就是（像素单位的）视差。

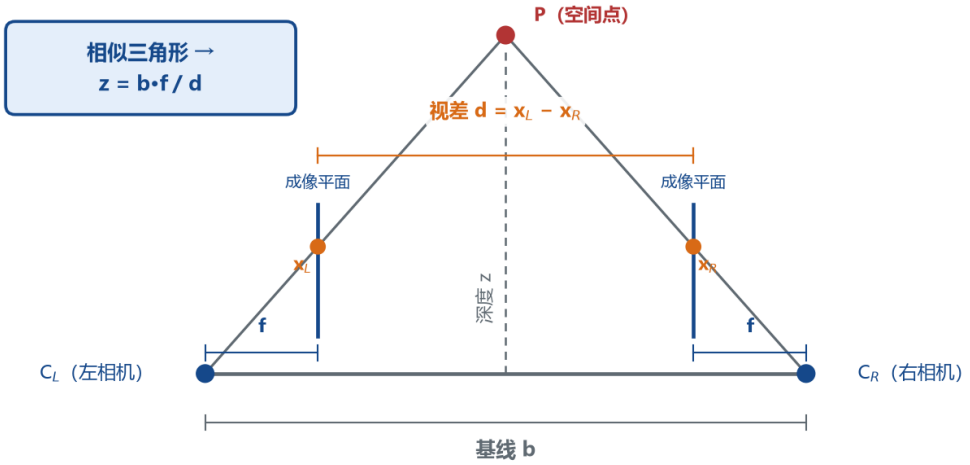


图 2.7　双目成像几何：两台相机光心相距基线 b，同一物体在左右图像中的成像位置错开 d 个像素（视差）；由相似三角形即可反解深度 z

由相似三角形（下图光线的交叉关系），四个量被一条公式锁死：

$$z = \frac{b\,f}{d}, \tag{2.1}$$

人话：b 是基线（米），f 是焦距（像素），d 是视差（像素），z 是深度（米）。深度等于“基线乘焦距、再除以视差”。分子是相机的“体检参数”（装好就不变），分母是每张图要算出来的东西——**测距问题由此变成找对应点的问题**。

这条公式还藏着一条规律：深度翻倍，视差减半。看下图的反比曲线——**近处视差变化剧烈、远处视差“挤成一团”**。这解释了两件事：双目系统擅长看近处、看不准很远；以及为什么视差差半个像素，近距离的深度能差出几十厘米——**视差必须算得准**。

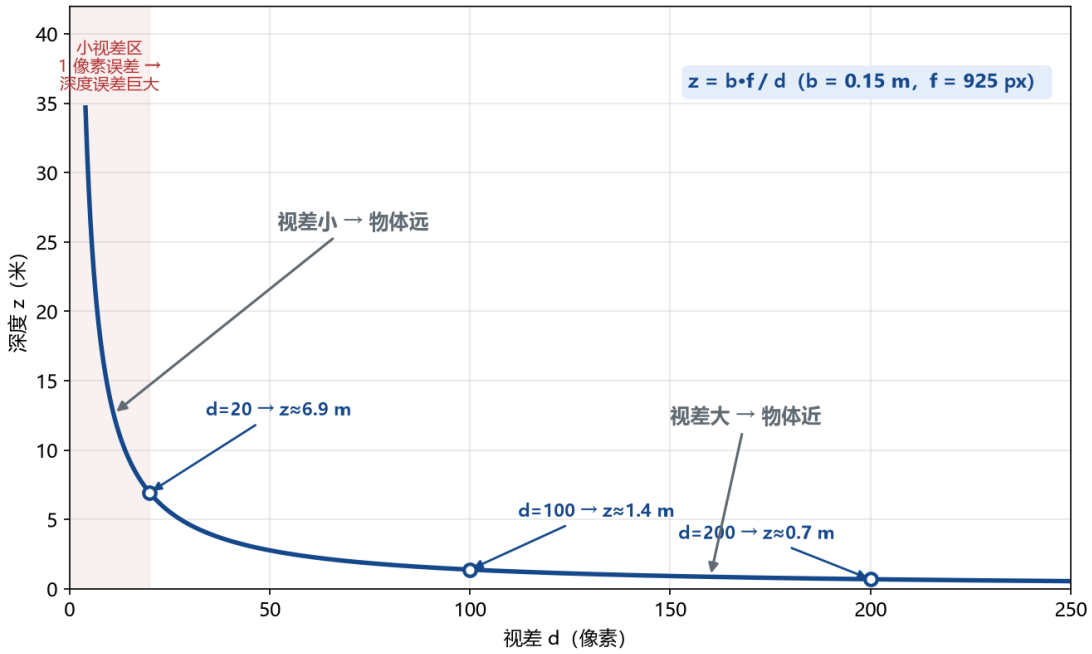


图 2.8　视差–深度反比关系：$z = bf/d$ 是一条双曲线，视差越大物体越近；远处视差挤在一起，微小的视差误差被深度放大

代个数找找感觉。本项目后来从视频里自标定出焦距 $f \approx 925.9$ 像素（第 2.5 节讲怎么来的），数据集里视差中位数约 19.2 像素；假设基线 $b = 0.2$ 米，代入式 (2.1)：$z = 0.2 \times 925.9/19.2 \approx 9.6$ 米——链条在镜头前约十米。你看，只要视差算对了，深度几乎是白送的。

动手时间：一根手指的“跳”——亲手测出自己的视差测距仪

(1) 伸出一根手指，放在眼前约 30 厘米处，背景对着窗户或书架；轮流闭左眼、闭右眼，交替几次——手指相对背景“跳”了一大步。(2) 把手指伸到一臂远，重复——跳动明显变小。(3) 把手指再放远到几乎看不清，跳动几乎消失。**结论：跳动的幅度就是视差的体感**；近跳大、远跳小，正是 $z = bf/d$ 的反比关系。第 (2) 步里你的大脑正在用“变小的视差”告诉你“手指变远了”——你刚刚人工执行了一遍立体匹配。ROV 拖着的双目相机，就是这台“眼睛机器”的钢铁版。

还差最后一块拼图：**立体校正**（stereo rectification）。真实装到 ROV 上的两台相机，不可能绝对水平对齐——总有一点歪斜。而式 (2.1) 成立的前提是：**同一个点在左右两幅图里位于同一行**（这样才能逐行搜索匹配）。校正做的事，就是把左右两幅图各自旋转、重新投影到“理想水平摆放”的坐标系里，让所有对应点的连线变成水平线（这条线叫**极线**，epipolar line）。一句话记牢：**校正 = 把“找同一个点”从全图搜索简化成“同一行里左右找”**。校正好不好，有个硬指标：真实的对应点偏离同一行的垂直像素数，叫**垂直极线误差**——本项目对它的要求是中位数小于 1 像素，实际做到了 0.254 像素（第 2.5 节的验收表）。



图 2.9　双目相机硬件示例：两只镜头并排、基线固定——一台“钢铁版双眼”，水下 ROV 用的正是这类结构

图片来源：Stereolabs ZED 立体相机官方产品图（官网公开资料）

项目连线：一条公式牵出整个项目

本项目：式 (2.1) 是全项目的“宪法”——组 1 数据组做的一切（去畸变、校正、自标定）都是为了把 b、f 弄准、把 d 好算；组 3 训练组训练的网络，本质就是一台“视差计算器 d”；组 4 评测组量的是 d 差了几个像素（EPE）。**六目拼接**（第 一 章）：那边六只眼睛围成一圈拼接全景，这边两只眼睛并排测距——相机阵列的两个方向，靠的都是“多看一个视角”。**数据基座**（第 三 章）：形态不同，方法论相通——都是把非结构化素材（视频/文档）加工成结构化、可检索、可训练的数据资产。

2.4　没有深度真值怎么办：自动打标签（本章高潮）

现在是全项目最硬核的问题。第 2.2 节说了：水下没有深度真值，甲方一帧标签都给不了。可训练一个深度网络，恰恰需要成千上万张“每个像素多远”的标签。先弄清标签长什么样，再讲没有真值怎么凭空“造”出可信的标签——这一节的方法论，是近几年机器人与三维视觉界最重要的范式转移之一，你学会了可以带走，用在任何“没有标注的领域”。

2.4.1　标签本体：一张逐像素视差图

先回答“标签是什么”。本项目里，一张标签就是**一张与原图等大的“灰度图”，每个像素填一个数——该像素的视差 d**（单位：像素）。有了它，代入式 (2.1)，深度逐像素白送。所以“给数据打标”＝“为每对左右图像生成一张视差图”。空气里这张图可以用激光雷达扫出来；水下没有，只能**算**出来——算出来的标签不是真值，是“伪”的，所以得名**伪标签**（pseudo-label）。

2.4.2　三步流水线：教师打标、多重过滤、学生迭代

自动打标签不是玄学，用一句话说清全部：**大模型教师逐帧打标 ＋ 多源一致性过滤 ＋ 学生自训练迭代**。看下图的闭环：